United States Department of Commerce
National Institute of Standards and Technology

NIST Special Publication 785

Proceedings of CIMCON '90

Albert Jones, Editor

NIST Special Publication 785

Proceedings of CIMCON '90

Albert Jones, Editor

Center for Manufacturing Engineering
National Institute of Standards and Technology
Gaithersburg, MD 20899

May 1990

U.S. Department of Commerce
Robert A. Mosbacher, Secretary

National Institute of Standards and Technology
John W. Lyons, Director

National Institute of Standards
and Technology
Special Publication 785
Natl. Inst. Stand. Technol.
Spec. Publ. 785
533 pages (May 1990)
CODEN: NSPUE2

U.S. Government Printing Office
Washington: 1990

For sale by the Superintendent
of Documents
U.S. Government Printing Office
Washington, DC 20402

Table of Contents

TOWARD A GLOBAL ARCHITECTURE FOR COMPUTER INTEGRATED MANUFACTURING

ALBERT JONES AND EDWARD BARKMEYER

ABSTRACT. This paper discusses several issues related to the design and implementation of a global architecture for Computer Integrated Manufacturing systems. It describes separate architectures for production management, information management, and data communications.

1. Introduction

Many companies are trying to integrate their manufacturing equipment with computer-based decision-support systems, control systems, information management systems, and communication networks. A popular name for this process is Computer Integrated Manufacturing (CIM). Our view is that the basis for achieving this integration lies in the design and implementation of a global architecture for CIM.

We believe that such a global architecture must integrate separate architectures for business, production, information, and communications management [BAR89, JON89]. This paper discusses issues related to the design, implementation, and integration of the last three.

2. Production management

Production management functions, as we use the term, can be divided into three groups: manufacturing data preparation (MDP), production planning, inventory control, and shop floor control (SFC). We discuss MDP and SFC.

2.1 Manufacturing data preparation

Manufacturing Data Preparation (MDP) includes those activities which generate and update the data needed to drive the remaining production management functions. MDP functions include design engineering, process planning, NC programming, robot programming, and inspection programming. Table 1 shows the way these functions are carried out today and, we believe, the way they will be carried out tomorrow.

Table 1. Evolution of MDP functions

MDP TODAY	MDP TOMORROW
HUMAN BEINGS	COMPUTERS
COMPUTER ASSISTED	HUMAN ASSISTED
OFF-LINE	ON-LINE
NO INTERACTION	INTEGRATED SYSTEM

Currently, as indicated in the table, MDP consists of many human-intensive, computer-assisted, activities. These activities are typically carried out far in advance of when their output is actually needed. Moreover, there is little or no exchange of ideas or information (before or after the fact) among the design engineers, process planners, and manufacturing engineers who perform those functions.

The trend toward "Just-in-Time" manufacturing and "Concurrent" engineering" is providing the impetus to change all of this. There is a big push to automate and integrate MDP functions as much as possible with computers doing most of the work and humans beings reviewing the decisions. In addition, there is an attempt to begin the execution of these functions closer to the time the part actually goes on the floor. This gives the decision-makers more accurate information about the state of the manufacturing equipment that they may wish to use. This, of course, can dramatically decrease the probability of selecting a piece of equipment which is either over-capacitated or broken. This can reduce shop floor congestion and improves plant performance.

The heterarchical structure described in [HAT85, DUF86] provides the basis for an MDP architecture. In such a heterarchy, there are no supervisors, hence no direct control. All entities are treated as co-operating equals in a negotiation process to carry out a complex task. In MDP, that task is to generate all the manufacturing data for each part. All decisions - best design, alternative process plans, etc. - are reached through mutual agreement and information is exchanged freely among the participants. This exchange of information takes place through the global database. It includes a standard description and definition of the part to be made as input, computer-based methods for executing the various functions, and standard data exchange formats.

The input to MDP consist of a complete part description, a portion of which is used by all of the functions. An international effort has been underway for several years to develop a standard product description. It is called **PDES/STEP** [SMI88]. Version 1 has already been submitted to the international standards community for comments. Second, commercial products are available which perform all of the internal functions. Some are fully automated, but most are not. Finally, there is the problem of data exchange formats. At this time, for example. it is not possible to take the output from an arbitrary design package and use it as input to an arbitrary process planning package. More work is needed to define both the contents and formats for these data structures.

2.2 Shop floor control

Almost all of the proposed shop floor control architectures are multi-level hierarchies. Each module in this structure has one supervisor, but may have many subordinates. The design of these hierarchies is usually based on three guidelines [ALB81]

- Levels are introduced to reduce complexity and limit responsibility and authority,

- o Each level has a distinct planning horizon which decreases as you go down the hierarchy
- o Control resides at the lowest possible level.

It is important to note that, in practice, the decomposition into levels corresponds to the specific arrangement of equipment on the shop floor. In addition, planning horizons are assigned somewhat arbitrarily. Consequently, major differences exist in the number of levels and functions assigned to each level from one implementation to another. At the moment, there are no quantitative methods available to compare different designs or to determine the "best" design for a particular application.

2.2.1 System decomposition

We propose to decompose the system in a somewhat different manner - based on the frequency of interaction and coordination required among the various decision-makers. In this case, the decision-makers are the control modules and the decisions they make are planning and scheduling. That is, given the information provided by MDP, they determine exactly how and when to execute their assigned task. Consequently, we propose [JON90] to mathematically formulate and decompose the planning and scheduling problems, thereby forming a decomposition of their respective decision spaces. It is this "spatial decomposition," that determines the number of levels in the shop floor hierarchy. It will depend on the physical arrangement of the equipment on the shop floor, the amount of coordination and interaction required to manufacture the company's products, and the sophistication of the computer hardware in the facility.

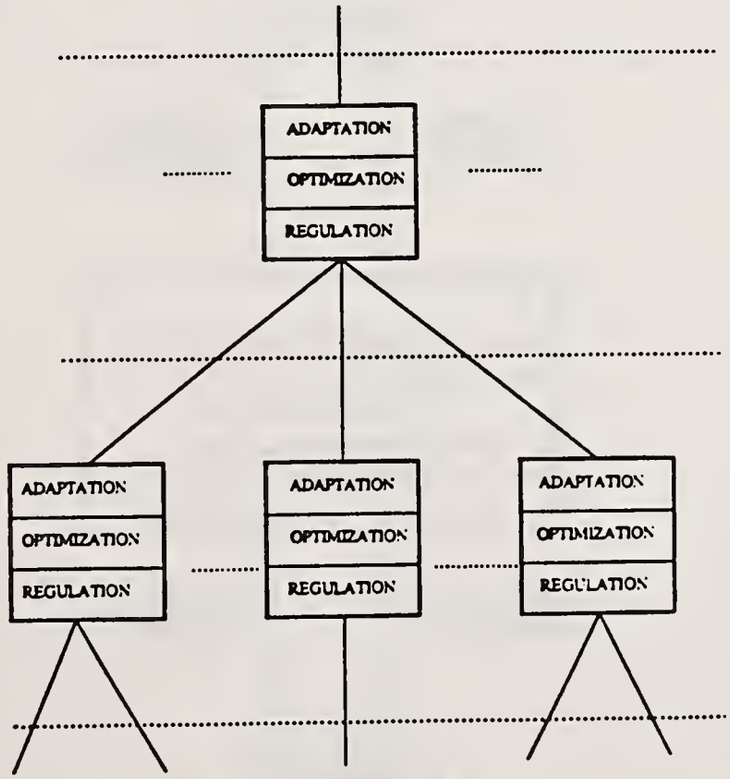


Figure 1. Hierarchical shop floor control system.

We next estimate the frequencies at which the planning and scheduling problems in different levels must be solved and the corresponding information sets updated. We note that, for most implementations, this automatically produces a "temporal decomposition" of the same system. This happens because the frequency with which the planning and scheduling problems are solved in the highest level may be dramatically different from the frequency with which they are solved in the lowest level. We stress that these frequencies should be determined not merely assigned. Villa [VIL86] and Yamamoto [YAM85] have discussed some of the stability issues associated with the selection of these frequencies.

As shown in figure 1, each module in this hierarchical structure performs three major control functions: adaptation, optimization, and regulation. These functions generalize those developed in [ALB81, SAR85, and JON85].

2.2.2 The adaptation function

The adaptation function (the adapter) generates a run-time production plan which will enable each job to be completed within the specified time limits. It is also responsible for changing an existing plan when conditions make it no longer feasible (see fig. 2). This run-time production plan will be one of the alternatives contained in the process plan generated by MDP. It will contain 1) the list of tasks assigned to each subordinate, 2) precedence relations among those tasks, and 3) proposed durations for each task.

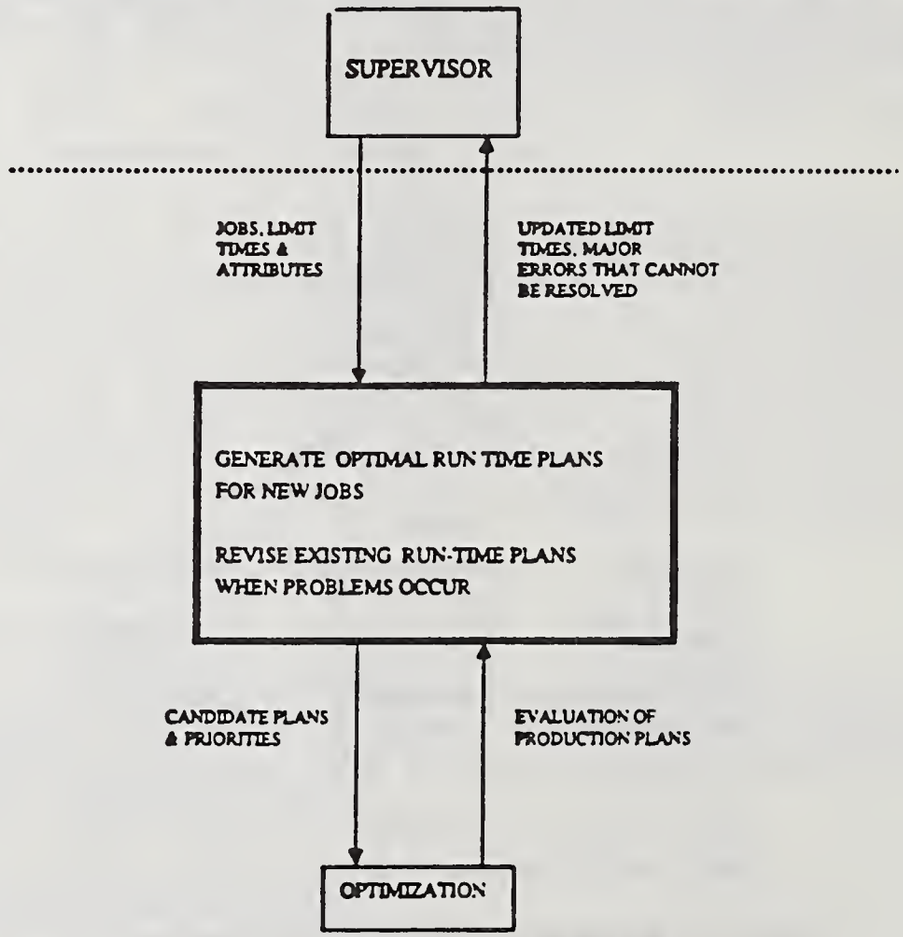


Figure 2. The adaptation function.

For a new job, this involves several steps. First, the adapter retrieves the process plan previously generated by MDP. The process plan contains all the possible alternatives for making this part. The adapter will select feasible ones, based on the current state of the system. This set of feasible alternatives is passed to the Optimizer where they are ranked by one or more performance criteria. The adapter will use these rankings to select the run-time plan. It then requests the optimizer to schedule the part using the selected run-time plan. It will calculate the start and finish times for the job based on that plan and passes them up as feedback to the supervisor. If additional resources are needed, these will be passed up as well. The run-time plan is put into the database for later use by the adaptation function of subordinates.

The feedback information from the Optimization function is used to determine the viability of the current production plan. Whenever shop floor conditions evolve to the point that supervisory constraints cannot be met with the current production plan, a new one must be created. The adapter must now specify a new set of a candidate production plans to meet these constraints. This, of course, may not be possible. When this happens, it will be necessary to negotiate a new set of limit times with the supervisor.

2.2.3 The optimization function

The optimization function (the optimizer) performs three major activities (see fig. 3). It evaluates proposed production plans from the adaptation function. It generates a list of tasks and corresponding start/finish times for subordinates - a schedule. Finally, it resolves, if possible, any conflicts and problems with the current schedule identified by the regulation function.

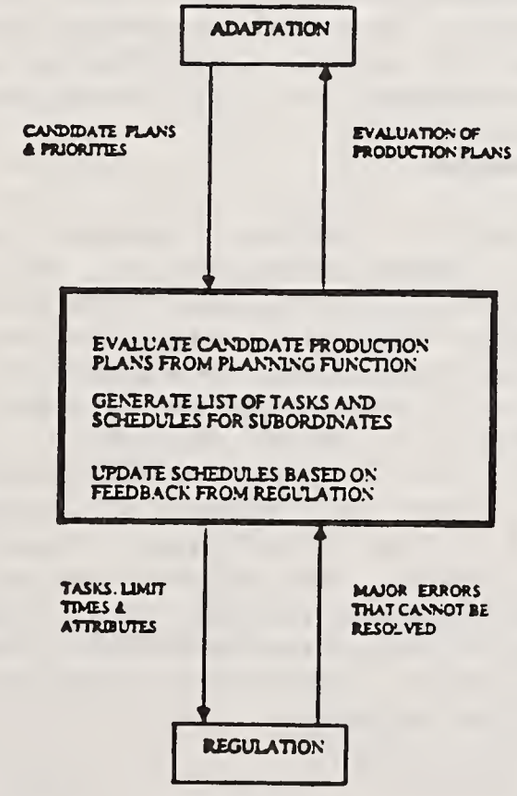


Figure 3. The optimization function.

As discussed above, the optimizer evaluates alternative production plans for each job. This analysis is performed to determine the impact that each plan would have on the rest of the workload (i.e., the active schedule if it were selected). This impact can be analyzed in terms of one or more performance measures passed down by the adapter. They can include tardiness of current jobs, utilization of subordinates, load on the system, and throughput, among others. The optimizer will prioritize these alternatives and pass the results back to the adapter which will make the final determination.

Sometime after the adapter selects the production plan to be used by subordinates in completing the job, the optimizer schedules all of tasks in that plan. The performance measures used in the scheduling analysis must be consistent with those used in generating the plan. A scheduling rule is found which optimizes those performance measures. The resulting schedule is then used by the adapters of each subordinate in generating their run-time plans. These times are also passed up to adapter so that it can update its own estimates of job completion.

The dynamic evolution of subordinate systems can cause delays which make the active schedule infeasible. The regulator detects such a situation and invokes the optimizer to resolve any conflict as quickly as possible. A two step process is envisioned. First, the impact of the delay must be determined using a technique such as perturbation analysis [SUR84] or match-up scheduling [SAL88]. The outcome of this analysis determines the viability of the current objectives, scheduling rule and run-time plan. They remain viable as long as there is enough slack in the original schedule to absorb the ripple effect of the delay. The optimizer can change objectives and scheduling rules, but it cannot change the current run-time plan. In either case, a new schedule can easily be generated using the method described in [DAV88]. Whenever rescheduling is insufficient, a new run-time plan is required. The procedure described above is followed. A new plan is selected, which eventually results in a new schedule.

2.2.4 The regulation function

The regulation function (the regulator) provides the interface between a module and its immediate subordinates (see fig. 4). It has three major activities. It releases jobs to subordinates: monitors subordinate feedback on those jobs; and guides subordinate error recovery. The job release strategy depends on the capabilities of the subordinate. If the subordinate can only manage one job at a time, which is the case with most controllers today, then the regulator will release one job at a time. A new job is released when the previous one is completed. If the subordinate can handle several jobs concurrently, then the regulator will assign jobs on a need-to-know basis. Feedback data from subordinates is used to determine if any unexpected problems have arisen. The regulator has the authority to resolve minor conflicts or error conditions provided the resolution does not violate any of the limit times in the current schedule. When that happens, the problem is passed up to the optimization function and, possibly, the adaptation function for resolution.

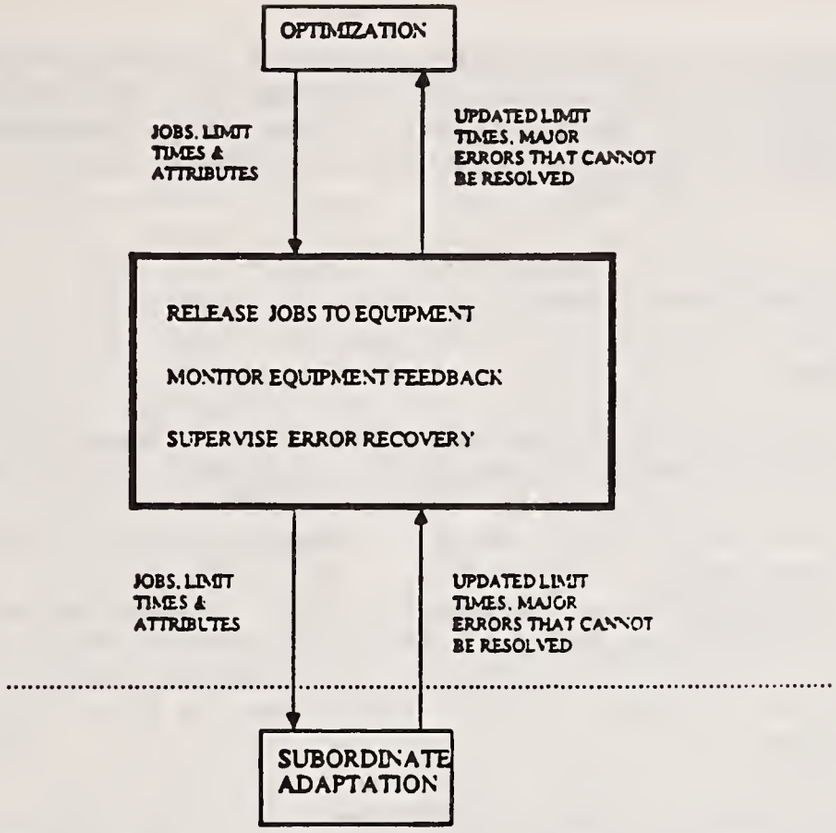


Figure 4. The regulation function.

2.2.5 Impact on Process Plans

The output from MDP that is used most frequently by controllers in the shop floor hierarchy are process plans. They contain the information needed to manufacture, transport, and inspect parts. In the long run, controllers at all levels will use process plans to determine how to plan, schedule and execute assigned jobs. This requires several changes from existing process plans. First, they must have a multi-level structure which parallels the control hierarchy. Second, process plans at each level must provide alternate processing sequences, with precedence relations, through subordinates. In addition, algorithms are needed to perform backtracking so that controllers can recover from problems. Third, plans at different levels must have the same general structure (AND/OR graphs are one possibility). This simplifies the software development and allows for standardization. Finally, since computers will be responsible for processing it, this information must be provided in a consistent, error-free, and machine-readable format.

2.2.6 Implementation issues

Implementation questions fall into two classes: structural and functional. The structural questions are 1) How many levels? 2) With what frequency should functions at each level be performed? and 3) When, if ever, does the structure change? Jones and Saleh [JON90] have outlined approaches to answering all of these questions. The functional question is "What methodologies should be used to execute the three functions-adaptation, optimization, and regulation?" As for the first two, the

method described in [ERS86] combined with the framework described in [DAV88] is most promising. It is attractive because it combines the best features of mathematical programming, expert systems, and simulation. In addition, it can be used to perform the adaptation and the optimization functions at every level in the hierarchy.

3. Issues in data management

3.1 Overview

The main purpose of a data management system in a CIM environment is to support functions in the production management architecture with timely access to all essential data. It is essential to design and implement a data management architecture which is separate from but integrated with the production management architecture. It allows the two structures to be developed independently, provided their interrelationships are well understood. There are, however, many characteristics of a CIM environment which make this approach difficult to implement. Su et. al. [SU86] have discussed this at length, and we now present a brief discussion of these characteristics.

3.1.1 Heterogeneous system environment. The computers and production equipment which make up these CIM systems will be purchased from a variety of vendors over a long period of time. This implies that data is likely to be physically distributed across a network of heterogeneous computers. These local repositories will have a wide range of data access, storage, management, and sharing capabilities. The CIM data management architecture must make these differences transparent to users, i.e., users simply make requests and receive data. In CIM systems, the users are production management functions. In addition, users should not be concerned about the effort required to satisfy their requests. To achieve these goals, the data system must provide users with a common method of accessing information. The data system must deal with the problem of translating requests in the common form to operations on the underlying data repositories, wherever and whatever they may be. This implies transmission and translation of component operations to the appropriate database management systems, assembly of information from multiple sources, and conversion of the information to the form the user expects.

3.1.2 Real-Time Operations. A variety of data is used by computer systems that control shop-floor equipment to make real-time decisions. If that data is not present when it is needed, erroneous decisions or no decision may be made, resulting in processing delays and reduced plant throughput. To complicate matters, some of that data may be shared by several users with different "real-time" access requirements. This implies that the data system must enable asynchronous interchanges of information between production processes which are effectively communicating with one another. This in turn requires the replication of some information units on two or more systems and the frequent and timely updates of those units.

3.1.3 Data delivery and job scheduling. Highly-automated systems are highly dependent on electronic information. It is important to realize that data delivery, like material delivery, takes time and must be included in the

planning of each job. Actual part production cannot start until all of the required information is transferred to the computer responsible for controlling the process performing that production. The notion that this transfer is effectively instantaneous is becoming obsolete as the speeds of automated systems themselves increase. This means that data is quickly becoming a critical resource which must be scheduled. Poor "data scheduling" will lead to delays, bottlenecks, and idle equipment. Therefore, the scheduling decisions made by the data manager have a direct impact on the scheduling decisions made by the production scheduler. This implies the need for coordination between the data scheduling function in the data management architecture and its counterpart in the production management architecture. To the author's knowledge, this type of integration does not happen in any existing CIM system.

3.2 What Constitutes an Architecture

To meet the requirements and constraints described above, a data management architecture must address three major concerns: data modeling, database design, and data administration.

3.2.1 Data Modeling. Developing a "conceptual model" of all the information involved in the entire production management spectrum is critical to the success of any integrated data management system for CIM. Because the amount of information is so large, a "divide-and-conquer" approach to performing the analysis must be taken. Experts on individual functions in the production management architecture will perform the analysis and develop a conceptual information model for each functional area. This results in models for product data, process plans, CAD designs, knowledge bases, etc. Then the resulting "component" models must be integrated into a single "enterprise model." The enterprise model is the conceptual representation of the global information base.

This integration must be based on the identification of common real-world objects and concepts, rather than trying to identify the "common data" elements. A "conceptual model," therefore, must represent the relationships among information units as arising from the real-world objects to which they apply. The organization of these information units, as stored in a given data system, is of secondary importance. Several powerful modeling techniques are available now which allow representation of the real-world objects themselves, as well as the information units which describe and distinguish them. Using such a technique, one can distinguish a concept from its computer representation. This avoids problems with multiple representations of the same concept or similar representations of somewhat different concepts. This capability is vital to the development of an integrated data model.

From this enterprise model, it is possible to extract subsets which represent "views" of the global information base possessed by individual production management processes. But the processes actually want to use data, and they want that data organized in a specific way, which we call an "external view." An external view is an interchange representation and is one of the elements of the interface between data management functions and the production management functions. Generating these views correctly is a matter of considerable current research interest [MAR87]. They are extremely

important because they make the abstract objects disappear and the modeled information units acquire specific physical representations.

3.2.2 Database design

Having arrived at a global enterprise model, we have the problem of mapping this model onto live databases. We must now choose systems, organizations and representations for the information units in the model. This process is called database design. It must result in databases which are consistent with the model and tuned to the timing and access requirements of the production management functions that use them. Because of the evolutionary nature of CIM, it is not possible to start this process from scratch. There are vendor-supplied databases and data systems, which are difficult to alter or augment. There are also "legacies" - large reservoirs of previously developed information which already have an imposed organization. Moreover, even when one has total freedom of design, the design of databases to perform optimally under the multitude of views possessed by different production functions is a black art. The only solution which meets production management requirements and the legacy and autonomy constraints is to divide up the data itself into multiple databases serving specific production functions well. Since much of that data must be shared, two problems result:

a) partitioning - some production functions must simultaneously access information stored in two or more databases, and /or

b) replication - some data must be stored simultaneously in two or more different databases, and maintained consistently.

The available options for the placement of databases in the CIM computer system complex, and for the selection of specific data management systems to support them, are dictated to a large extent by the architecture of the "global" data administration system.

3.2.3 Data **Administration**. The administration portion of the data management architecture provides the data services controlling access to all data:

- "query processing," which is concerned with the interface to user programs, the interpretation of the data manipulation language, and the validation of user transaction requests,

- "transaction management," which is concerned with the identification of databases participating in a given transaction, transaction scheduling and conflict resolution, and

- "data manipulation," execution of the operations on the databases.

There are three control architectures for data administration systems which have been used with varying degrees of success in various business applications: centralized data and control, distributed data and control, and hybrid systems.

The totally centralized approach is the traditional design, the simplest,

and the most workable. Whether this is a feasible architecture for CIM in the long-run is unresolved. There are currently available high-speed, internally redundant, fault-tolerant, integrated centralized systems. But even if such systems can keep pace with the growing demands and time-constraints of automated production systems, the centralized architecture is not workable from the point-of-view of subsystem autonomy. Vendors of design and planning systems, for example, cannot assume that every customer will have such a facility, and will therefore develop local databases and data services to meet the needs of the products they provide. Consequently, the nominally centralized architecture will in fact consist of a collection of autonomous systems copying information to and from a single centralized facility, according to some externally-specified plan. At best, if the external plan provides uniformly for concurrency control, security and the like, what results is centralized data with distributed control.

The canonical architecture for the totally distributed approach (fig. 5) consists of local data management systems which process locally originated and locally satisfiable requests and negotiate with each other to process all other requests. In this case, difficult problems of concurrency control, distributed transaction sequencing and deadlock avoidance occur and must be resolved by committee. While there has been considerable research in these areas, satisfactory solutions have not been found. Moreover, the dynamic evolution of CIM leads to the problem of configuration changes in the complex, which requires informing all existing participants and modifying their distribution information.

The hybrid architecture depicted in figure 6 attempts to combine the best features of both centralized and distributed architectures. Subsystem autonomy and high throughput are achieved by allowing local data systems to process locally originated operations on local data. Operations which transcend the scope of a local system are sent to a centralized "global query processor" for distribution to the appropriate sites. The global query processor acts as a central arbiter for resolving the characteristic problems of distributed transactions and for handling configuration changes in the data administration system itself. There are a number of ways of implementing such an architecture, but they are all characterized by standardized interfaces between the local data systems and the global query processor.

We note that many new commercial distributed data systems are of this hybrid variety, but in general, they have not yet resolved the control problems associated with distributed transaction management to the extent necessary to provide robust support for CIM [THO88]. Unless the local system is aware of potentially global consequences of local changes, and can propagate those correctly, the integrity of the global information bases is always in doubt.

3.3 Our Approach

It is our view that separating the query processing and transaction management functions from the data manipulation functions, producing a layered hybrid architecture, is essential for effective distributed data management in CIM systems. Each module within the layered architecture now manages a queue of database operations resulting from the decomposition of

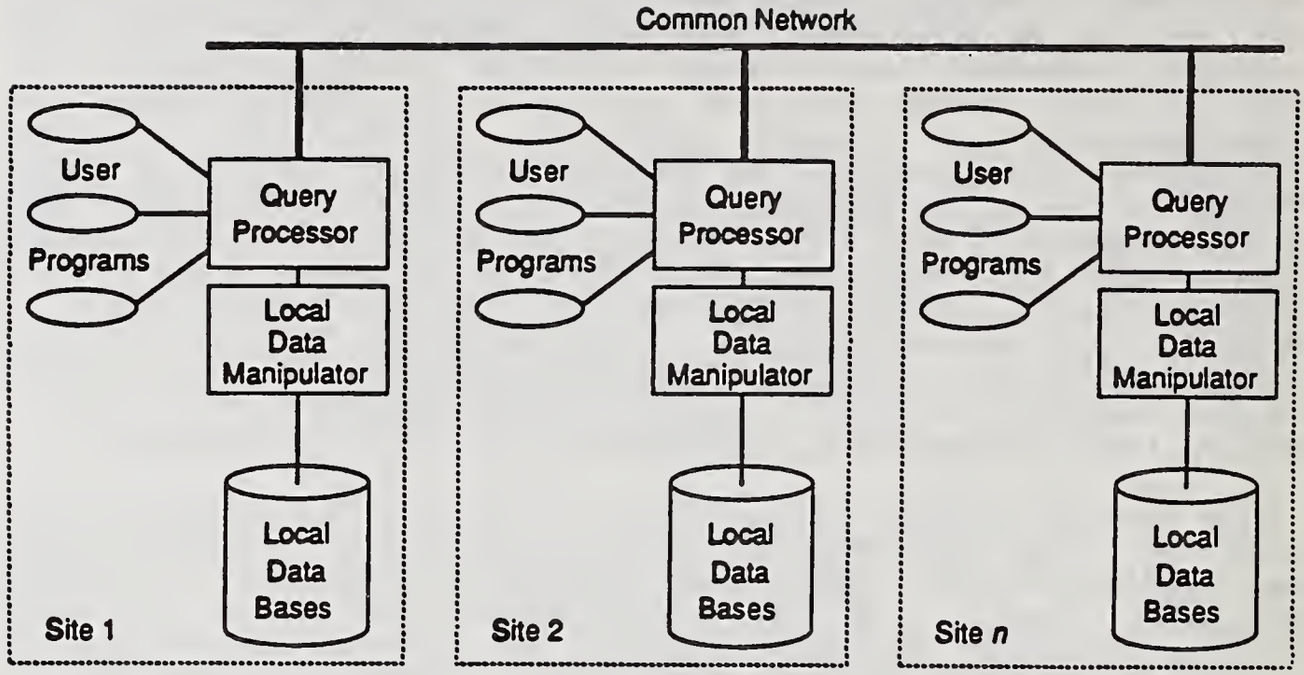


Figure 5. Distributed data system with distributed control.

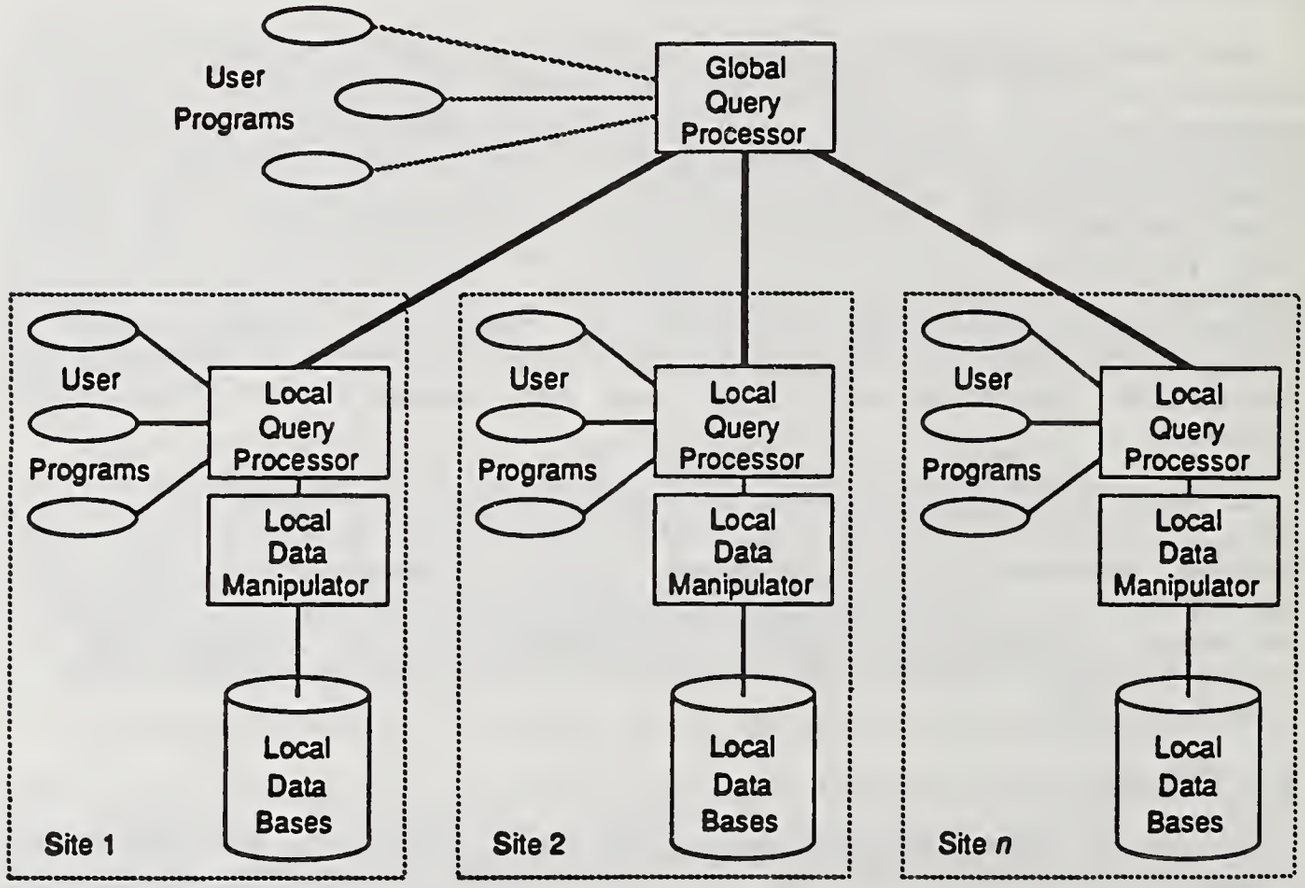


Figure 6. Distributed data system with hybrid architecture.

some complex query. Each query decomposition can be posed as an optimization problem having both static and dynamic characteristics. Techniques used to solve the static problem within a centralized system [CHU86] work equally well here. But, not much is known about approaches to solving the dynamic optimization aspect of query decomposition problems in a distributed environment.

Once this decomposition has been completed, the resulting operations must be scheduled and sequenced. These scheduling problems and job scheduling problems have similar characteristics. Scheduling these database operations is complicated by the difficulty involved in 1) estimating the time required to complete a given operation, 2) obtaining a "due date," and, 3) coordinating the database operations across multiple layers which may be involved in the completion of a single complex query. Little is known about approaches to solving these problems.

These real-time decisions must be integrated into the data management architecture. We have already discussed some of the problems in integrating decision-making into a distributed computing architecture. Furthermore, as noted above, these scheduling decisions must be integrated with the job scheduling decisions that are made in the shop floor control architecture. To the authors knowledge, no research is being done to address this critical problem.

4. Issues in communications

The CIM communications system provides those functions needed to transmit messages between computer programs executing production and data management tasks. In planning an integrated CIM network, we believe that three ideas are fundamental:

1) that the production management and data management programs themselves use one common connection service specification for communication with other programs, regardless of function or location;

2) that the physical networks are transparently interconnected, so that any program could conceivably communicate with any related program anywhere in the CIM complex;

3) that the technology and topology of subnetworks are chosen to provide optimal communications responsiveness for the primary functions.

In this section we describe a CIM communications architecture whose design is based on these ideas. This architecture ensures that ANY production management or data management architecture that is deemed to be desirable can be conveniently constructed with the CIM network as-built.

4.1 Types of communication

Communications can be divided into two classes: those WITHIN computer systems, and those ACROSS computer systems. The first type is often referred to as "interprocess communication" while the second is often

called "network communication."

Interprocess communication is dependent on features of the operating system. Many systems provide no such facility at all, or provide only for communication between a "parent" process and "child" subprocesses which are created by the parent. On such systems the coordination of multiple production management and data management activities is extremely difficult. On the other hand, properly implemented "network communication" software provides for the case in which the selected correspondent process is resident on the same computer system as the process originating the connection. That is, the proper solution for the future is to make local interprocess communication a special case handled by the network software. This solution has the added advantage that all communications by a production management or data management process, regardless of the location of the correspondent process, have the SAME interface.

The accepted paradigm for network communication is the Open Systems Interconnection Reference Model (OSI) which separates the concerns of communication into seven layers [DAY83].

1. Physical layer deals with cables, connectors and signals, and the protocols for controlling access to a shared medium.

2. DataLink layer deals with packaging the signals into elementary messages (called frames), checking for errors in transmission and (perhaps) recovering lost frames. It provides the control and checking for the physical link.

3. Network layer deals with making logical end-to-end connections out of one or more physical connections, i.e., finding a path that gets a message from station A to station B.

4. Transport layer controls and checks the end-to-end connections so that complete messages are delivered in logical order and without losses.

5. Session layer distinguishes separate processes or functions communicating between the same two stations, and implements rules for message flow between those processes or functions.

6. Presentation layer converts between local data representations and interchange data representations.

7. Application layer deals with establishing links between processes providing network service to the actual applications software.

Traditionally, "network communication" has meant concentration on the lower four layers and "exposure" of the Transport layer to production management programs. The important aspect of this model is that it formalizes and separates the logical process-to-process link (in layers 5-7) from the physical network service considerations (in layers 1-4). By exposing only the Application layer service, which implements a common program-to-program communications capability, to the production management and data management programs, we insulate them from the networking concerns. (We note that even

local interprocess communication has elements of layers 1,2,5, and 7.) Consequently, we believe that it is meaningful and proper to build an Application layer interface which is common to ALL program-to-program communications, both "local" and "networked."

4.2 CIM network architecture

A great deal of flexibility is created by implementing the OSI model. On one hand, a single physical medium can multiplex many separate process-to-process communications. On the other hand, a given process-to-process connection can use several separate physical connections with relays between them. This gives rise to the general CIM network architecture shown in figure 7. Ideally, all stations on the network implement common OSI protocol suites in the intermediate layers (3-5) and some globally common protocols for moving data sets in layers 6 and 7. In addition, other standard application layer protocols will be shared among systems performing related functions. The choices of protocol suites in the Physical and DataLink layers and the connectivity of individual stations will vary. They will depend on the physical arrangement and capabilities of the individual stations, and their functional assignments and performance requirements. There may be one physical network, or many. All of these separate physical networks, however, must be linked together by "bridges" that implement the proper Network layer protocols. This results in a SINGLE LOGICAL NETWORK on which any given production management or data management process can connect to other process regardless of location. We note, that because this architecture is layered, multiple subnetworks become transparent to our interprocess communication paradigm.

The Manufacturing Automation Protocols (MAP) concept of one physical bus connecting all factory-floor stations may be appropriate for some manufacturing facilities. It is not, however, general enough to meet all communications requirements of the CIM systems of tomorrow. However, the "enterprise networking" concept [MAP88], connecting MAP control networks with Technical Office Protocols (TOP) engineering networks, demonstrates that the generalized CIM network architecture is, in fact, currently practical. We believe that this will lead customers to demand, and vendors to produce, products consistent with that architecture.

It is likely that emerging physical networking technologies will, in time, make the physical layer standards selected by MAP/TOP obsolete. This will lead to the addition or substitution of subnetworks with new physical and datalink protocols to current CIM networks. Nevertheless, the transparent, multiple subnetwork architecture we advocate should result in little or no impact on process-to-process communication and on CIM networks already in place. At the same time, adherence to at least the layering, but preferably also the intermediate layer protocol suites, in various types of "gateway" machines, provides for the transparent interconnection of subnetworks based on proprietary, or nonstandard protocol suites in the lower layers.

4.3 Technology

There are now many standard protocol suites for the DataLink and Physical layers, and there will soon be more. They all provide frame delivery and

integrity checking; some provide for reliability and recovery, others defer those considerations to the transport layer. The real distinguishing characteristics among these standards are the signalling technologies and the sharing algorithms. Loosely speaking, the signalling technology determines the raw transmission speed, the relative immunity to electronic noise, and the cost. The sharing algorithm determines the nature of network service seen by the station. There are generally three choices:

a) connection to one other station or one other station at a time, with fixed dedicated bandwidth (point-to-point, time- and frequency-division);

b) connection to multiple stations simultaneously, with variable bandwidth with fixed lower and upper bounds depending on the number of stations connected to the medium (token bus and ring);

c) connection to multiple stations simultaneously, with variable bandwidth from zero to the bandwidth of the medium depending on the traffic generated by all stations connected to the medium (CSMA/CD).

In general, engineering and administrative activities, which have infrequent and variable communications requirements, can tolerate and use the type (c) services more effectively. The production control activities, which have frequent and regular messaging requirements, however, prefer type (a) or (b) services.

There are also several "standard" protocol suites for the intermediate layers as well. But in this area, the differences are historical rather than functional. It is clear that the existing intermediate layer protocols will be THE standard in the near future. In the upper layers, standards are still evolving. Here the only problem will be to determine the suite of protocols necessary to a given production management or data management function.

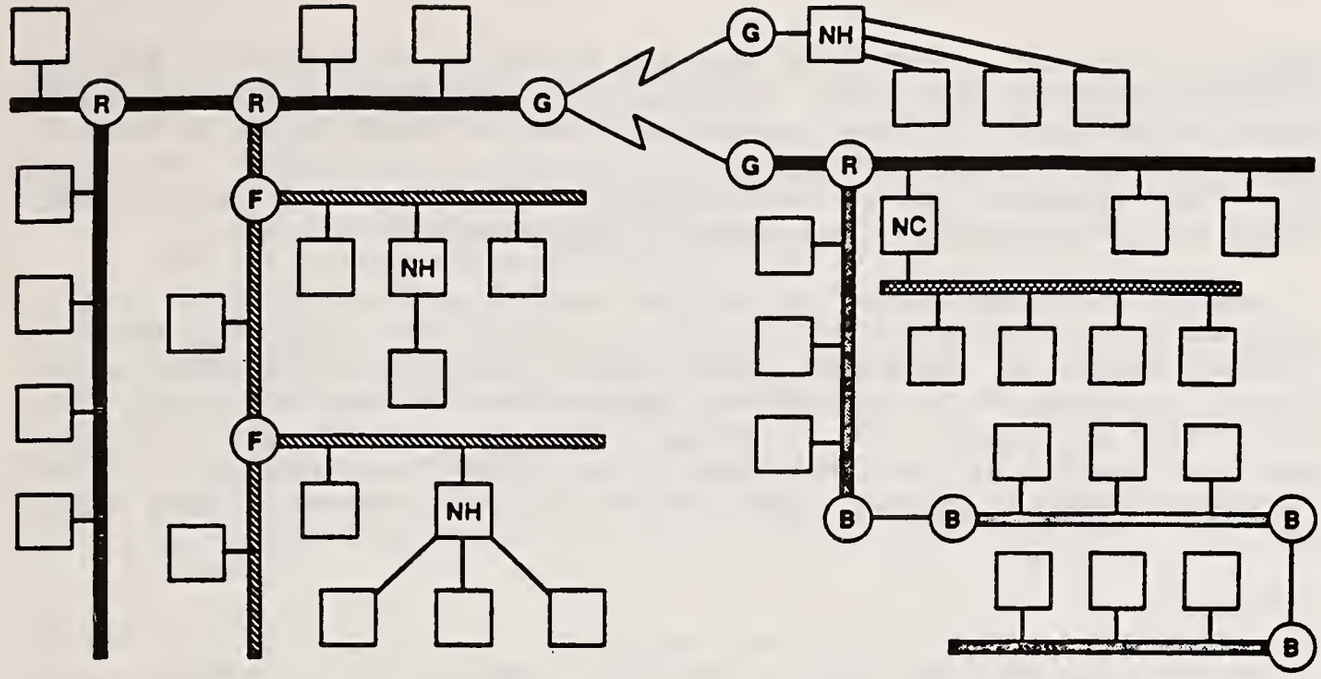
4.4 Topology

Topology is that aspect of network design which concerns itself with the connection of stations to subnetworks and the interconnection of the subnetworks. Topology is at least as important as bandwidth and access protocols in determining the effective performance of integrated networks. Processes which need to communicate frequently should be directly connected, or connected to a common bus/ring, if at all possible. Only two factors should really motivate dividing a network into subnetworks:

(a) the feasibility or cost of connecting all of the potential stations to the same physical network;

(b) the ability of the single network to carry the total traffic load.

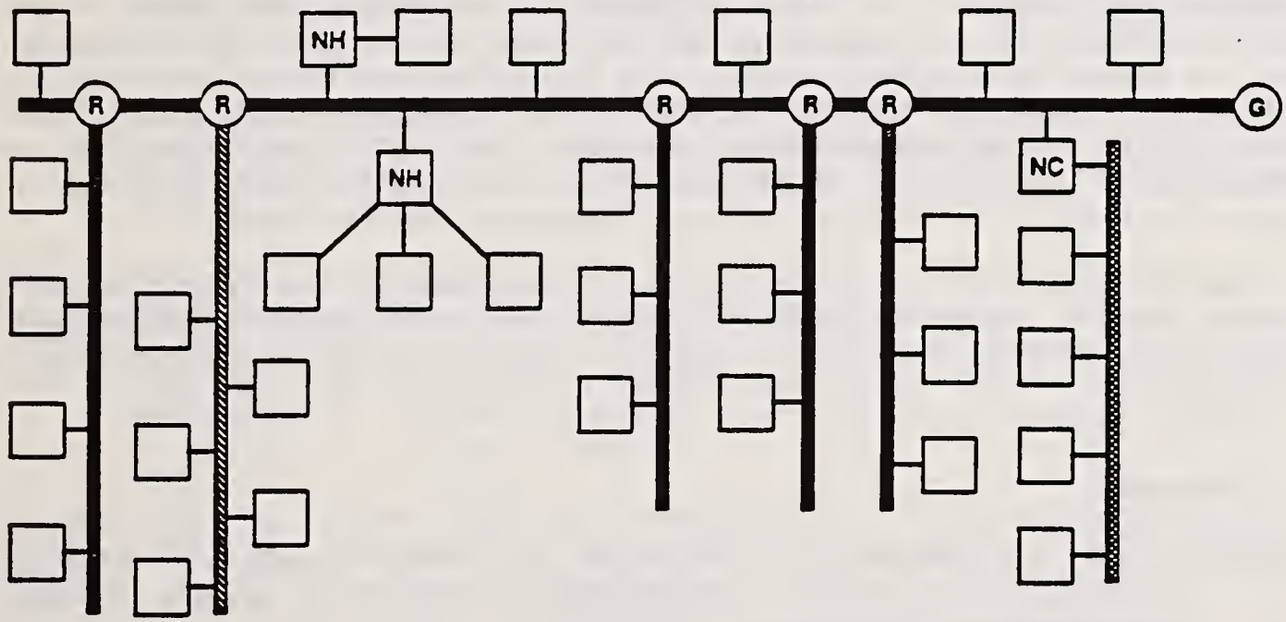
Several varieties of bus/ring networks have limitations on the total number of stations which can be connected, or on the total cable length. When this limit is reached, partitioning is unavoidable. In addition, the performance of most bus and ring networks is inversely proportional to the



Legend:

- F — Filter/Repeater
- B — Bridge (Physical Relay)
- R — Router (Internet Relay)
- G — Gateway (Message Relay or Protocol Converter)
- NH — Networked Host (Attachment point for others)
- NC — (Sub)Network Controller

Figure 7. Generalized CIM network.



Legend:

- G — Gateway to Other Sites
- R — Router (Internet Relay)
- NH — Networked Host (Attachment point for others)
- NC — (Sub)Network Controller

Figure 8. Ideal single site CIM network.

number of stations or volume of messages placed on the network. When the performance of a subnetwork degrades the performance of the primary production management or data management functions using it, it is time to partition that subnetwork or replace the networking technology. The former is usually adequate, easier and cheaper; and adherence to the OSI model should make it invisible to the communicating processes.

Ideally, the generalized CIM network architecture in figure 7 will be implemented in the much more restricted form depicted in figure 8. There is a common "spine" or "backbone network" which connects to ALL subnetworks, although some the of the subnetworks may be directly interconnected. This architecture guarantees that the maximum number of relays on any process-to-process connection is two. While this is not always practicable, it is, in our view, always the desirable goal for the network architecture of a single site.

6. Summary

In this paper, we have asserted that developing a system architecture is one of the key ingredients for a successful CIM implementation. We have argued that such an architecture has three interrelated management components: production, information, and communications. We have discussed many of the problems impeding the integration of the functions within and across these components.

In production management, we examined the problems involved in automating and integrating human decision-making into a hierarchical, distributed, computer architecture. In data management, we discussed the impact of the CIM environment on data modeling, database design, and data administration. And, we argued for a hybrid architecture for administration of that data. In the area of communications, we stressed three principles that should guide the design of CIM communication systems. We used those principles in proposing a generalized architecture for CIM communications which is independent of the production and data management architectures.

Our conclusion is that a great deal of work must be done in the areas of system design, automated decision-making, and other manufacturing-related technologies before the dream of computer integrated manufacturing becomes a reality.

7. References

[ALB81] Albus, J., Barbera, A., and Nagel, N., 1981, "Theory and Practice of Hierarchical Control," **Proceedings of 23rd IEEE Computer Society International Conference.**

[BAR89] Barkmeyer, E., 1989, "Some Interactions of Information and Control in Integrated Automation Systems," **Advanced Information Technologies for Industrial Materials Flow**, 39-57, Springer-Verlag, New York, NY.

[CHU86] Chu, W., 1986, ed., **Distributed Systems, Vol. II: Distributed Data Base Systems**, Artech House Inc., Dedham, MA.

[DAY83 Day, J. and Zimmerman, H., 1983, "The OSI Reference Model," **Proceedings of IEEE**, Vol. 71, No. 12, 102-107.

[DAV88] Davis, W. and Jones, A., 1988, "A Real-time Production Scheduler for a Stochastic Manufacturing Environment," **International Journal of Computer Integrated Manufacturing,** Vol. 1, No. 2, 101-112.

[DUF86] Duffie, N. and Piper R., 1986, "Non-hierarchical Control of a Flexible Manufacturing Cell," **Proceedings of the International Conference on Intelligent Manufacturing Systems**, Budapest.

[ERS86] Erschler, J., Roubellat, F., and Thomas, V., 1986, "Real-Time Production Scheduling for Parts with Limit Times," Technical Report No. 86063, Laboratory for Analysis of Automated Systems, Toulouse, France.

[HAT85] Hatvany, J., 1985, "Intelligence and Cooperation in Heterarchical Manufacturing Systems," **Robotics and Computer Integrated Manufacturing**, 2(2), 101-104.

[JON85] Jones, A., and McLean, C., 1985, "A Production Control Model for the AMRF," **Proceedings of the International ASME Conference on Computers in Engineering.**

[JON89] Jones, A., Barkmeyer E., and Davis, W., 1989, "Issues in the Design and Implementation of a System Architecture for Computer Integrated Manufacturing," **International Journal of Computer Integrated Manufacturing special issue on CIM architecture**, 2(2), 65-76.

[JON90] Jones, A. and Saleh A., "A Multi-level/Multi-layer Architecture for Intelligent Shop Floor Control," **International Journal of Computer Integrated Manufacturing** special issue on Intelligent Control, (to appear)

[MAP88] MAP and TOP Version 3.0 Specifications, 1988, Society of Manufacturing Engineers, Detroit, MI, USA.

[MAR87] Mark, L., 1987, "The Binary Relationship Model - 10th Anniversary," University of Maryland Institute for Advanced Computer Studies, Technical Report UNIMACS-TR-87-50.

[SAL88] Saleh, A., 1988, "Real-time Control of a Flexible Manufacturing Cell," Ph.D dissertation, Lehigh University, Bethlehem, PA.

[SAR85] Saridis, G., 1985, "Foundations of the Theory of Intelligent Controls," **Proceedings of IEEE Workshop on Intelligent Control.**

[SMI88] Smith, B. and Rinaudot, G. (eds.), 1988, Product Data Exchange Specification, **NISTIR 88-4004**, NIST, Gaithersburg, MD.

[SU86] Su, S., 1986, "Modeling integrated manufacturing data with SAM*," **Computer**, Jan., pp. 34-49.

[SUR84] Suri, R. and Dille, J., 1984, "On-Line Optimization of Flexible Manufacturing Systems Using Perturbation Analysis," **Proceedings of the First ORSA/TIMS Special Interest Conference on flexible manufacturing**, 15-17.

[THO88] Thomas, G. et. al. 1988, "Heterogeneous Distributed Data Systems for Production Use," ACM Computing Surveys Special Issue on Distributed Data Systems, **Association for Computing Machinery**, New York, NY.

[VIL86] Villa, A. and Rossetta, S., 1986, "Towards a Hierarchical Structure for Production Planning and Control in Flexible Manufacturing Systems," **Modeling and Design of Flexible Manufacturing Systems** (Elsevier Science, Amsterdam).

[YAM85] Yamamoto, M. and Nof, S., 1985, "Scheduling/Rescheduling in the Manufacturing System Environment," **International Journal of Production Research**, 23 (4).

A SCALEABLE ARCHITECTURE FOR CIM SHOP FLOOR CONTROL

BY
DRS. S. JOSHI[+], R. WYSK[+]
AND
A. JONES[*]

[+]The Pennsylvania State University
[*]National Institute for Standards and Technology

Abstract

This paper presents a generic architecture for the shop floor control of CIM systems. In the paper a 3-level control hierarchy is presented. Task decomposition for each control level is discussed along with a set of control grammars. The architecture is illustrated using an existing CIM system at Penn State University. Examples and implementation issues are also included.

1. Background

Computer Integrated Manufacturing (CIM) has existed, in concept, for several years. While it is still generally believed that CIM can have a major positive impact on U.S. productivity, it is far from commonplace in American factories. The high costs of software development, maintenance, and integration are among the most prominent reasons for our slow evolution to CIM. That is, while many computer packages have been developed to address specific manufacturing tasks like scheduling or process planning, they fail to provide the "hooks" needed for total Manufacturing System integration. This happens because these programs are 1) developed by different vendors, on different hardware platforms, 2) not meant to be integrated with programs from other vendors, and 3) not part of a overall CIM architecture.

This paper addresses these topics as they relate to the generation of a shop floor control system (SFCS) for CIM. The objectives of the paper are to show a generic flexible cost effective, scaleable architecture to plan, schedule and control shop floor actions. This will eventually lead to a reduction in the time required to convert a manufacturing system design into an operational system.

By scaleable, we imply an architecture where the complexity of individual elements comprising the controllers can be adjusted according to the size and function of the system, and retain the flexibility of the system.

2. The Proposed System Architecture

In this paper a prototype hierarchical shop floor control system (SFCS) consisting of several levels is described. The detail of the cell and workstation levels for CIM is also presented. The equipment includes NC machining centers, robot systems, AGVs, or any other computer controlled manufacturing equipment and related tooling resources. We will assume that the cell controller periodically receives a list of jobs, with associated due dates from an emulated Master Production Scheduler. For each job, a process plan will be retrieved either from a

database or as a file transfer. The SFCS will determine the sequence of activities necessary to execute that process plan. This involves the selection of a specific process routing and activities, scheduling of all activities at both the cell and equipment levels, coordinating those activities across the equipment, monitoring activities, job tracking, and some error recovery. To our knowledge, no system exists which distributes and integrates all of these functions within and across levels of a control hierarchy.

2.1 Architecture

A "systems approach" to this problem is essential to achieve the level of integration that is required for CIM. A primary component of any system theory is the definition of the state of the system. For any complex system like manufacturing, the definition of the system state could involve thousands of variables, some discrete, some continuous, many stochastic. As the system evolves in time, those variables take on many different values. The goal is to define a set of decision rules which govern the way in which that evolution takes place. These rules can be generated "off-line" or "on-line", and are invoked to "control" how and when the system changes.

Since the complete problem is too large and with complex interactions, a decomposition of the problem is necessary to create a series of well defined solvable subproblems [1]. The need for decomposition, creates the necessity of providing clear and well defined "hooks" to integrate the various decomposed problems into an integrated whole. The ability to effectively decompose and define the necessary "functional hooks" provides the scaleability in the architecture.

A 3-level hierarchical decomposition of the SFCS is proposed, as shown in Figure 1. Each controller performs 3 main functions - planning, scheduling and control, with typical architectural characteristics for each level shown in Table 1.

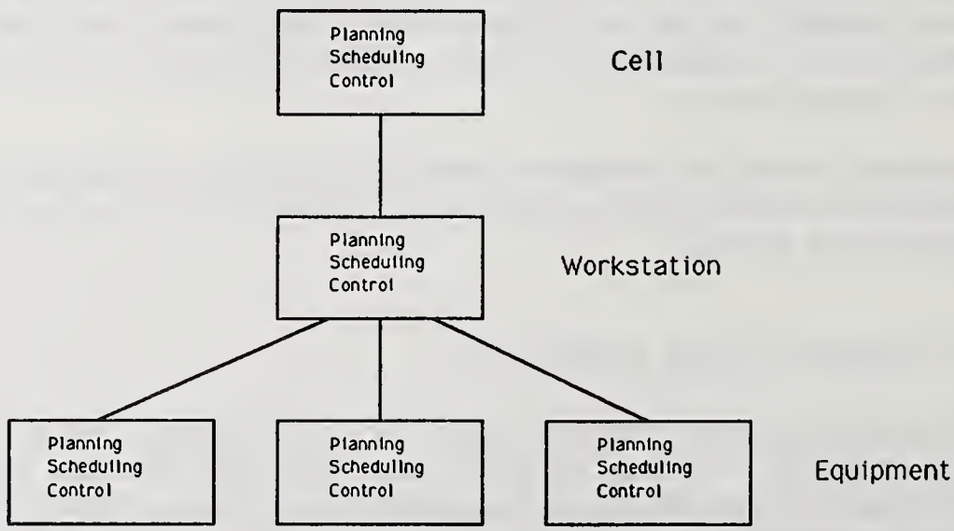


Figure 1. Proposed System Architecture.

Table 1. Typical Architectural Characteristics.

Item	Equipment	Workstation	Cell
EXAMPLES Hardware	Lathe, Mill, T-10 Bridgeport Series I IBM 7S4S Robot	Robot tended Machine Center, Cartrac Material Handling System	Variable Mission System, Several Integrated Workstations
Controller Hardware	Mark Century 2000, Accuramatic 9000, Custom-single-board system	Allen-Bradley PLC-4, IBM-PC/AT, etc.	VAX 11/750, SUN Workstation, etc.
Type Controller	Single-board processors, Machine tool controller, Servo-Controller, etc	PLC, PC Minicomputer	PC, Microcomputer Super-Minicomputer
Language Application	Assembler, Part programming, Robot programming, etc.	C, Lodder logic, Pascal and other sententual languages	C, LISP, FORTRAN and other high level languages
Memory/Size Requirements	8k - 128k RAM plus custom ROM, EPROM, etc.	256k - 1 Meg RAM 1 Meg - 80 Meg Hard drive	512k - 4 Meg RAM 10 Meg - 1Gigabyte Hard drive
RESPONSE TIME	< 10exp(-3) sec	< 1 sec	< 20 sec
MACHINES/ INTERCONNECTS	1 - 1 connect	1 - many 1 - [1,8] Machine tools 1 - [1,50] Material handling	1 - many 1 - [1,15] Workstations

For the proposed SFCS, *planning* is the activity responsible for selecting/updating the process plan to be used in executing assigned jobs and generating revised completion times for those jobs. Cell level process plans contain routing summaries for each job. Each workstation plan contains the operations to be done and the information necessary to generate part programs or their equivalent. At each level, *scheduling* is responsible for evaluating candidate process plans and generating/updating expected start and finish times (the schedule) for the activities in the selected plan. *Control* is responsible for interfacing with all subordinates and other applications. It initiates start-up and shutdown, issues commands to perform assigned activities, uses feedback to monitor the execution of those activities, and oversees error recovery. We emphasize that these functions are executed at different frequencies within the same level and across separate levels.

The equipment level corresponds to the physical devices such as Numerically Controlled (NC) machines, robots, measuring and inspection machines, material handling devices, programmable fixtures, etc. The equipment is usually controlled by a device controller which determines the various capabilities of the equipment. The equipment may perform a single or variety of operations on a part.

The workstation level as defined in the hierarchy, corresponds to several integrated pieces of equipment. At the simplest level, a workstation could be a robot tending a single machining center, along with the requisite fixtures, buffers, and sensors. A more complex workstation could be a single robot tending several NC machines. As an example, a rotational workstation could consist of a robot and several machines (NC turning centers, NC grinders, etc.)

performing various rotational operations. The definition of what constitutes a *unit operations* determines the configuration, size and individual characteristics of a workstation. The notion of a *unit operations* is vital to the definition of a workstation.
A cell is viewed as several integrated workstations, coupled by material transport workstations. The primary activity of the cell is to provide organizational control of the workstations.

A key element of this architecture is the fact that each level controller performs the same functions--planning, scheduling and control. Execution of commands by each controller occurs from a top-down execution. Figure 2 shows the functional breakdown of the control architecture for the various tasks associated with each function at each level. In case of failure and error recovery, the information and requests flow in a "bottom-up" manner, both with each controller and between the different levels as shown in Figure 3. Table 2 illustrates the functions typically associated with each level.

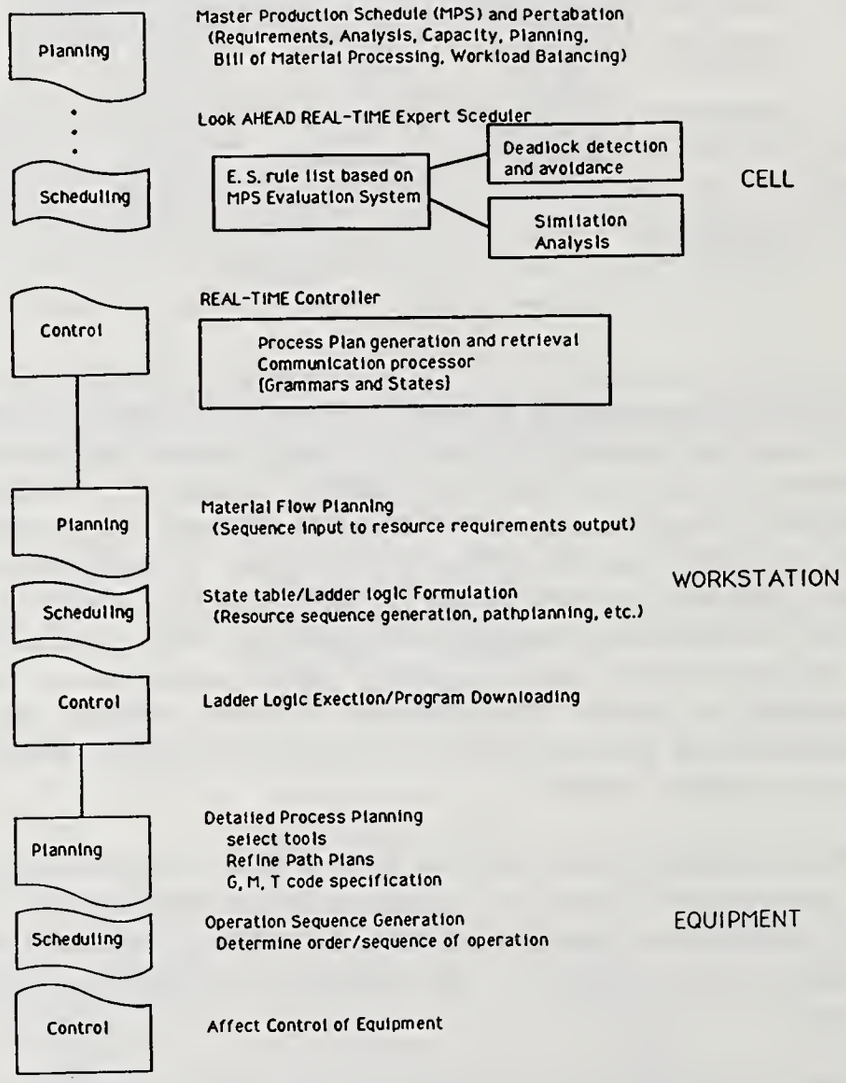


Figure 2. Functional Breakdown of Control Architecture.

Table 2. Function Breakdown of Control Architecture.

Functions \ Level	Equipment	Workstation	Cell
Planning	Tool selection, parameter specification, tool path refinement, G,M,T code, tool assignment to slots, job setup planning	-Resource allocation jobs -Batch splitting and equipment load balancing	Batching, Workload balancing between workstations, Requirements Planning -Task allocation to workstations
Planning Horizon	Millisecs - Minutes	Minutes - Hours/Days	Hours - Days/Weeks
Scheduling	-Operation sequencing at individual equipment	-Sequence equipment level subsystems -Deadlock detection and avoidance -Gantt chart or E. S. based scheduling -Buffer management	-Assignment of due dates to individual workstations -Look ahead ES/ simulation based scheduling -Optimization based tech -Batch sequencing
Control	-Interface to workstation controller -Physical control (motion control at NC and robot pick and place level) -Execution of control programs (APT, AML, etc.)	-Monitor equipment states and execute part and information flow actions based on states -Syncromize actions between equipment (e.g. robot and machine while loading/unloading parts) -Ladder logic execution	Organizational control of workstations, interface with MPS, generation of reports, etc.

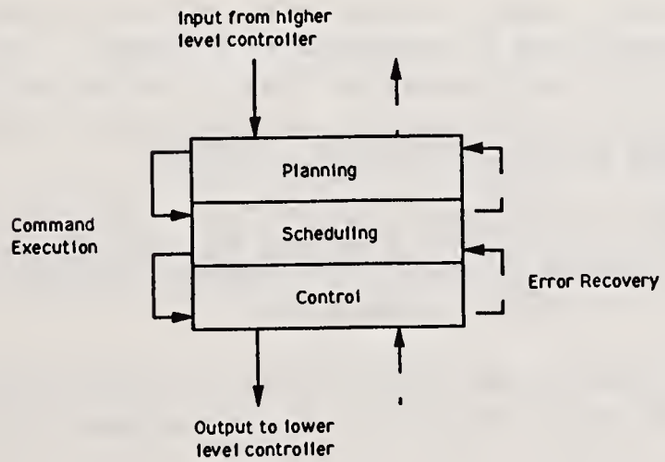


Figure 3. Command Execution and Error Recovery Flow.

2.1.1 Detail Architecture of Workstation Controller

The workstation controller (Figure 4) can be the most complex controller in the proposed architecture. It receives commands from the cell controller, and plans the activities of the various equipment to successfully insure completion of activities in the desired time frame. The expert scheduling system forms the "brain" of the controller. It uses a simulation model coupled with expert system logic to determine future course of action [2,3]. Thus the effects of the decision can be previewed before execution. The simulation is based on actual shop status maintained in the database, incorporates flexible process plans, and utilizes deadlock detection, avoidance and recovery modules to determine "best" part movement strategy. This enables dynamic routing/re-routing of parts in case of machine failure or other unforeseen circumstances.

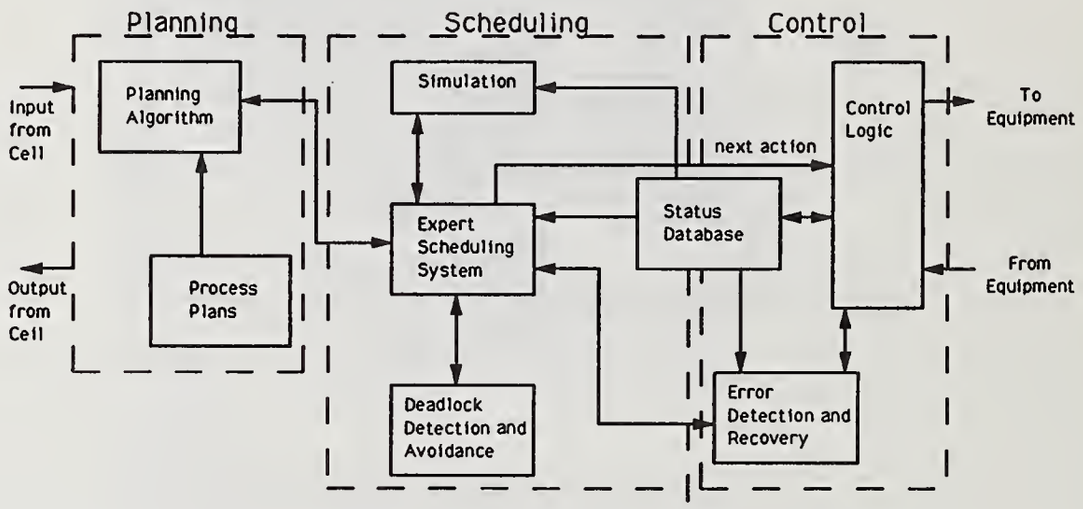


Figure 4. Detail of Workstation Controller.

The error recovery module, houses requisite error recovery algorithms/procedures and is invoked when commands cannot be completed as requested. The error recovery module determines the type of error, viable actions for recovery, interface with scheduler to determine efficiency/performance of the recovery strategy, initiate actions to effect the requisite part and information flow required to complete the recovery process.

The dynamic deadlock and detection avoidance and recovery provides algorithms and procedures required to maintain the system in a deadlock free state where possible, and to initiate recovery from an unavoidable situation [4].

It should be noted that many of the same functions and activities will be performed at both the Workstation and Cell levels. The concept of *unit operations* is the discriminator between these levels.

2.1.2 Process Plan Representation

Process plans play an important role in the definition of the control structure, as well as in defining alternative sequences of operations that must be known by the planning and scheduling modules. The need for alternatives manifests itself in two important ways

(i) it provides the dynamic schedules with various alternatives available at any instant

(ii) the process of creating control software to implement the scheduling actions requires that alternatives be known apriori, so that control software can provide for the necessary links in the control architecture.

A process plan representation is required that is capable of representing all multi-level interactions and possible precedence that occur among the planning and processing decisions.

An AND/OR precedence graph based representation is proposed as a compact representation of the process plan [5]. The hierarchical nature of CIM control makes the graph representation even more attractive. Figure 5 shows the use of hierarchical graphs to represent different levels of process planning required. The cell level process plan indicates that the part must visit four workstations, and two alternative sequences exist ({W1-W2-W3-W4}, {W1-W3-W2-W4}) based on the precedence requirements. Within each workstation there may be alternate routes through machines, and this is represented by the expanded graph of the cell level node. In the example, if the part is processed at workstation, W1, then the two alternative machine sequences that exist are ({M1-M2-M3}, {M1-M2-M4}). The individual machine nodes at the workstation level can be further expanded to represent the alternate tools and tool sequences. In the example, the part can be processed at machine M1 using the two tool alternatives ({T1-T2}, {T3}).

The process plan disaggregation parallels the decomposition in the control architecture. This ensures that the alternatives are provided for decision making at every level.

3. Functional Activities

3.1 Planning

This function determines the best among several candidates that appear in the process plan. The selection is based upon an analysis to predict the impact that each candidate will have on capacity, potential bottlenecks, and other system performance measures. Little has been done in this area. We expect to use the same approach for carrying out this analysis that we are proposing for the scheduling function.

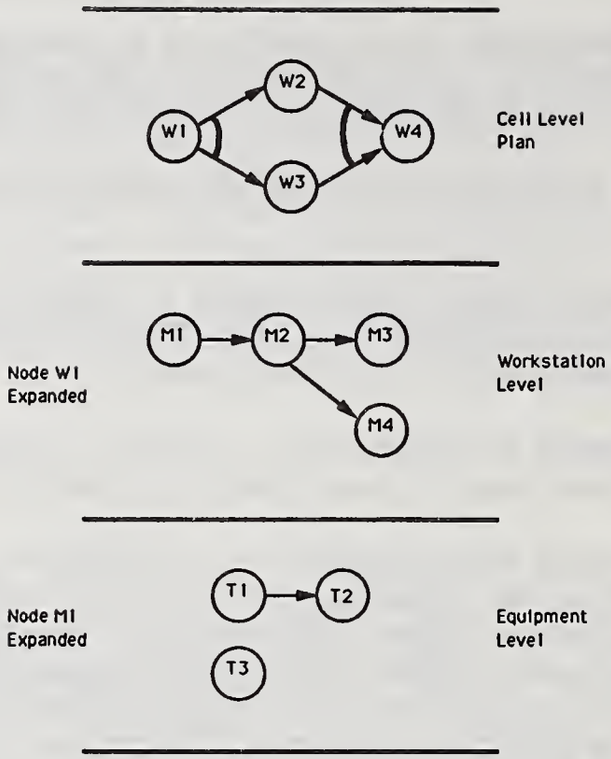


Figure 5. Hierarchical Decomposition of Process Plans.

3.2 Scheduling

Current approaches to scheduling are inadequate for the type of distributed scheduling proposed. They typically do not include the complex constraints arising from material handling and tool management; allow multi-criteria optimization; provide for the coordination of distributed schedulers across hierarchical levels: and do not condition the decision upon the current state of the system.

There are four major issues which must be addressed before this approach can be implemented in a real system: state definition at each level; robust schedules; statistical analysis and rule selection; and error recovery.

State definitions for the cell and workstation have been defined in [6], but little has been done to develop efficient data structures. At this point, we expect to use some type of AI knowledge representation scheme. This allows us to capture the dependencies that exist between the two and to have efficient updating procedures.

The key to developing this type of distributed system lies in the generation of robust schedules". Robust means that the schedule remains close to optimal (relative to some performance measures) over a realistic range of uncertainties. This means that the schedule is rigorous enough to absorb minor perturbations. This will limit the number of *reschedulings* that must be done when something goes wrong.

Error recovery in this context means dealing with the impact of delays. This is a two step process: determine the impact on the current schedule and update the schedule. The first step requires a robust data structure for the schedule which allows one to determine the "ripple effect" of a delay on the current start and finish times. Based on this analysis, the recovery can be a quick fix (in which a few times get revised), a major rescheduling, and/or a major replanning.

3.3 Control

Control provides the mechanisms for executing actions and manages the actual flow of information/parts between the computer/machines.

Concepts from formal language and automata theory [7] are used, and control is exercised as a by-product of grammar recognition. In this case, each control module is viewed as a layered parser (Figure 6). The highest layer parser is an automatically constructed push down automata which interprets command/feedback inputs, and activates the appropriate actions. Examples include the movement of parts in the system, consultation of schedulers, and preparation of reports. The next layer is the Synchronization layer which is activated upon recognition of grammatical constructs reserved for synchronization of machine interactions. A good example is the synchronization required when a robot holding a part in a machine fixture requests that the machine grasp the part. Error actions are executed upon recognition of error conditions and implemented via the use of the error parser.

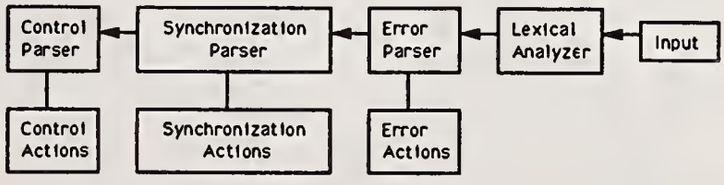


Figure 6. Layered Parsers for Manufacturing Control.

The process involves the following steps [8]: (i) developing a formal model of manufacturing system, (ii) creating context-free grammars for controllers, (iii) developing the parsers, (iv) associating semantic actions with control grammars, and (v) automatic control software generation. A software generation system significantly increases the likelihood of producing correct control software with greatly reduced software development times [9]. Since the control elements of the software will be generated from a formal model of the system, automatic upgrades will be possible when changes to the system layout or product mix occur. Additionally, the use of a formal model to generate run time software can be used to generate the pre- and post-conditions required for theoretical proofs of correctness associated with each action in the manufacturing system. The required "hooks" into the schedules and error recovery are built into the control software, thus creating independence between control and scheduling, and control and error recovery. This allows testing and using any scheduling approach without any changes to control software.

4. A Prototype System

Figure 7 contains a schematic of an operational Flexible Machining Cell (FMC) at The Pennsylvania State University. The FMC currently consists of five Workstations: a Rotation Workstation, a Prismatic Workstation, an Assembly Workstation, a Material Handling Workstation, and a Warehousing Workstation. It should be noted that the various Workstations *couple* and *uncouple* from the Material Handling Workstation through predefined *coupling/uncoupling* points. It should be further noted that the Workstations contain from one (1) to four (4) Equipment components, with the Rotation Workstation containing four (4) equipment elements: a Fanuc M-1 robot, a Daewoo Puma 6 Turning Center, a Pratt & Whitney Horizon V Machining Center and a parts inverter/queue.

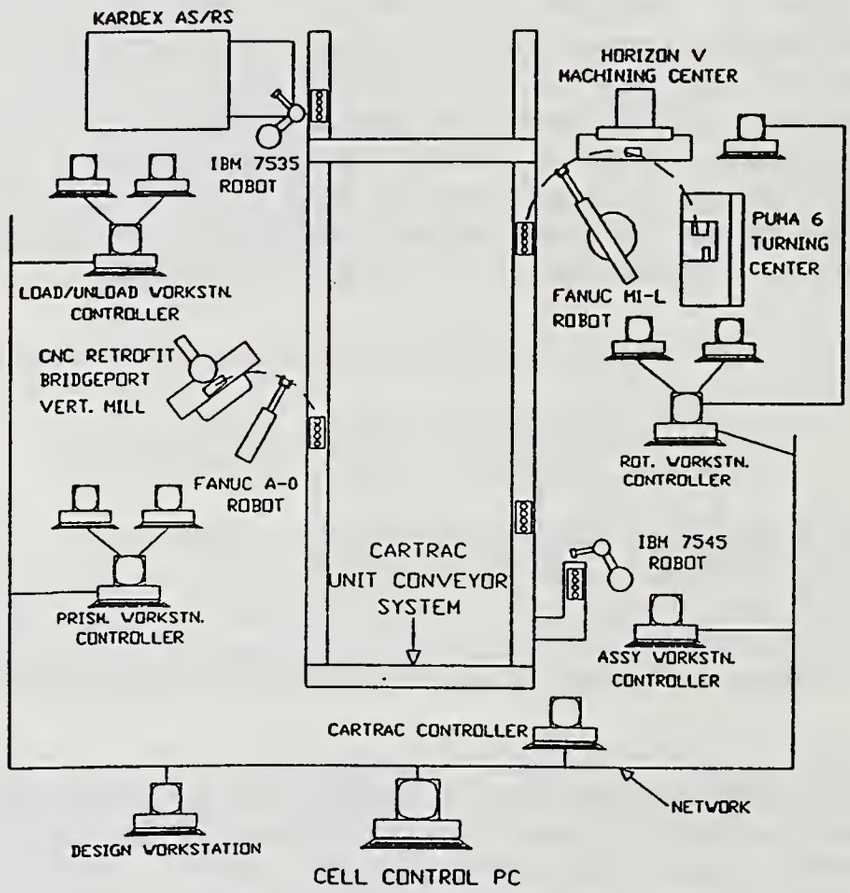


Figure 7. Architecture as Implemented on the Penn State FMS.

Control of the system is initiated by the FMC Controller - an 80286 PC based controller. The FMC controller communicates to the Workstation controllers (also 80286 based PC's) via a DECNET LAN using a common set of grammars. Vocabulary from the FMC controller [10] to one of the machine workstations consists of instructions such as:

MANUFACTURE (batch)

The Rotational Workstation then begins to plan, schedule and control the necessary equipment required to produce the parts in the batch identified by the *pt_no*. The batch quantity is variable between 1 - 4 (constrained by the material handling device in use). Instructions are also sent to the material handling station to deliver the raw material to rotational workstations. The batching task is performed at the cell controller. Table 3 lists a sample of cell-workstation commands. If the part *pt_no* has not been produced in the workstation previously, process planning information will be exchanged. If the part has been produced, a resource requirements list and part program will remain resident in memory. If the Rotation Workstation is currently idle and empty, the coupler (Material Handling Workstation) will be inspected until a part *pt_no* arrives. The workstation controller then coordinates the actions of the various equipment using the set of commands (Table 4) for communicating with the various equipment controllers. The management of the interactions of the equipment may or may not require scheduling (for a single machine) or deadlock detection. If these functions are required, they are included in the workstation controller. The degree of complexity of the scheduling and deadlock detection can be easily adjusted to suit the scale of the FMS.

Table 3. Cell-Workstation Commands.

Cell Workstation Commands	
Command	Returned Values
GET STATUS - Requests the current status of the workstation.	Block of information detailing the status of the workstation. Includes information on the current batch, expected completion time, and inoperative machines.
SEQUENCE (batch) - Requests the workstation to sequence the parts in batch. This function will be used by the cell controller when determining a viable batch configuration.	Runtime of the batch INVALID PART TYPE INVALID BATCH ERROR
MANUFACTURE (batch) - Requests the workstation to start processing batch.	Expected completion time INVALID PART TYPE INVALID BATCH ERROR

The above FMC is operational and produces several parts belonging to two part families (rotational prismatic). Early experience with the architecture looks promising from both a control and scaleability viewpoint. There are however many control problems requiring significant research and development before all *generic* procedures will exist.

Table 4. Workstation Equipment Commands.

Workstation Equipment Commands	
Control Function	Response
Machine Tools	
GET_STATUS - Requests the operational status of the machine.	Block of data including information on the currently loaded part (if any), currently loaded NC file name, and currently loaded tools.
GRASP (part) - Instructs the machine to close the fixture or part chuck. The parameter part gives the part information required for flexible fixtures.	FIXTURE CLOSED PART TYPE UNKNOWN ERROR
RELEASE (part) - Instructs the machine to open the fixture.	FIXTURE OPEN PART TYPE UNKNOWN ERROR
PROCESS (part) - Instructs the machine to start processing the part.	PROCESSING COMPLETE PART TYPE UNKNOWN MACHINE DOWN ERROR
STOP - Stops processing immediately	MACHINE STOPPED
Load/Unload Robots	
MOVE (part, loc1, loc2) - Instructs the robot to move part from location loc1 to location loc2. Part is defined for flexible grippers.	MOVE COMPLETED LOCATION INVALID PART TYPE UNKNOWN ERROR
MOVE (loc) - Instructs the robot to move to the location loc.	MOVE COMPLETED LOCATION INVALID ERROR
GRASP (part) - Instructs the robot to close the gripper.	GRIPPER CLOSED PART TYPE UNKNOWN ERROR
RELEASE (part) - Instructs the robot to open the gripper.	GRIPPER OPEN PART TYPE UNKNOWN ERROR
STOP - Stops the robot immediately	ROBOT STOPPED
Automated Storage and Retrieval System (AS/RS)	
RETRIEVE (part) - Instructs the system to locate part and bring it to the load/unload station. Part could represent finished goods or raw materials.	PART RETRIEVED PART NOT FOUND PART TYPE UNKNOWN ERROR
STORE (part) - Instructs the system to find part's storage location and either bring the location to the load/unload station or take the part to the storage location from the load/unload station.	PART STORED SPACE NOT AVAILABLE PART TYPE UNKNOWN ERROR
LOCATE (part) - Requests the current storage location for part.	PART NOT FOUND PART TYPE UNKNOWN ERROR
ALLOCATE (part) - Instructs the system to allocate space for part.	SPACE ALLOCATED SPACE NOT AVAILABLE PART TYPE UNKNOWN

5. Summary

In this paper, a scaleable architecture for FMS control has been presented. The key elements of this architecture are that each controller performs to same set of functions (planning, scheduling and control), and the degree of complexity of the controllers can be scaled without affecting the control elements.

Various elements of the control architecture have already been implemented at the Penn State FMS Lab, and further research is underway towards implementation of the complete architecture. Early experience has been promising, but further refinements may have to be made to adjust for some implementation difficulties that may be encountered.

6. Acknowledgements

The authors would like to acknowledge Dr. Wayne Davis for some of the ideas in this paper. Further acknowledgements are due to Drs. P. Cohen and C. Harmonosky and several students for their help in the implementation of the architecture.

References

[1] Davis, W. and Jones, A., "A Functional Approach to Designing Architectures for Computer Integrated Manufacturing", Systems, Man, and Cybernetics special issue on Manufacturing, Vol. 19, No. 2, 164-174, March, 1989.

[2] Wu, S. and Wysk, R., "An Application of Discrete Event Simulation to On-line Control and Scheduling in Flexible Manufacturing", International Journal of Production Research, (to appear).

[3] Davis, W. and Jones, A., "Issues in Real-Time Simulation for Flexible Manufacturing Systems", Proceedings of the European Simulation Multiconference, Rome, Italy, June 7-9, 1989.

[4] Yang, N. and Wysk, R., "A Procedure for Deadlock Detection in FMSs", Penn State working paper 89-121.

[5] Mettala, E. and Joshi, S., "A Compact Representation of Process Plans for FMS Control Activities," submitted to IEEE.

[6] Davis, W. and Jones, A., "A Real-Time Production Scheduler for a Stochastic Manufacturing Environment", International Journal of Computer Integrated Manufacturing, Vol. 1, No. 2, 101-112, 1988.

[7] Hopcroft, J. and Ullman, J., "Introduction to Automata Theory, Languages, and Computation", Addison-Wesley, Reading, MA, 1979.

[8] Mettala, E., Joshi, S., and Wysk, R., "CIMGEN - A Case Tool for CIM Development", Proceedings of the FMS Conference, Boston, MA, August, 1989.

[9] Naylor, A. and Volz, R., "Design of Integrated Manufacturing System Control Software", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 17, No. 6, 1987.

[10] Smith, J., Masters Thesis, IMSE Department, Penn State, 1990 (in progress).

CAM-I CIM REFERENCE MODEL

HISTORICAL REFLECTION

Robert E. Boykin III
CAM-I

THE ENVIRONMENT

In 1984, Computer Integrated Manufacturing or CIM was the budding harbinger of success in the international manufacturing arena or so said many corporate visionaries of that era. Virtually all corporations struggling to emerge as a leader in the newly defined 'global marketplace' initiated programs and projects which were hoped would position them for this future. It was soon realized that an unorchestrated approach to CIM produced ineffective advancement of the manufacturing process (typically referred to as islands of automation) and in some cases suboptimized the total process to those before pre-CIM activities.

THE CHALLENGE

This condition of the environment drove many international CAM-I members to initiate activity within the Advanced Technical Planning Committee (ATPC) to address this strategic CIM planning void, ie: The Challenge. The ATPC functions in an advisory capacity to CAM-I's executive board and program staff. ATPC comprises senior representatives from member organizations whose expertise and management vision are focussed on providing advice and guidance to CAM-I activities. This monumental task was accepted and provided focus for numerous meetings, open forums, constructive debates, and supplemental position papers.

It must be appreciated that this document represents only a snapshot in research activities and should not signify the 'End of the Journey'. Many activities within CAM-I are advancing these concepts and in a constructive way the following observations and analysis are made. It must first be understood that the objective of the document was to:

A. Provide an accepted semantic platform for defining CIM;
B. provide a basis of understanding for further research;
C. view insights into the scope and potential of the CIM Concept;
D. And, illuminate the technological developments required to achieve CIM.

The content of the CAM-I CIM Architecture was not intended to provide 'THE ANSWER' to the many complexities of CIM implementation, but rather to establish a common understanding of some of the identified elements comprising the CIM jigsaw puzzle. Maintaining an appreciation of a bias toward an industrial view, the focus on implementation intent, and an awareness of the evolution in CIM since this release; the material remains extremely valuable and relevant to corporate strategic CIM planning.

CAM-I CIM ARCHITECTURE

SCOPE

The planning scope of the document was admittedly 'discrete parts manufacturing oriented' but not limited in application by product size, organizational complexity, process volume, or diversity. Their view was enterprise wide in appreciating the now well acknowledged understanding of the interfunctional activity relationships. Activities ranging from Product R&D and marketing through production and field support were considerated and analyzed. This enterprise-wide perspective of CIM and its strategic contribution to corporate goals necessitated the development of a "CIM Enterprise Architecture".

VIEWPOINT

As stated previously, the view was one from the discrete parts manufacturing industry; but, current research indicates its application base is much larger and can be of positive contribution to many processes and some service industries. The view is truly strategic in vision and not hard-bound within the 'manufacture product' activities of the enterprise. Notably it is primarily focussed on information technology evolution and automation facilitation within the CIM Architecture; but to a lesser degree, does recognize the important contribution of organization, human resources, and policy within a successful CIM implementation.

METHODOLOGY

Several mechanisms were employed within the document to represent and convey understandings and logical meanings to the application and integration of CIM based concepts. First, a high level 'Model of a CIM Enterprise' was developed (Reference Attachment A). Many functions and activities were grouped together to simplify the model at this high level.

Detailed descriptions of the major building blocks are provided and assists in the establishment of common semantics and comprehensions. Within this architectural representation, five basic views or structures were developed to examine the interdependencies and relationships of these structures to each other (Reference Attachment B). Each of the five structures is explored in greater detail within Attachments A - E of the CIM Architecture and employed IDEF0 modeling variations, ANSI 3 Scheme models, and the ISO seven layer open system interconnect standards. The concluding sections of this research work are devoted to presentation of the technical issues, justification, and management challenges which are presently serving as a foundation for continued research within the CAM-I Program structure.

CAM-I Architecture Analysis

In cooperation with the international efforts within ISO TC184, CAM-I provided the 'CAM-I CIM Architecture' for benchmarking other similar research activities toward achieving a internationally recognized conscientious on CIM Architecture(s) and supporting structures. A review was commissioned by ISO TC184/SE5-WG1 and contracted with The Johns Hopkins University. Their results, conclusions, and recommendations were made available in November 1989.

ANALYSIS REVIEW

It first must be stated that the intent of the CAM-I CIM Architecture document and that of WG1 initiatives are different but complimentary. Therefore direct applicability to WG1 research should not be anticipated. Within the critique it becomes quite apparent that the concept scope is not similar. First, the CIM Architecture proposes an enterprise-wide view while the review is admittedly limited to the 'product engineering functions through manufacturing engineering and production' vision of CIM. Secondly, the review mandates a 'computer processable description' while the belief of many is that the strength is within the integration and optimization of enterprise activities rather than the concentration on computerization. And lastly, the CIM Architecture was intended to be of a somewhat generic level allowing for the uniqueness of individual implementation environments. Detailed structural levels were not provided but have been further developed within CAM-I programs.

The viewpoint of the contracted review should be perceived positively but incomplete. It was stated and is reinforced within CAM-I Programs that no one model can address all the CIM complexities for each activity within an organization. Some of the latest research indicates the need for two or more related models to provide this vision of success. Additionally, it was recognized within the review that the requirement for a CIM Architecture varies with each researcher. CAM-I's ATPC was attempting to establish a base line of research understandings and a vision regarding CIM. The review was seeking to identify the 'things that should be done and the relationships that would need to exist within a CIM enterprise'. The architecture of a CIM enterprise should be viewed as a blueprint for the future, not a detailed migration strategy or strategic planning roadmap. These elements are best left to the individual implementation requirements and restrictions.

CONCLUSIONS AND RECOMMENDATIONS

In summation, the review by The Johns Hopkins University staff was objective and constructive to the 'CAM-I CIM Architecture'. The difference in document intent and objectives should be noted but viewed complimentarily to the diverse semantics and concepts explored within CIM. The CIM Architecture is a snapshot in time, a benchmark along the CIM Journey. The general recommendation of the review was that the models were not adequate to meet the objectives of WG1. The following improvements were recommended:

- o Development of a definitive organizational structure to support CIM in the new manufacturing era.
- o Development of the product model concept as well as a way to manage and control all information needed throughout the product life cycle.
- o Development of a process architecture.

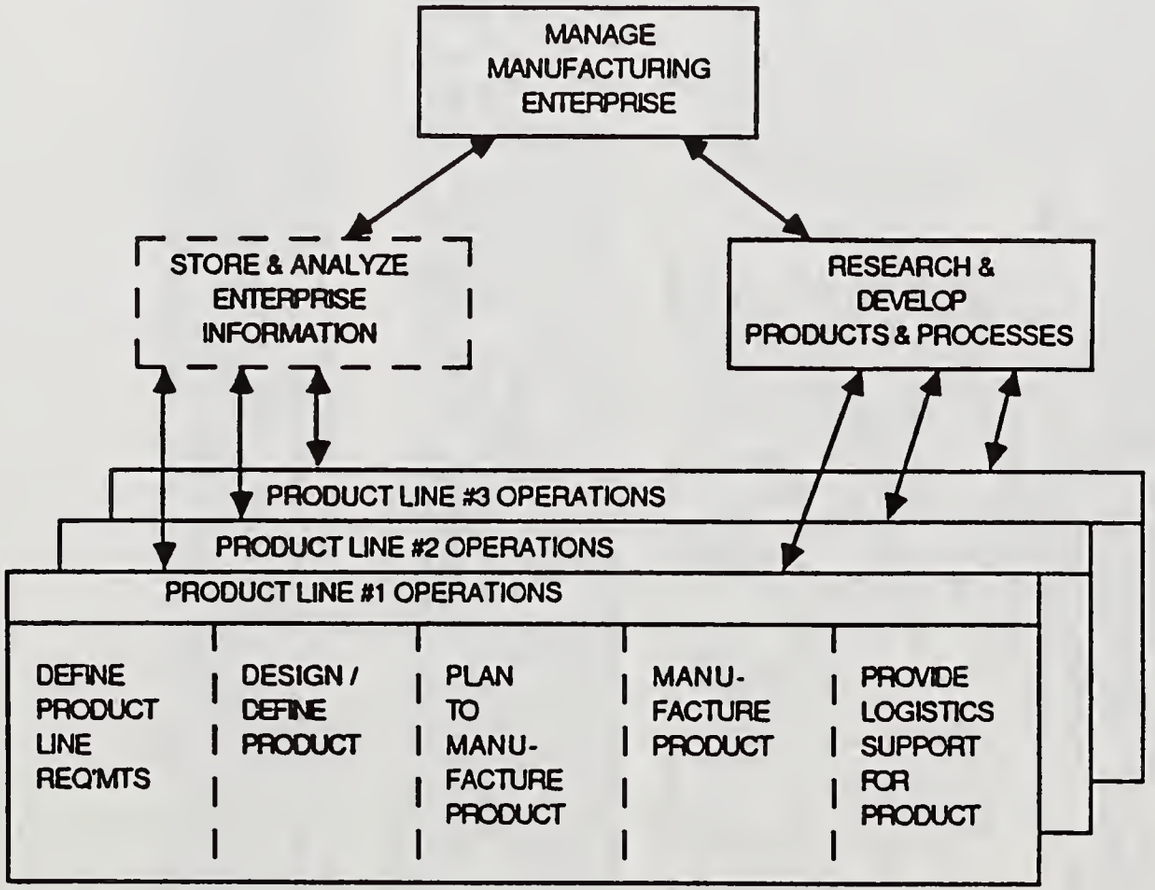
CAM-I Supplemental Research Activities

The CAM-I ATPC CIM Architecture served its purpose well. Several research programs were initiated within CAM-I to focus on many of the critical CIM components and concepts. These programs are conducting leading edge research into the management and strategic planning challenges of the integrated enterprise (Computer Integrated Enterprise Program), the justification and activity costing issues (Cost Management System Program), and optimization of the product and manufacturing process (Product Optimization Program).

Specifically, the Computer Integrated Enterprise (CIE) Program has assumed the proactive role in defining CIE and activity relationships across the entire spectrum of the enterprise, modelling these interdependencies with a focus on the informational technology requirements, development of a strategic plan for implementation, and addressing the organizational challenges and management of this new technological evolution. The CIE Program has confirmed the enterprise-wide perspective of the CAM-I CIM Architecture. But CIE is viewed as much as a management and cultural challenge as the technology itself. CIE has been defined as "a management approach to the integration of enterprise activities"; thus, recognizing several critical elements. First, CIM or CIE must be structured to support the goals and objectives of the enterprise at the highest level. CIM or CIE mandates senior executive commitment and the enlightened support of each stakeholder (employees, stockholders, the community, etc.).

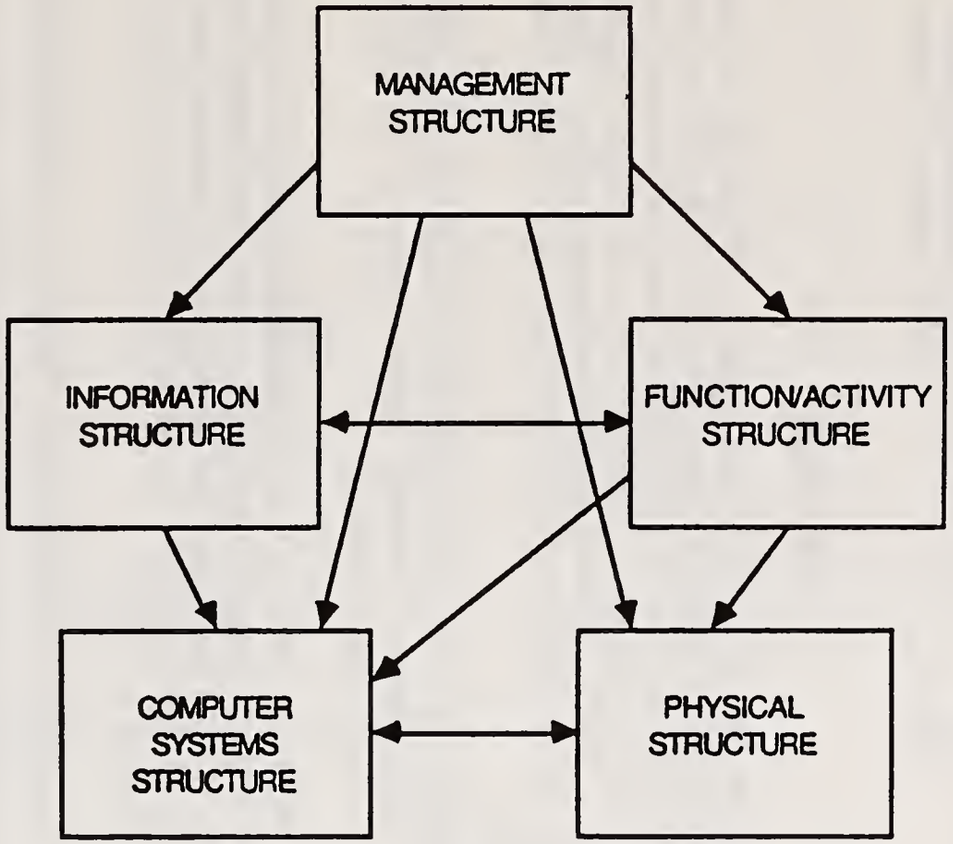
CIE Sponsors have been researching CIM/CIE structures and World Class Best Practices for two years and are actively in the process of synthesizing this research into a generic conceptual framework. This initiative is intended to provide an industry perspective and focus on implementation. How we link corporate goals to the strategic technology plan and develop a feasible migrate strategy to this competitive nirvana are but two of the areas of significant investigation.

In summary, the CAM-I CIM Architecture provided a definitive work which has served as a milestone for critique and additional research. Without this documented understanding, progress would not be focussed to the level it is today and this very meeting may not have occurred. Judge not this work critically but appreciate the thought and courage in establishing a plateau for advancement.



Model of a CIM Enterprise

ATTACHMENT A



Relationship of CIM Structures

ATTACHMENT B

A REFERENCE MODEL FOR COMPUTER INTEGRATED MANUFACTURING FROM THE VIEW POINT OF INDUSTRIAL AUTOMATION

Clyde R. Van Haren
James River Corporation
Neenah, Wisconsin 54956

and

Theodore J. Williams
Purdue Laboratory for
Applied Industrial Control
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

This paper describes the CIM Reference Model developed by the International Purdue Workshop on Industrial Computer Systems. This work is based on a major research project carried out by the Purdue Laboratory for Applied Industrial Control in cooperation with many companies of the US and Canadian steel and paper industries. Both organizations are based at Purdue University, West Lafayette, Indiana. This paper will present a short overview of the complete model which has recently been published in full detail in book form [35].

The model combines the information and control hierarchy with the data flow diagram and a new implementation hierarchy view to present the CIM system, its architecture, its tasks and their implementation.

The model is restricted to the so-called automatable functions of the plant and all functions considered to require human innovation are considered beyond what can be mathematically modeled. It thus offers one way of separating the computer based functions in a CIM system from those which must be carried out by humans, a problem which has plagued the CIM field since its initiation.

This paper will also present a survey of other CIM Reference Models existing in this field.

A SURVEY OF AVAILABLE REFERENCE MODELS FOR CIM

The success of the International Standards Organization (ISO) in the development of series of a communications standards through the use of its Reference Model on Open Systems Interconnection, the OSI/ISO model [2]*, has encouraged many groups to develop and apply such models to other problems. The International Purdue Workshop on Industrial Computer Systems, based at Purdue University, West Lafayette, Indiana, USA, has carried out such a development for computer integrated manufacturing (CIM) as applied to all industries.

---

*Literature References are indicated in this paper by brackets. Parenthesis are used for parenthetical expressions or enumeration.

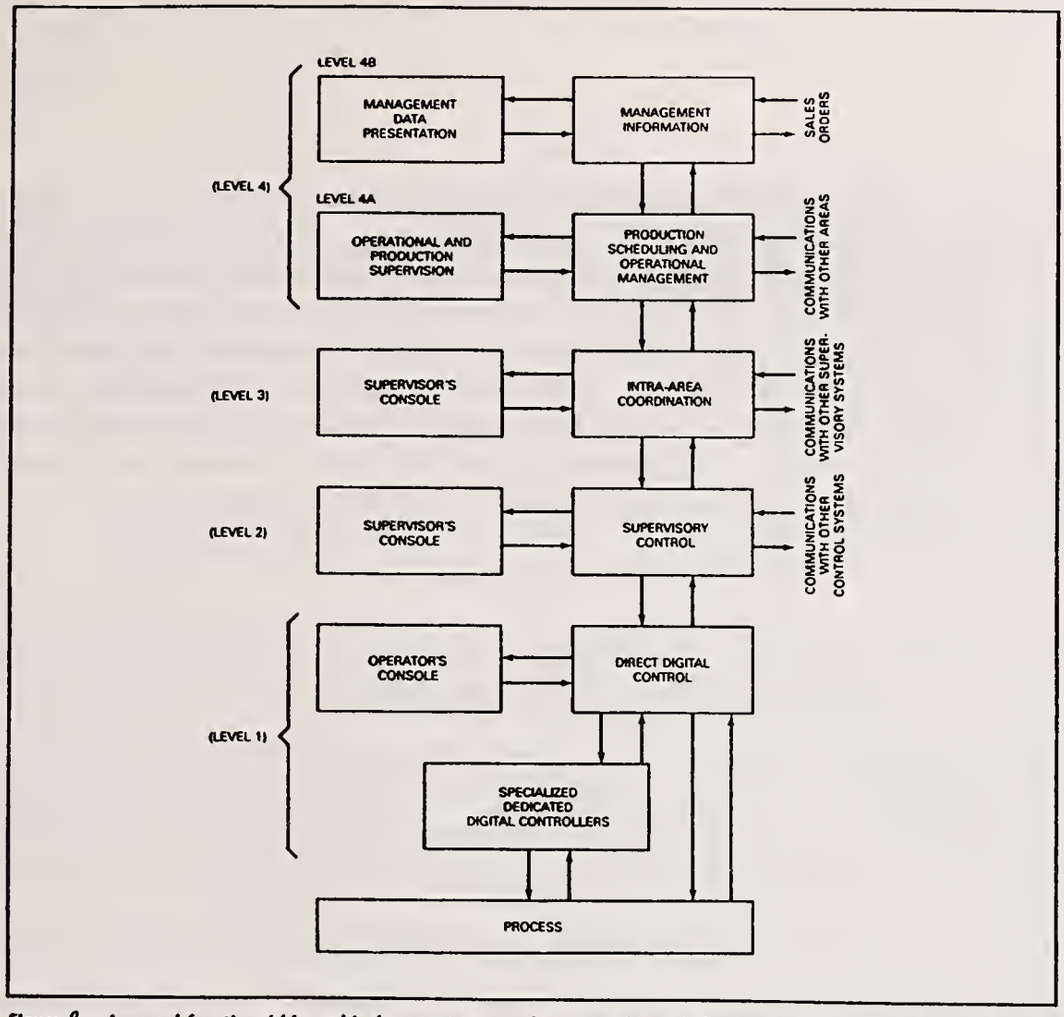


Figure 2 Assumed functional hierarchical computer control structure for an industrial plant (continuous process such as petrochemicals).

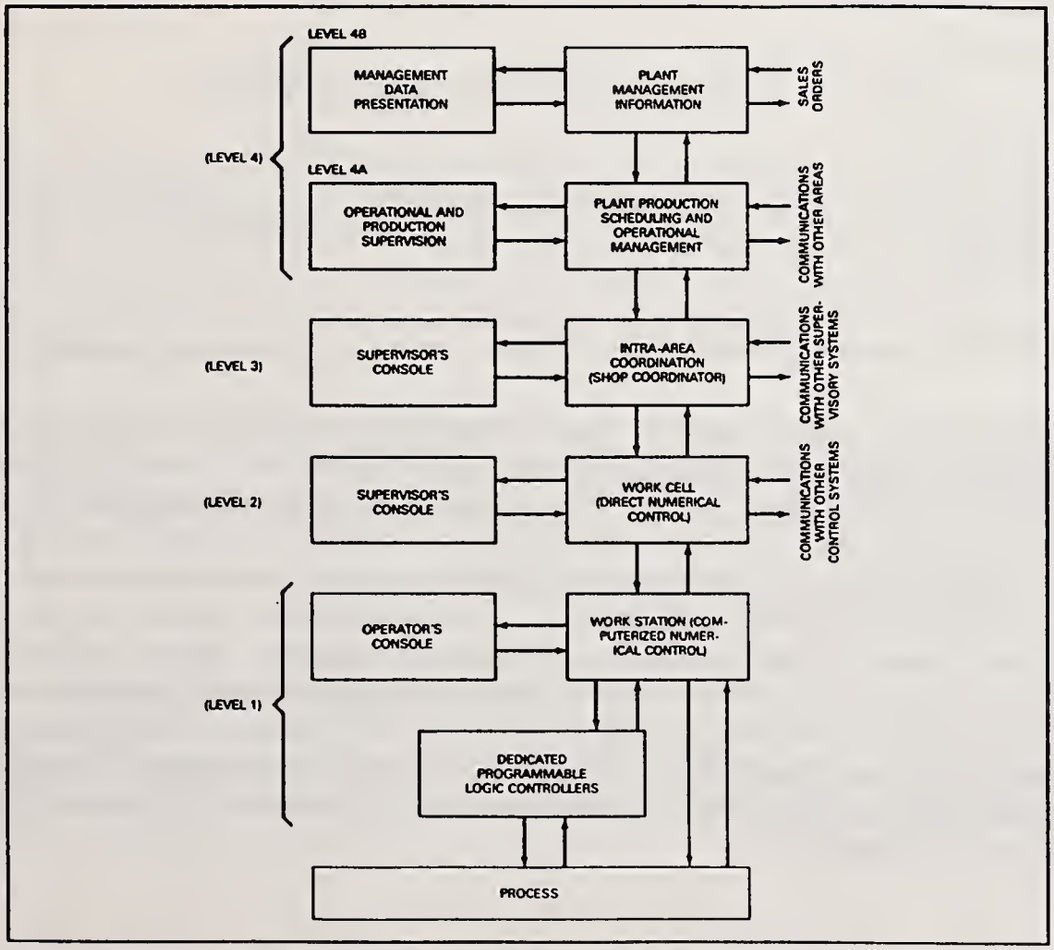


Figure 3 Assumed functional hierarchy computer system structure for a large manufacturing complex (Computer Integrated Manufacturing System (CIMS)).

Table I presents a listing of the major reference models for computer integrated manufacturing which are known to be in the open literature, including a set of references to each. Others have also been developed by individual companies for their own use but are therefore not readily available. Probably the best known of the available models is that developed by the United States National Bureau of Standards (now the National Institute of Standards and Technology) to describe the operation of their Automated Manufacturing Research Facility (AMRF) in 1980 [1,15-17]. The oldest of these models is that developed by the Case-Western Reserve University group in the late 1960s to attempt a theoretical base for multilevel control systems [11,18,21,22,27].

TABLE I

SOME AVAILABLE REFERENCE MODELS FOR CIM

1.	Digital Equipment Corporation	[10,30]
2.	ESPRIT, CIM-OSA/AMICE	[4,6]
3.	International Business Machines	[5,36]
4.	ISO/TC184/SC5/WG1	[8,12-14,31]
5.	Loughborough University, UK	[32]
6.	National Institute of Standards and Technology, AMRF, USA	[1,15-17]
7.	Phillips Industries, The Netherlands	[7]
8.	PROCOS-AS, Denmark	[23]
9.	Purdue University, USA	[26,28,33-35]
10.	Society of Manufacturing Engineers, USA	[29]
11.	Case-Western Reserve University, USA USA	[11,18,21,22,27]

Note: Items 2,4-6,8 and 9 are discussed in the (March/April - 1989) issue of the Int. J. of Computer Integrated Manufacturing.

The majority of the models seek to describe the operations of a complete industrial enterprise including management and external influences (Items 1-4, 7 and 10). The other five (Items 5, 6, 8, 9 and 11) confine themselves effectively to the factory itself.

Table II presents a set of generalities which can be used to describe each of the listed models. Within the limitations of their described coverage, it appears to be readily possible to convert each of the successful models into any one of the others. The differences between them are therefore encompassed in the presentation methods used, the scope of coverage involved, and the degree of detail of the functional descriptions listed. This is to be expected since each of them strives to be a generic description of the field of manufacturing and this interchangeability is further proof of the concept of generality of the field of CIM modelling itself.

TABLE II

CHARACTERISTICS OF CIM REFERENCE MODELS

1. ALL OF THE DESCRIBED MODELS USE A MULTILEVEL, FUNCTIONAL HIERARCHICAL STRUCTURE AS AT LEAST ONE OF THEIR METHODS OF VIEWING THE DESCRIBED SYSTEMS. THIS HAS THE FOLLOWING BENEFITS:

   A. THE USE OF LEVELS REDUCES THE SIZE AND COMPLEXITY OF EACH CHOSEN ELEMENT OF THE MODEL.

   B. LIKEWISE, LEVELS LIMIT THE SCOPE OF RESPONSIBILITY AND AUTHORITY OF EACH ELEMENT.

   C. LEVELS PERMIT A DIFFERENTIATION BETWEEN THE LENGTH OF PLANNING HORIZON AND THE REQUIRED SPEED OF RESPONSE OF EACH ELEMENT DEPENDING ON THE LEVEL IN WHICH IT IS DESCRIBED. THAT IS, THE PLANNING HORIZON DECREASES AND THE REQUIRED SPEED OF RESPONSE INCREASES AS ONE DESCENDS THE HIERARCHY.

2. EACH LEVEL DECOMPOSES COMMANDS FROM UPPER LEVELS INTO PROCEDURES TO BE EXECUTED AT THAT LEVEL OR INTO SUBCOMMANDS TO BE ISSUED TO ONE OR MORE SUBORDINATE LEVELS.
3. FINAL DECISION MAKING AND CONTROL RESIDES AT THE LOWEST POSSIBLE LEVEL -- WHERE THE MOST COMPLETE AND UP-TO-DATE INFORMATION RESIDES.

4. THE HIERARCHY VIEW ALONE IS NOT SUFFICIENT TO COMPLETELY DESCRIBE THE CIM SYSTEM. ALL SUCCESSFUL MODELS INCLUDE AT LEAST THREE SEPARATE VIEWS. OTHER VIEWS USED INCLUDE: DATA OR INFORMATION FLOW DIAGRAMS, DATA ENTITY RELATIONSHIP DIAGRAMS, IMPLEMENTATION PROCEDURE DIAGRAMS, COMMUNICATIONS ARCHITECTURAL DIAGRAMS, AND RESOURCE AND ORGANIZATIONAL DIAGRAMS. BECAUSE MANY OF THESE DIAGRAMS ARE REDUNDANT TO OTHERS, A COMPLETE DESCRIPTION CAN BE ACCOMPLISHED WITH SEVERAL DIFFERENT SELECTIONS OF VIEWS.

5. MOST ARCHITECTURES ALLOW ONLY VERTICAL CONTROL FLOW (EACH MODULE HAS ONLY ONE SUPERVISOR). THERE IS A SUBSTANTIAL THEORETICAL BASIS FOR THIS.

6. THERE IS NO THEORETICAL BASIS FOR PEER-TO-PEER COMMANDS. INFORMATION FLOWS IN THESE DIRECTIONS ARE ALLOWED AND USED IN MOST MODELS.

INTRODUCTION TO THE PURDUE MODEL

The Purdue CIM Reference Model, as it is commonly referred to, is designed to discuss the overall generic functional requirements of any manufacturing facility, regardless of industry, that are amenable to computerization within the foreseeable future and to define the viable relationships between these "automatable" functions and the other many functions of a manufacturing system for which no such possibility is attainable with currently foreseen technology.

One of the criteria for assessing whether or not a particular function is automatable, in the broadest sense of the work, is whether or not the operation of the function and its related physical equipment can be expressed in mathematical or computer program terms. Those functions which are not systematically expressible, particularly those which require human innovation for their implementation, are considered nonautomatable.

Therefore, there are two concepts or principles which are paramount to our work. These are automatability and innovation:

Automatability requires that the operation of the function and its related physical equipment be expressible in mathematical or computer program terms.

If this is not possible, then by definition a human being must supply the information or action which would otherwise be lacking. This is human innovation.

As is noted below, there are many forms in which the Reference Model for Computer Integrated Manufacturing could be expressed and many ways of describing the interrelationship of the functional requirements to be discussed. The committee has chosen to describe a definite control and information system structure and to treat the requirements so generated as firm. This is for emphasis only and to present one basic story. It is realized by all that there are many ways the model could be accomplished - this being only one of these.

Such a model must be a list of all of the truly generic tasks of the CIM system we are discussing here, and would arrange them in their proper relationship to each other (temporal, spatial and subordination). It would detail the generic units required to carry out these tasks, both the application entities (process units) and the service entities (computer system communications, data base, etc.). In addition, the model should also allow one to develop the best structure for the automatic control system (scheduling and dynamic control) for the plant, and to specify the best location or locations (within the structure) for carrying out each task.

The resulting model must have the following characteristics [8]:

1. Simply structured, flexible, modular, and generic.
2. Based upon readily understandable and acceptable terminology.
3. Able to be applied to a wide range of manufacturing operations and organizations.
4. Independent of any given, predetermined, realizations in terms of system configurations or implementations.
5. Open-ended in its ability to be extended and in its ability to encompass new technologies without unreasonably invalidating current realizations.
6. Independent of existing technologies in manufacturing automation and computer science.

What is being proposed for the CIM reference model is a blending of two types of methods for viewing the industrial automation picture; the hierarchical [33,34] and the data flow [9]. The hierarchical is the oldest and has had the most exposure and use over the years. This fits many of the existing plants such as chemical, steel and paper. The data flow type model helps define the interrelationship of all the functions required of the system which is not possible using the hierarchical model alone. In addition, a new Implementation Hierarchy view will help detail the actual implementation of the tasks involved. As a result this latter view is not necessarily generic.

This model then discusses the automation system and information handling requirements of the CIM system as diagrammed in the central box of Figure 1. While process equipment, machine tools and material handling equipments are considered parts of the CIM system in many circles, they are not so considered in this model because of their non-generic nature, (i.e., these are not included as a separate level in the model diagrams). Also excluded are the enterprise functions such as R & D, Engineering, Corporate Management, Sales, etc., listed here as external entities.

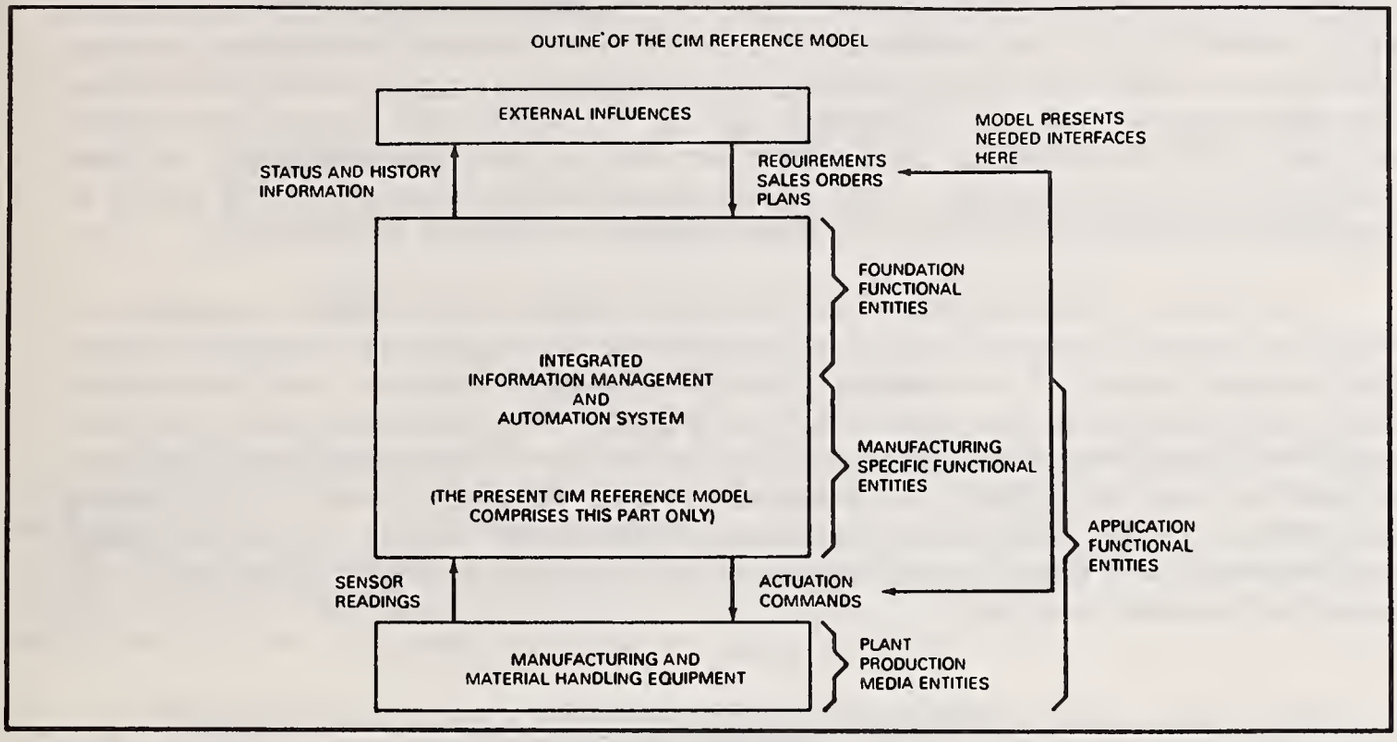


Figure 1 ***Relationship of the several classes of functional entities which comprise the CIM Reference Model and computer integrated manufacturing itself.***

THE GENERIC GOALS IN THE DESIGN AND OPERATION OF ANY PRODUCTION PLANT

The first step in the development of a statement of plant needs is a comprehensive list of long range plant goals (i.e., such as a five year plan). The goals to be stated here are truly generic for any production plant regardless of the industry involved. In view of this fact it is the thesis of the CIM Reference Model Committee that such a set of generic goals can best be satisfied by a system for the plant whose requirements are similarly generic in nature. Further, these requirements and the nature of the CIM system can be defined by a reference model which would thus be applicable to any industry or any plant in that industry.

The principal goal is to achieve a lower cost of operations or a higher process throughput for the plant through the application of process control and information systems technologies.

In the process industries, the term "CIM" is not used as often as the phrase "Plant-wide Control." The meaning is the same; the interconnection of information and control systems throughout a plant in order to fully integrate the coordination and control of the operation. Since process plants in the paper, steel, sugar and textile industries are known as "mills", these industries refer to "Mill-wide control." In this model, the term "Plant-wide Control" will be used generically to mean both plants and mills.

Improved human operator productivity will be realized through the implementation of individual work stations which provide the tools for decision-making. Timeliness of data will be assured through the interconnection of all work stations and information processing facilities with a high-speed, plant-wide local area network, and a global relational data base. The plant-wide control and information system will utilize and support the cultural resources of the organization as it adapts to changing business conditions. As new automation system technology is introduced, standard network interfaces will permit its integration with the plant-wide system (thus integrating islands of automation).

The broad goal is to improve the overall process and business operations by obtaining the benefits that will come from a completely integrated plant information system. The continual growth of the linkage of the process operations data with product line, project and business systems data will be supported. The system will make such data readily available, interactively in real-time, to any employee with a need to know, at work stations scattered throughout the plant and, above all, easy to use. The resulting comprehensive plant information management system will be the key to long-range improvements to: process control; product line management; plant management; and, support of business strategies.

THE CIM REFERENCE MODEL

A reference model is a previously agreed-upon or "standard" definitive document or conceptual representation of a system. The reference model defines requirements common to all implementations but is independent of the specific requirements of any particular implementation. The CIM Reference Model is thus a reference for computer integrated manufacturing. It is a detailed collection of the generic information management and automatic control tasks and their necessary functional requirements for the manufacturing plant.

The CIM Reference Model should be descriptive rather than prescriptive. Table III outlines the uses to which the Committee expects the CIM Reference model will be put.

TABLE III

USES OF THE CIM REFERENCE MODEL

1. ANY REFERENCE MODEL IS THE BASIC DESCRIPTIVE MEDIUM TO BE USED FOR FUTURE DISCUSSIONS OF THE SUBJECT AREA INVOLVED (HERE COMPUTER INTEGRATED MANUFACTURING (CIM)).
2. IT SHOULD ALLOW ANY MANUFACTURING SYSTEM AND ITS ASSOCIATED INFORMATION MANAGEMENT AND AUTOMATION ARCHITECTURE TO BE EVALUATED FOR COMPLETENESS, CAPABILITY, AND EXTENSIBILITY.
3. IT CAN ALSO SERVE AS A DESIGN GUIDE FOR THE DEVELOPMENT OF A NEW INFORMATION MANAGEMENT AND AUTOMATION ARCHITECTURE FOR A NEW ("GRASS ROOTS") OR RETROFITTED PLANT.
4. IT SERVES TO HIGHLIGHT THOSE FUNCTIONS WHICH ARE AMENABLE TO THEIR ESTABLISHMENT AS STANDARDS.
5. THE MODEL SHOULD HELP PROVIDE A MIGRATION PATH FROM THE CURRENT PLANT SYSTEM TO A NEW SYSTEM; BY MAKING EVIDENT THE CRITICAL FUNCTIONS FOR EARLY IMPLEMENTATION; AND BY PROVIDING A FRAMEWORK FOR THE REQUIREMENTS DEFINITION PHASE OF THE PROJECT.

6. SOME OTHER IMPORTANT USES ARE:

   A. EDUCATION - TO GET THE ORGANIZATION DIRECTED TOWARD A COMMON STRATEGY AND APPRECIATION OF THE STEPS REQUIRED TO ACHIEVE INTEGRATION.

   B. GUIDE - TO MEASURE PROGRESS TOWARD THE FINAL GOAL.

   C. MODULARIZATION OF THE STRATEGY - TO DIVIDE THE ATTACK ON THE PROBLEM INTO READILY SOLVABLE PIECES.

   D. ORGANIZATIONAL SUPPORT - TO DEVELOP A COMMITTED TEAM APPROACH TO THE PROBLEM.

THE MANUFACTURING PLANT IN TERMS OF THE CIM REFERENCE MODEL

The manufacturing plant is a collection of application functional entities which carry out the primary mission of the factory in producing marketable product and the associated information streams. The plant production media are supported by an integrated information and automation system composed of foundation and manufacturing specific functional entities which support the means of production. The plant interfaces the external world through a set of external influences or external entities.

The manufacturing mission and the established manufacturing policy of the company are articulated through the set of tasks and functional specifications assigned to each of the functional entities of the plant.

The CIM Reference Model is a generic description of the collection of functional entities which make up a factory and of their interaction through their assigned tasks and functional specifications.

Table IV defines the objective of the development of the CIM Reference Model by defining the idealized plant which is to be modelled. Figure 1 defines the interrelationships of the terms noted above which are necessary in defining the CIM Reference Model. Note that the external influences and the manufacturing equipment of the plant interact with the present model through appropriate interfaces to transmit all necessary information and commands.

TABLE IV

BASIS FOR THE FORMULATION OF THE

CIM REFERENCE MODEL

THE CIM REFERENCE MODEL AND ITS RELATED SET OF GENERIC FUNCTIONAL REQUIREMENTS WILL TAKE AS THEIR IDEALIZATION:

1. THE FULLY AUTOMATED PLANT (I.E., STAFFED BY AGENTS (HUMAN OR MACHINE) WHOSE DECISIONS ARE EFFECTIVELY COMPUTABLE).

2. THE TOTALLY RESPONSIVE (I.E., CONTROLLABLE) MANUFACTURING SYSTEM CARRYING OUT THE ESTABLISHED MANUFACTURING POLICY OF THE COMPANY.

3. AN ALLOWANCE FOR HUMAN IMPLEMENTED PROCESSES IN THE PRODUCTION SYSTEM BY ASSURING THE NECESSARY FUNCTIONAL COMMUNICATIONS FOR THOSE PROCESSES OF THE FACTORY.

4. A SYSTEM THAT WILL BE FLEXIBLE ENOUGH TO ALLOW FORESEEABLE CHANGES IN THE ESTABLISHED MANUFACTURING POLICY.

THE GENERIC DUTIES OF A CIM SYSTEM AND THEIR EXPRESSION VIA THE HIERARCHICAL FORM OF THE REFERENCE MODEL (SCHEDULING AND CONTROL HIERARCHY VIEW) OF THE SYSTEM

THE GENERIC TASKS OF A PLANT-WIDE COMPUTER CONTROL SYSTEM

Overall automatic control of any large modern industrial plant regardless of the industry concerned involves each of the requirements listed in Table V.

Because of the ever-widening scope of authority of each of the first three requirements of the Table V in turn, they effectively become the distinct and separate levels of a superimposed control structure, one on top of the other. Also in view of the amount of information which must be passed among the above four "tasks" of control, a distributed computational capability organized in a hierarchical fashion would seem to be the logical structure for the required control system. This must be true of any plant regardless of the industry involved.

TABLE V

AN OVERALL PLANT AUTOMATION SYSTEM MUST PROVIDE

1. AN EFFECTIVE DYNAMIC CONTROL OF EACH OPERATING UNIT OF THE PLANT TO ASSURE THAT IT IS OPERATING AT ITS MAXIMUM EFFICIENCY OF PRODUCTION CAPABILITY, PRODUCT QUALITY AND/OR OF ENERGY AND MATERIALS UTILIZATION BASED UPON THE PRODUCTION LEVEL SET BY THE SCHEDULING AND SUPERVISORY FUNCTIONS LISTED BELOW. THIS THUS BECOMES THE CONTROL ENFORCEMENT COMPONENT OF THE SYSTEM. THIS CONTROL REACTS DIRECTLY TO COMPENSATE FOR ANY EMERGENCIES WHICH MAY OCCUR IN ITS OWN UNIT.

2. A SUPERVISORY AND COORDINATING SYSTEM WHICH DETERMINES AND SETS THE LOCAL PRODUCTION LEVEL OF ALL UNITS WORKING TOGETHER BETWEEN INVENTORY LOCATIONS IN ORDER TO CONTINUALLY IMPROVE (I.E., OPTIMIZE) THEIR OPERATION. THIS SYSTEM ASSURES THAT NO UNITS ARE EXCEEDING THE GENERAL AREA LEVEL OF PRODUCTION AND THUS USING EXCESS RAW MATERIALS OR ENERGY. THIS SYSTEM ALSO RESPONDS TO THE EXISTENCE OF EMERGENCIES OR UPSETS IN ANY OF THE UNITS UNDER ITS CONTROL IN COOPERATION WITH THOSE UNITS' DYNAMIC CONTROL SYSTEMS TO SHUT DOWN OR SYSTEMATICALLY REDUCE THE OUTPUT IN THESE AND RELATED UNITS AS NECESSARY TO COMPENSATE FOR THE EMERGENCY. IN ADDITION, THIS SYSTEM IS RESPONSIBLE FOR THE EFFICIENT REDUCTION OF PLANT OPERATIONAL DATA FROM THE DYNAMIC CONTROL UNITS, DESCRIBED JUST ABOVE, TO ASSURE ITS AVAILABILITY FOR USE BY ANY PLANT ENTITY REQUIRING ACCESS TO IT AS WELL AS ITS USE FOR THE HISTORICAL DATA BASE OF THE PLANT.

3. AN OVERALL PRODUCTION CONTROL SYSTEM CAPABLE OF CARRYING OUT THE SCHEDULING FUNCTIONS FOR THE PLANT FROM CUSTOMER ORDERS OR MANAGEMENT DECISIONS SO AS TO PRODUCE THE REQUIRED PRODUCTS FOR THESE ORDERS AT THE BEST (NEAR OPTIMUM) COMBINATION OF CUSTOMER SERVICE AND OF THE USE OF TIME, ENERGY, INVENTORY, MANPOWER AND RAW MATERIALS SUITABLY EXPRESSED AS COST FUNCTIONS.

4. A METHOD OF ASSURING THE OVER ALL RELIABILITY AND AVAILABILITY OF THE TOTAL CONTROL SYSTEM THROUGH FAULT DETECTION, FAULT TOLERANCE, REDUNDANCY, UNINTERRUPTIBLE POWER SUPPLIES, MAINTENANCE PLANNING, AND OTHER APPLICABLE TECHNIQUES BUILT INTO THE SYSTEM'S SPECIFICATION AND OPERATION.

A hierarchical arrangement of the elements of a distributed, computer-based, control system seems ideal arrangement for carrying out the automation of the industrial plant just described. Figures 2 and 3 lay out one possible form of this distributed, hierarchical computer control system for overall plant automation. Note that Figure 2 uses the nomenclature common to the continuous process industries while Figure 3 presents the computer integrated manufacturing system or CIMS commonly used in the discrete manufacturing industries to represent the hierarchy. Note also that the levels represented here are "functional" levels. Whether or not they represent actual physical hardware levels

depends on the size and complexity of the manufacturing plant. Nevertheless it is our thesis that the two diagrams of Figures 2 and 3 are exactly functionally equivalent.

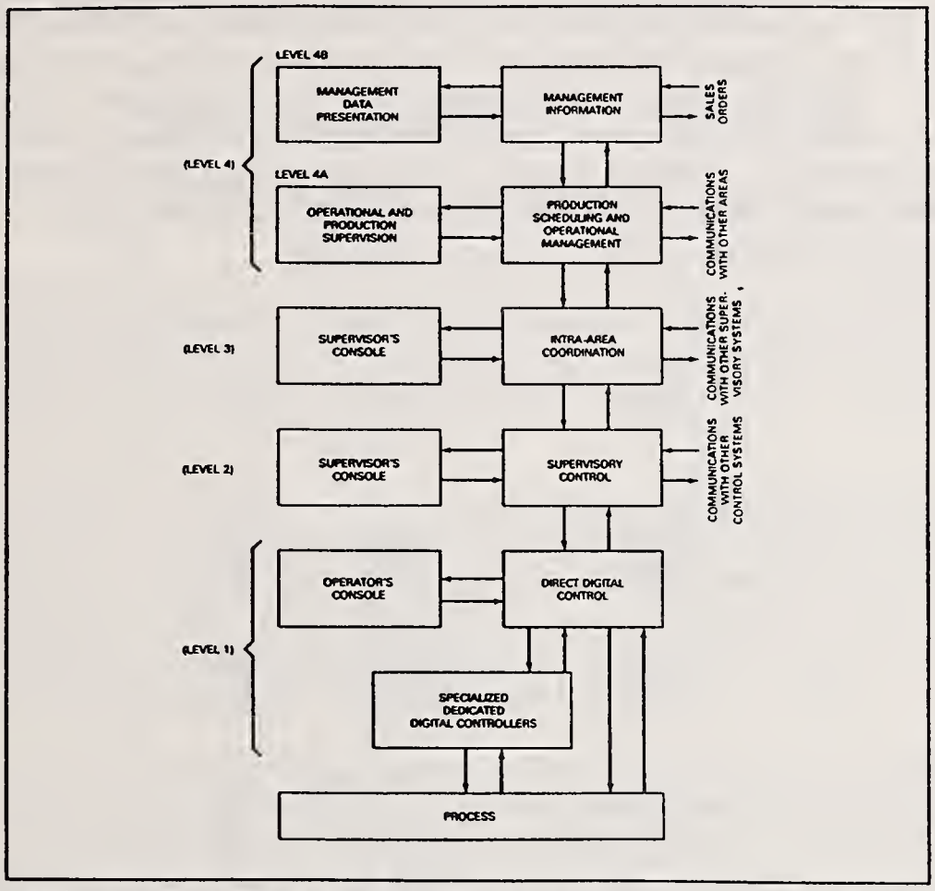


Figure 2 Assumed functional hierarchical computer control structure for an industrial plant (continuous process such as petrochemicals).

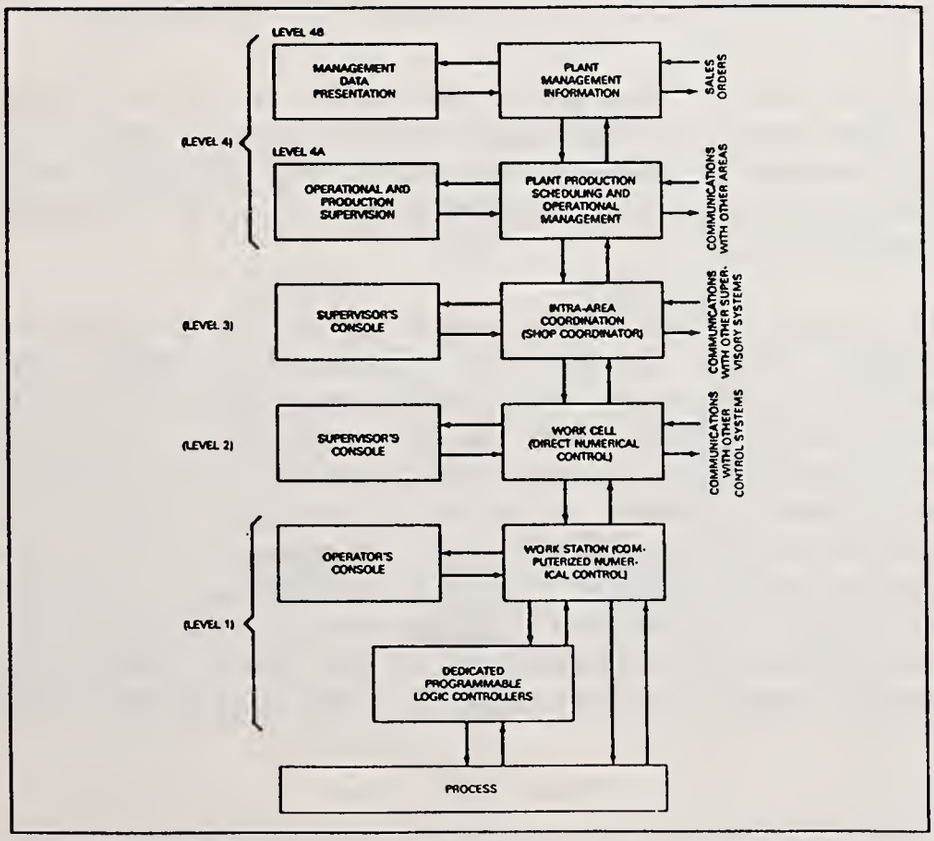


Figure 3 Assumed functional hierarchy computer system structure for a large manufacturing complex (Computer Integrated Manufacturing System (CIMS)).

Figures 2 and 3 represent the simplest situation - that of a company with only one manufacturing plant. The corresponding situation with a multiplant company is represented in Figure 4 in that an additional level is necessary to separate the company's distribution or assignment of orders to the various plants from the plant's own production scheduling activities. In addition, the company management functions of Level 4B are now transferred to a new Level 5B. With this simple discussion of potential expansion of the model, continuing discussion of the model in this paper will concentrate, for ease of consideration, on the single plant company, i.e., Figures 2 and 3. The tasks carried out at Level 5B would be the same as those assigned here at Level 4B with suitable allowance for the wider horizon of interest of the management of the larger company.

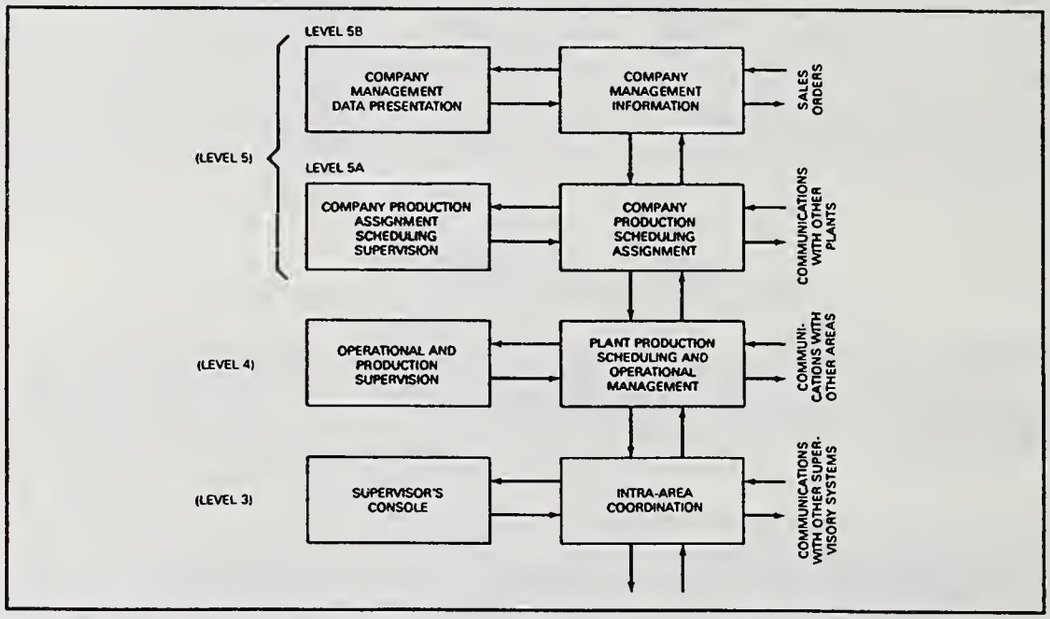


Figure 4 Assumed functional hierarchical computer control structure for an industrial company (multi-plant).

The detailed tasks that would be carried out at each level of the hierarchy can be readily described. These tasks are easily subdivided into those related to production scheduling, control enforcement, systems coordination and reporting, and reliability assurance. These can then be extensively expanded to include any degree of detail desired as in the model itself [26,28,33-35].

Such lists can outline the tasks which must be carried out in any industrial plant, particularly at the upper levels of the hierarchy. Details of how these operations are actually carried out may vary drastically, particularly at the lowest levels, because of the nature of the actual process being controlled. We all recognize that a distillation column will never look like or respond like an automobile production line. But the operations themselves remain the same in concept, particularly at the upper levels of the hierarchy. Thus it is our contention that, despite the different nomenclature of Figures 2 and 3, the major differences in the control systems involved in concentrated in the details of the dynamic control technologies used at Level 1 and the details of the mathematical models used for optimization at Level 2. The differences are thus concentrated in the details of the control and operation of the individual production units (the application entities) of the factory.

Commonality is in the support functional entities (computational services, communications, data base technology, management structure, etc.). Sensing and communication techniques are similar in both systems. Similar optimization algorithms can be used, computer systems technology and programming techniques and production scheduling technology should be similar, to name only a few.

THE DATA-FLOW GRAPH, A FUNCTIONAL NETWORK VIEW OF THE CIM REFERENCE MODEL

There is need in the CIM Reference Model to have a mechanism to show the interconnection and precedence of the several tasks assigned to the overall mill-wide control system. An excellent method for showing this is the so-called Data-Flow Graph or Information-Flow Graph using a technique known as Structured Analyses [9] also known as the Yourdon-DeMarco technique [3].

This method shows diagrammatically the interconnection of the several tasks carried out by the control system and allows the potential for an ever greater detailing of these tasks in the form of subtasks and the resulting interconnections of these subtasks with each other and the main tasks. These diagrams are restricted to the model as defined earlier (i.e., the definable scheduling and control system for the manufacturing facility and including only interfaces to the external influences).

The set of diagrams begins with the interconnection of the influencing external entities on the factory itself (Figure 5). In the present model one very important external influence of the factory is the company management itself. As noted in Figures 6-7, management interfaces through the staff departments who provide services to the factory itself or express managements' policies in sets of requirements to be fulfilled by the factory.

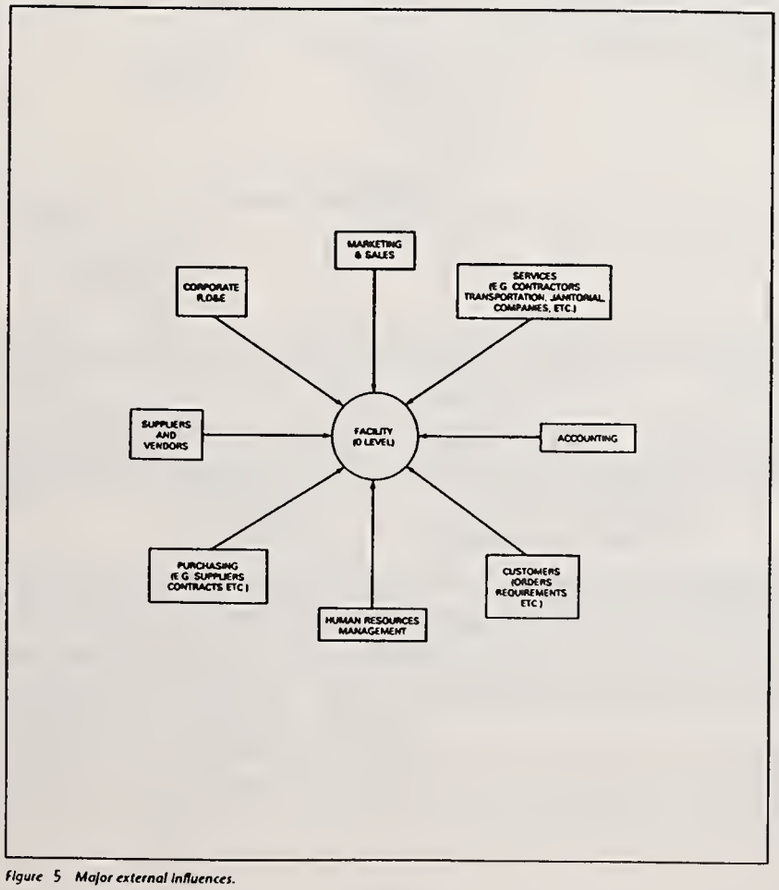


Figure 5 Major external influences.

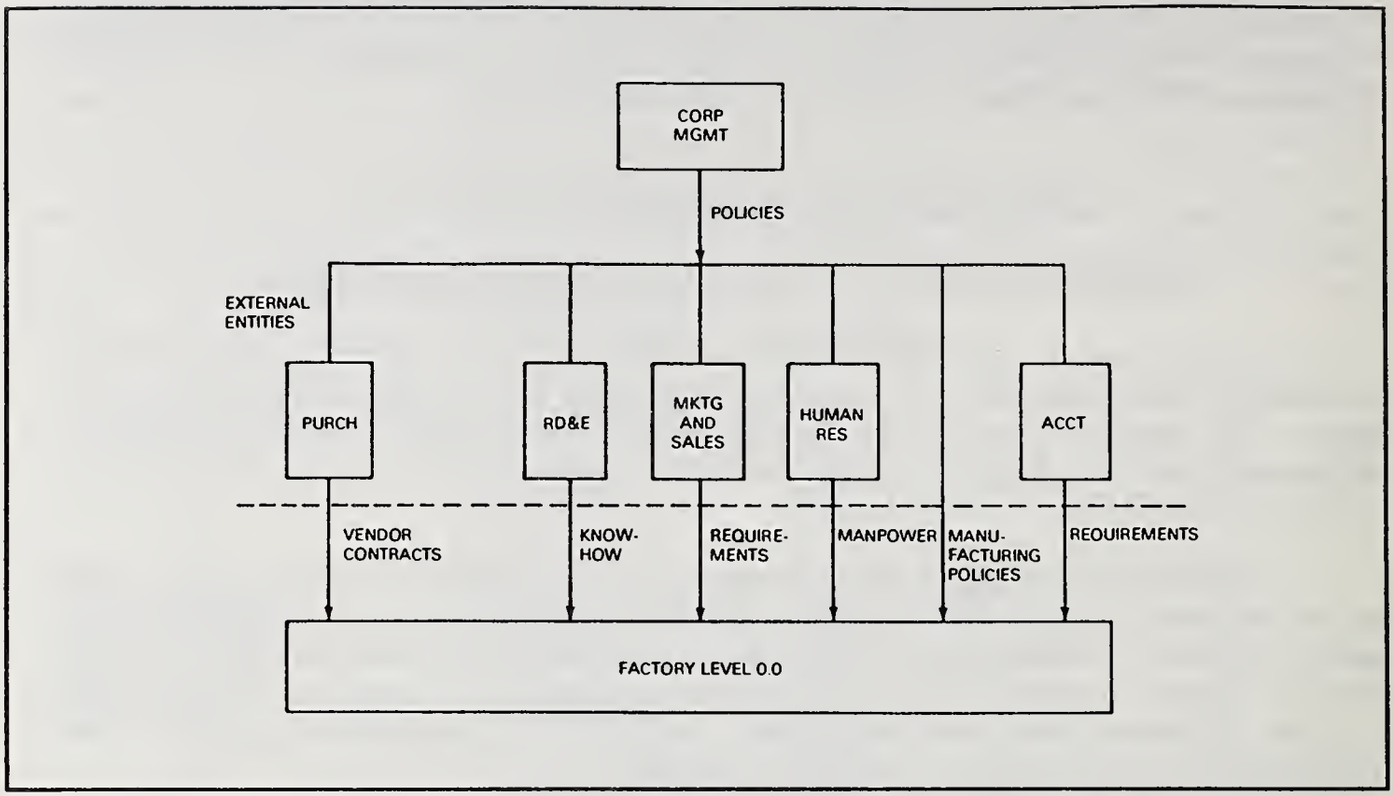
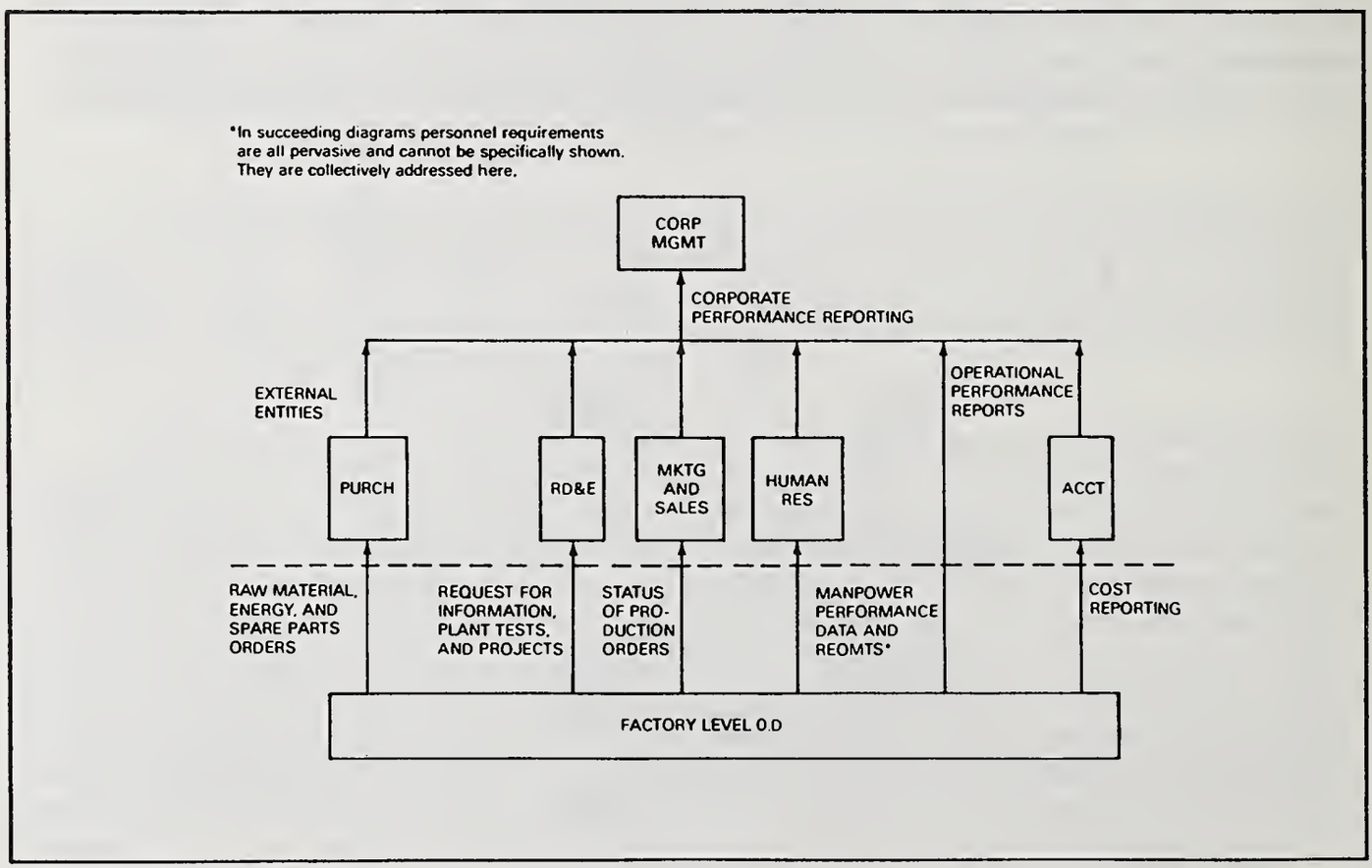


Figure 6 Requirements interfacing of corporate management and staff functional entities to the factory.



Fiigure 7 Report interfacing to corporate management and staff functional entities from the factory.

Figure 8 shows the initial number in the expansion of information in the description of the plant functions as ever greater detail is desired. Such an analysis can be carried out for any function or set of functions to the detail necessary or desired. Only an example is shown here.

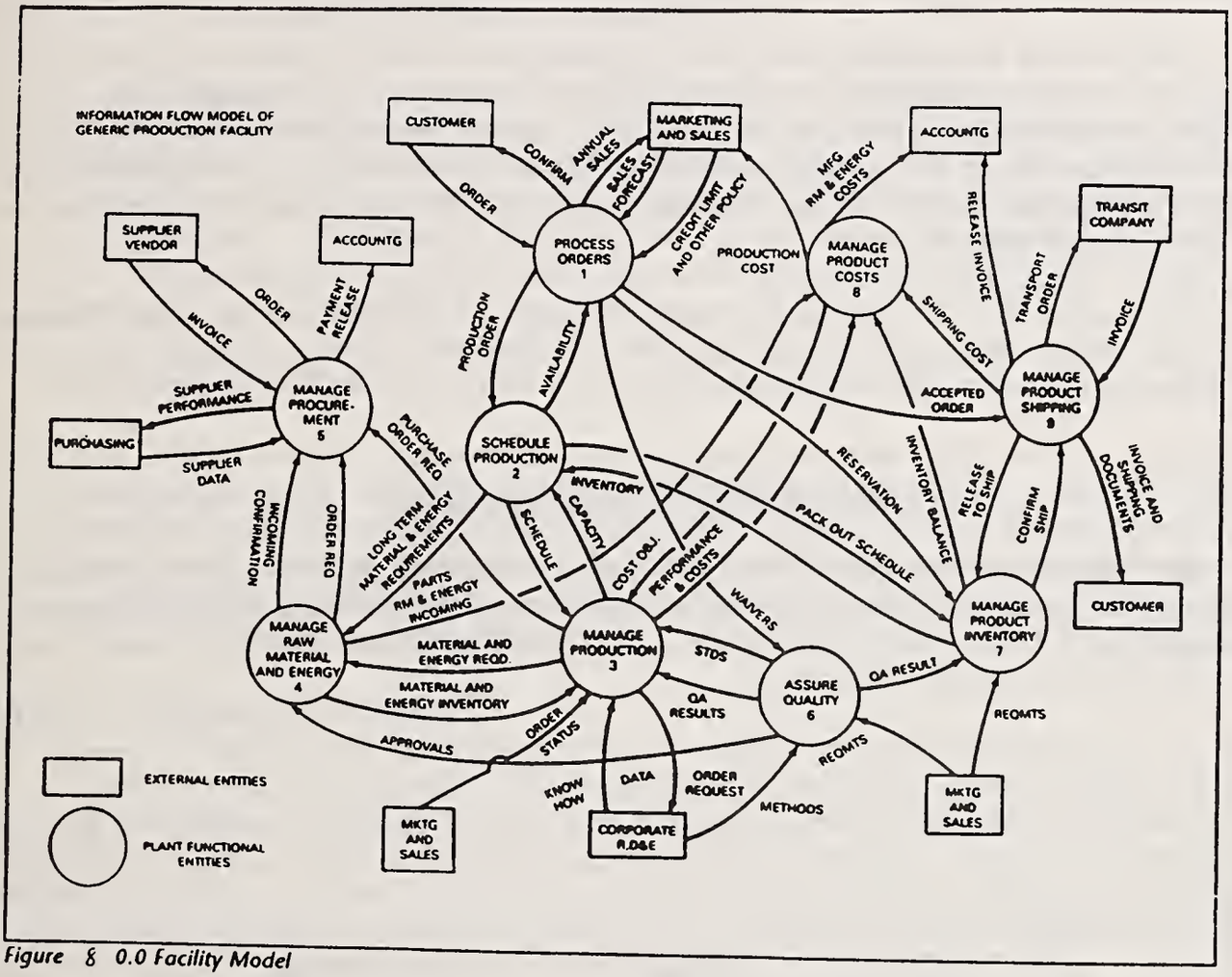


Figure 8 0.0 Facility Model

SOME INADEQUACIES OF THE DATA-FLOW-GRAPH-MODEL

Foundation functional entities cannot be shown on the data flow diagram. That is, the data flow diagram mainly shows the interconnection of manufacturing-specific functional entities.

The data-flow graph will accommodate all functional entities which exhibit the principle of locality. Those which are diffuse cannot be accommodated because of the number of lines involved. The principle of locality may be a virtual consolidation of operations for the sake of the diagram. Most foundation functional entities are diffuse, e.g., data base, communications, management, etc.

THE IMPLEMENTATION HIERARCHY VIEW OF THE CIM SYSTEM

Figure 9 represents one concept of the components of elements required for the implementation of the CIM system for each "bubble" or Manufacturing Specific Functional Entity of the data-flow diagram or for each Task of the Scheduling and Control Hierarchy view.

This concept allows each task to be expressed in as many layers as required (11 maximum). Layers may be used or nulled as necessary. Thus such a model needs to be developed in specific form for each "bubble" of the data-flow diagrams presented in the previous chapter using the generic model of Figure 9 as a base and supplying the appropriate implementation details. Likewise this would be done for each separate task of the overall control system.

Just as the ISO-OSI model [2] breaks the tasks of the communications between systems into layers, this model breaks the tasks of plant control into functional layers. A brief discussion of each layer of the model follows.

DESCRIPTION OF THE LAYERS

As diagrammed in Figure 9 the CIM model is represented in the implementation hierarchy view by an eleven layer structure. The hardware elements of the system are represented by the lower five layers of the system (1-5), while the software elements are represented by the top six layers (6-11).

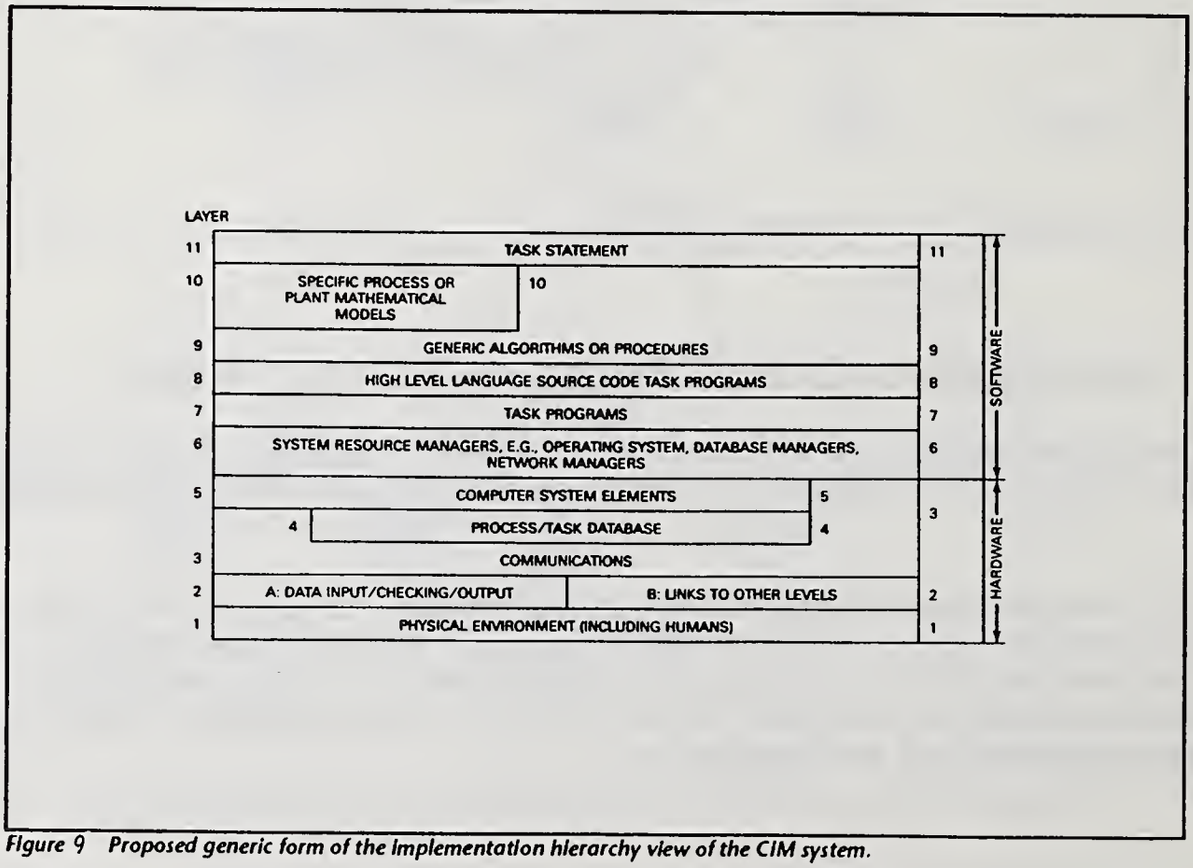


Figure 9 Proposed generic form of the implementation hierarchy view of the CIM system.

LAYER 1 - PHYSICAL ENVIRONMENT (INCLUDING HUMANS)

This layer would typically represent the process equipment (i.e., reactors and distillation columns), machine tools (i.e., CNC and human operators) and supporting areas such as utilities and packaging. As noted often earlier, these elements would be non generic in any specific case and are included here for completeness.

LAYER 2 - A - DATA INPUT/CHECKING/OUTPUT

The detection and measurement of the status and of the actions occurring in Layer 1 are contained in this layer. This layer represents the eyes and ears of the CIM System. This includes the determination of the values of such variables as temperature, level, pressure, chemical analysis, position, weight, etc., from sensors and detectors. Also included are inputs from touch screens, mice, bar code readers, etc. Typical system actuators would include valves and valve positioners, hydraulic drives, solenoids, relays, CRTs, printers, etc.

B- LINKS TO OTHER LEVELS

Where higher level functions are involved such as overall production scheduling in large industrial production plants, then Layer 2 represents the link to other computer and control equipment involved in implementing the task described. In the case of Overall Production Scheduling (which itself takes place in Levels 4A and 3 of the hierarchical system of Figure 2 or 3) this would include all elements of Levels 1 and 2 of the latter diagram.

LAYER 3 - COMMUNICATIONS

Layer 3 moves the data within the system. The clients of this communication system would be the various computer systems and databases which manipulate and store this information and the various functional entities which use the resulting information. Included in this layer are device gateways and drivers as/if required. The communications structure should, as far as possible, follow the OSI model and agreed upon industry-wide standards for their implementation.

LAYER 4 - PROCESS/TASK DATABASE

The global database of the factory or plant resides in this layer. It becomes the collective memory for the CIM system. This database may be distributed as determined by the implementation plan.

LAYER 5 - COMPUTER SYSTEM ELEMENTS

Exact content at this layer will be determined by the functional requirements of the system, but must encompass the entirety of the intelligent computing devices contained in the CIM System which are required for the task at hand. Examples include computers, disks, and database machines.

LAYER 6 - SYSTEM RESOURCE MANAGERS

This layer contains the software which allocates and manages the elements which comprise the system. Examples are operating systems, database management systems, network managers, system utilities, and data dictionaries.

LAYER 7 - COMPILED OR INTERPRETED CODE

This layer represents the program as actually executed by the computer system. It may be stored in random access or read only memory as required by the application at hand.

LAYER 8 - HIGH LEVEL LANGUAGE SOURCE CODE

This layer contains the source code in the form of a high level language such as Ada, FORTRAN, C or 4th generation languages.

LAYER 9 - GENERIC ALGORITHMS/PROCEDURES

Each process has calculation, algorithmic, or modeling requirements. These would reside in this layer and service the lower layers. Examples would be linear programs as used for a catcracker optimization routine or a dynamic optimization technique as used for robot optimal path determination.

LAYER 10 - SPECIFIC PROCESS OR PLANT MATHEMATICAL MODELS

For certain applications, models will be included. These models allow simulation of the process to generate information not otherwise available. Uses would be supplying unmeasurable data, verify existing data, or predicting future data. Examples are process unit models for advanced control systems or business models for scheduling functions.

LAYER 11 - TASK STATEMENT

Description or specification of the task or function to be accomplished (the application functions of the manufacturing plant) would reside at this layer.

DEVELOPMENT OF THE MODEL AND ACKNOWLEDGMENTS

The CIM Reference Model described in this paper was developed as a, Reference Model for Computer Integrated Manufacturing (CIM), A Description From the Viewpoint of Industrial Automation, by the CIM Reference Model Committee of the International Purdue Workshop on Industrial Computer systems during the period April 1986 - May 1988. It was published in book form by the Instrument Society of America in October 1989. This text expands greatly on the details of the model sketched in this paper. A listing of the active members of the CIM Reference Model Committee is presented in the text [35].

The International Purdue Workshop is an activity of the Purdue Laboratory for Applied Industrial Control of Purdue University. The Workshop has been supported by the international industrial control community for the past twenty years.

Professor Theodore J. Williams, Director, Purdue Laboratory for Applied Industrial Control served as Chairman of the CIM Reference Model Committee and Editor of the text describing its work [35].

REFERENCES

1. Albus, J., Barbera, A., and Nagel, N., "Theory and Practice of Hierarchical Control," Proceedings of the 23rd IEEE Computer Society International Conference (1981).

2. Anonymous, Data Processing - Open Systems Interconnection-Basic Reference Model, ISO DIS 7498, International Standards Organization, Geneva, Switzerland (December 3, 1980).

3. Anonymous, Information Flow Model of Generic Production Facility, the Foxboro Company, Foxboro, Massachusetts (1986).

4. Anonymous, CIM-OSA Reference Architecture Specification, ESPRIT Project Number 688, CIM-OSA/AMICE, Brussels, Belgium (May 1989).

5. Anonymous, Computer Integrated Manufacturing, The CIM Enterprise, Document G320-9802-00, International Business Machines Corporation, White Plains, New York (October 1989).

6. Beeckman, Dirk, "CIM-OSA: Computer Integrated Manufacturing-Open Systems Architecture, "Int. J. Computer Integrated Manufacturing, Vol. 2, No. 2, pp. 94-105 (March-April 1989).

7. Biemans, Frank P. M., A Reference Model for Manufacturing Planning and Control, doctoral Dissertation, University of Twente, Enschede, The Netherlands (October 27, 1989).

8. CIM Reference Model Working Group, ISO/TC184/SC5/WG1, Technical Report on CIM Reference Model (July 1987).

9. DeMarco, Tom, Structured Analysis and System Specification, McGraw-Hill Book Company, New York, NY (1985).

10. Flateau, Ulrich, Digital's CIM Architecture, Revision 1.1, Digital Equipment Corporation, Marlboro, Massachusetts (April 1986).

11. Findeisen, W., and Lefkowitz, I., "Design and Applications of Multilayer Control," Proc. Fourth IFAC Congress, Warsaw, Poland (1969).

12. Graefe, Udo, and Thomson, Vince, "A Reference Model for Production control," Int. J. Computer Integrated Manufacturing, Vol. 2, No. 2, pp. 86-93 (March-April 1989).

13. ISO (International Organization for Standardization), "The Ottawa Report on Reference Models for Manufacturing Standards," ISO TC184/SC5/WG1 N51 (September 14, 1986).

14. ISO (International Organization for Standardization), "Reference Model for Shop floor Standards, Part 1," ISO TC184/SC5 N131 (July 15, 1988).

15. Jones, Albert, Barkmeyer, Edward and Davis, Wayne, "Issues in the Design and Implementation of a System Architecture for Computer Integrated Manufacturing," Int. J. Computer Integrated Manufacturing, Vol. 2., No. 2, pp. 65-76 (March-April 1989).

16. Jones, A. and Mclean, C., "A Proposed Hierarchical Control Model for Automated Manufacturing systems," Journal of Manufacturing Systems, Vol. 5, pp. 15-25 (1986).

17. Jones, a., and Whitt, N., Eds., Proceedings on Factory Standards Model Conference, National Bureau of Standards, Gaithersburg, Maryland (1985).

18. Lefkowitz, I., and Schoeffler, J. D., "Multilevel Control Structures for Three Discrete Manufacturing Processes," Proc. Fifth IFAC Congress, Paris, France (1972).

19. McCarthy, J. J., "MAP and the Integration of Plant Data Bases," in Standards in Information Technology and Industrial Control, N.E. Malagardis and T. J. Williams, Editors, North Holland Publishing Company, Amsterdam, The Netherlands (1988).

20. McCarthy, J. J., and Ruckman, R. P., "The Application of the CIM Reference Model to a Continuous Process Plant," Advanced Control in Computer Integrated Manufacturing, Morris, H. M., Kompass, E. J. and Williams, T. J., Editors, Purdue University, West Lafayette, Indiana (September 28-30, 1987), pp. 87-96.

21. Mesarovic, M. D., "Multilevel Systems and Concepts in Process Control," Proc. of IEEE,58, 111-125 (1970).

22. Mesarovic, M. D., Macko D., and Takahara, Y., Theory of Hierarchical Multilevel Systems, Academic Press, New York (1970).

23. Moss, s. P., "A Management and Control Architecture for Factory-Floor Systems: From Concept to Reality," Int. J. Computer Integrated Manufacturing, Vol. 2, No. 2, pp. 106-113 (March-April 1989).

24. Parunak, H. V., and White, J. F., Private Communications to CIM Reference Model Committee, (March 2-3, 1987).

25. Prage, J. H., Personal Communication to the CIM Reference Model Committee, International Purdue Workshop on Industrial Computer Systems, Inland Steel Company, E. Chicago, Indiana (March 8, 1988).

26. Project Staff, Tasks and Functional Specifications of the Steel Plant Hierarchy Control System, Report Number 98, Purdue Laboratory for applied Industrial Control, Purdue University, West Lafayette, Indiana (September 1977; Revised June 1984).

27. Schoeffler, J. D., "Multilevel Systems and Decomposition for the Solution of Static Optimization Problems; Decomposition and Multilevel Methods for On-Line Computer Control of Industrial Processes," In Wismer, D. A., ed., Optimization Methods for Large-Scale Systems, McGraw Hill, New York (1970).

28. Systems Engineering Group, INCOS Project, Tasks and Functional Specifications of the Bhilai Steel Plant Integrated Control System (INCOS), Steel Authority of India, Ltd., Delhi, India (April 1986).

29. Thacker, R. M., King, R. E., Robert, Edward, and Ploskonka, C. A., A New CIM Model: A Blueprint for the Computer-Integrated Manufacturing Enterprise, Society of Manufacturing Engineers, Detroit, Michigan (1989).

30. Temple, Barker and Sloan, Inc., "Customer Perspectives on CIM," Proceedings, CIM Management Forum, Digital Equipment Corporation, Orlando, Florida (January 25-27, 1988).

31. Thomson, V., and Graefe, U., "CIM - A manufacturing Paradigm, SME MM86-722," 5th Canadian CAD/CAM and Robotics Conference, Toronto, 1986.

32. Weston, R. H., Hodgson, A., Gascoigne, J. D., Sumpter, C. M., Rui, A., and Coutts, I., "Configuration Methods and Tools for Manufacturing Systems Integration," Int. J. Computer Integrated Manufacturing, Vol. 2., No. 2, pp 77-85 (March-April 1989).

33. Williams, T. J., "Computer Control in the Steel Industry," Computers in Mechanical Engineering, Vol. 2, No. 4, pp 14-16 (January 1989).

34. Williams, T. J., Editor, Analysis and Design of Hierarchical control Systems, Elsevier Science Publisher, Amsterdam, The Netherlands (1985).

35. Williams, T. J., Editor, A Reference Model for Computer Integrated Manufacturing, A Description From the Viewpoint of Industrial Automation, CIM Reference Model Committee, International Purdue Workshop on Industrial Computer Systems, Purdue Laboratory for Applied Industrial Control, Purdue University, West Lafayette, Indiana (May 1988); Instrument Society of America, Research Triangle Park, North Carolina (1989).

36. Zachman, John, Application and Data: Factory of the Future, Applications Technology Group, Information Systems Department, International Business Machines Corporation, Los Angeles, California (May 23, 1989).

AN APPROACH TO IMPLEMENTING CIM IN SMALL & MEDIUM SIZE COMPANIES

ROBERT E. YOUNG
JOHAN VESTERAGER

Abstract

In this paper we describe results from the Danish CIM/GEMS project that demonstrate it is possible for manufacturing people with little or no prior computer experience to design and build their own CIM systems. The approach is applicable to small and medium size companies and is a way to overcome the software backlog that currently exists so that the projected growth in the CIM area can occur. In addition, we developed a novel approach to knowledge and technology transfer that allowed industry to receive technology and begin using it as soon as it is developed.

1. Introduction

This paper describes an approach to implementing CIM in small and medium size companies in which manufacturing people with little or no prior computer experience can design and build their own CIM systems. It is the result of the Danish CIM/GEMS (General Methods for Specific Solutions in Computer-Integrated Manufacturing) project, a multiyear project that developed tools and methods for system development, and developed knowledge and technology transfer mechanisms to provide industry with the information. Although small and medium size companies may not seem important, they have a significant impact on the economies of Denmark and America, and possibly most western industrialized countries. In Denmark, 98% of the discrete-parts manufacturing companies employ 200 people or less. These small and medium size companies employ 52% of the people working in discrete-parts manufacturing and produce 33% of the country's value-added goods. In America, 95% of discrete-parts companies employ 250 people or less, which is 43% of the people working in discrete-parts manufacturing, and produce 18% of the country's value-added goods. From discussions with colleagues in other western European countries we believe most industrial countries have a similar distribution of small and medium size companies. Consequently, the results of our project have implications not only for Denmark and America, but possibly for other western industrialized countries.

Although most effort to date in CIM system development has focused upon large corporations, the general feeling is that in the next decade small and medium size companies will be the fastest growing segment of the CIM market [BW86]. However, this market growth can't occur because the resources necessary to build the systems are not available. There currently is a severe shortage of software professionals. There is also an estimated development backlog for software-based systems of four years and a hidden backlog of eight years; and complicating development for the 1990s is a projected increasing demand for software professionals of ten times the current level [SHEM87]. Consequently, if CIM systems can only be built by software professionals then there is today, and will continue to be, insufficient numbers of computer scientists to build the large number of systems needed, and the market growth will not occur. Since CIM is generally viewed

as a principle mechanism for productivity improvement [BW86], this development bottleneck has serious implications for manufacturing.

Our primary project goal was to test an alternative approach to CIM system development -- have manufacturing personnel with little prior computer experience build their own CIM systems. Our project developed tools and methods that supports this approach and through projects with industry, demonstrated its validity.

The next section provides a background of the project followed by a description of the tools and methods, and development approach we used in the project. The last section describes our knowledge and technology transfer approach.

2. Background of the CIM/GEMS project

In 1987, the Danish government awarded a multi-year, Dkr 15 million ($2.5 million) research contract in CIM to the Driftsteknisk Institut (Institute of Industrial Engineering and Production Management) at the Technical University of Denmark. The project funded by this award was called CIM/GEMS, and its goal was to develop and transfer CIM tools and technology to small and medium-sized manufacturing companies in Denmark (200 employees or fewer). The CIM/GEMS project was composed of a university development group and an industry "follow group" made up of participants from twelve companies.

The university group undertook all technology assessment. Its primary task was to evaluate and apply selected tools, methods, and technology in building a complete laboratory CIM factory at the university. This factory fulfilled two functions:

(1) As a research and development site, it permitted the project to build a CIM facility following the phases of Life-Cycle Engineering (see Figure 1).

(2) As a site for knowledge and technology transfer, it served as a laboratory factory and demonstration site for teaching industry participants about CIM.

The focus of the project's CIM development was on managing the indirect labor and costs behind manufacturing's direct process operations. The project chose to computerize a small manufacturing system in the Institute that had been used for ten years to train Danish manufacturing engineers. This factory of seven workstations produced several types of wooden rulers. Since the process operations were straightforward and well-established, the project could stay focused on the information processing and indirect activities that supported direct labor and manufacturing. This laboratory facility--though smaller and simpler than its real-world counterparts--was still a complete manufacturing facility requiring a substantial amount of information processing to run (see Figure 2). For our project, it was an ideal facility for experimentation and training.

Because the work of the CIM/GEMS project was aimed at small and medium-sized companies, the project established the following general requirements for building a new CIM Rulers Factory:

Figure 1. CIM project development life cycle

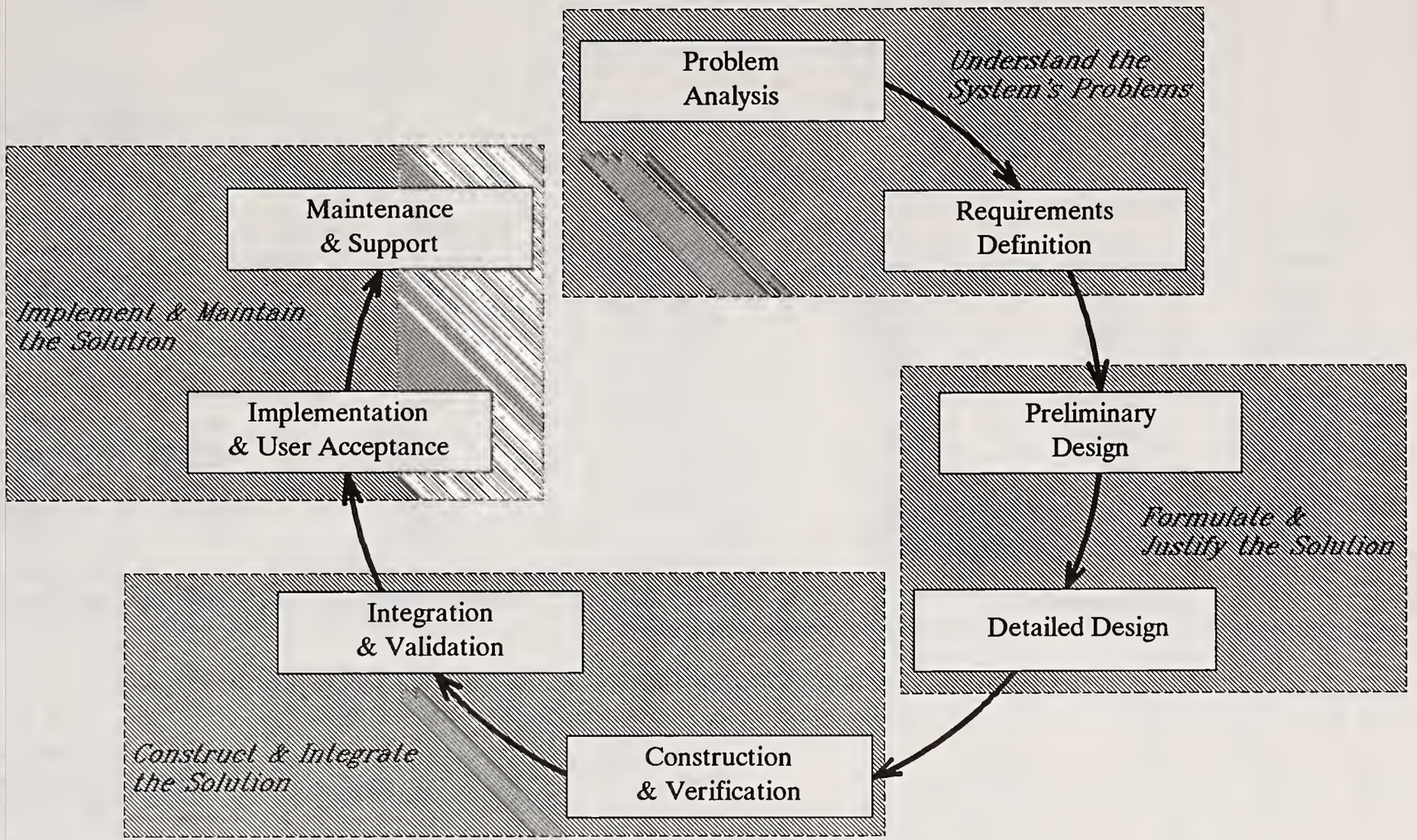
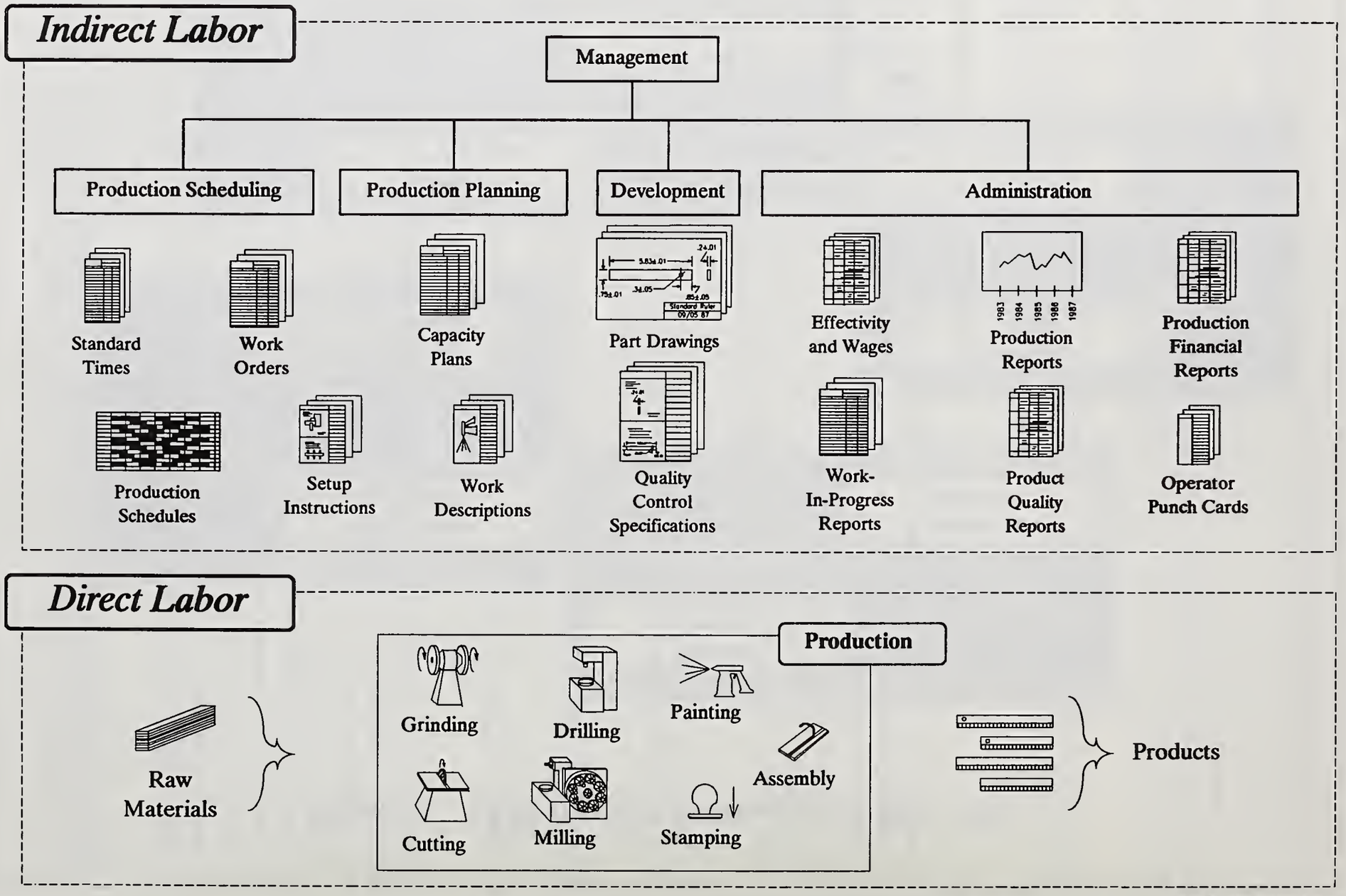


Figure 2. Rulers Factory Manual Manufacturing System



* The factory would be low-cost and use only PC technology.

* The manufacturing facility supported by PCs would include both manual and automated operations, which is a typical configuration in smaller companies.

* The factory would be upwardly expandable, that is, able to add and manage more information and new stations/operations without changing the basic system.

* The design, development, and installation of the CIM factory could be done by manufacturing people and engineers without requiring computer specialists.

3. CIM system development approach

To accomplish the project objectives we selected a proven toolset and approach developed by the USAF ICAM Program. The approach was based upon the System Engineering Methodology (SEM) [SEM85] and the ICAM Definition (IDEF) tools: IDEF0 and IDEF1 [IDEF081, IDEF181]. Although an IDEF2 was developed [IDEF281], it is an unsupported simulation language. Instead of IDEF2, we used the SIMAN simulation language and CINEMA animation package from Systems Modeling Corporation [SMC].

The System Engineering Methodology is a detailed description of how to establish and manage a CIM development project. It describes how to use the IDEF toolset to transform a manual manufacturing system into a computerized system. IDEF0 is a function modeling technique for modeling a system's activities and the information flowing among them. IDEF1 is an information modeling tool for specifying a CIM system's information structure and requirements. It is used to model the structure of the information flowing among the activities defined with IDEF0. SIMAN was used to model the CIM system's time-varying behavior. Like IDEF2, it allows you to first model the system as a node/arc graph and then translate the graph into a simulation program for analysis. CINEMA was used to build a simulation driven animation of the Ruler's Factory as a manual system and as a CIM system.

Taken together, these tools formed a toolset that supports the development of a CIM system throughout the development life-cycle. However, the Systems Engineering Methodology and the IDEF tools were developed during the late 1970s and early 1980s and reflect a centralized computing, mainframe oriented mindset. To achieve our project goals, the tools had to be modified and extended so that computer inexperienced manufacturing personnel could use them to build distributed CIM systems.

3.1 IDEF0

IDEF0 is a dataflow technique that evolved from SADT [ROSS85]. SADT consists of two dataflow techniques, function on arc with the information flow on node, and function on node with the information flow on arc. IDEF0 is equivalent to the latter, it has function on node with the information flow on arc. It was adopted by the ICAM program office to support their concept of manufacturing architectures and as a modeling technique to support the analysis and design of CIM systems. For an excellent discussion on the concept of *architectures* as used here, please see

Zachman [ZAC87]. The use of IDEF0 as an analysis and design tool is discussed in the System Development section later in the paper. The intent of this section is to describe IDEF0.

Key elements of IDEF0 are its graphical representation and its hierarchical structure. Figure 3 shows the basic syntax and hierarchical structure of an IDEF0 model. As a function modeling technique, IDEF0 is used to model manufacturing activities that transform input into output under constraints. An activities' inputs , constraints, and outputs are information. Mechanisms can be information, devices that facilitate the function activity, or an IDEF0 function in another model. The concept of a mechanism as a function in another model allows multiple models to be interconnected and provides a powerful tool for modeling complexity. In addition, a diagram at a lower level can be a function in more than one diagram at a higher level. An example of this convergence is shown at the lowest level in the hierarchical structure of Figure 3.

As a model evolves from the general to the specific, activities and their information flows are decomposed into more and more detail. This *topdown* approach allows the complexities of manufacturing to be systematically organized into a logically consistent model and represented in a graphical context easily understood by management and manufacturing personnel. The graphical representation allows analysts to easily walk these people through a model for validation reviews on models of existing systems and to describe to them the structure of proposed systems. A more detailed explanation of the IDEF0 technique is given in [IDEF081 and ROSS85].

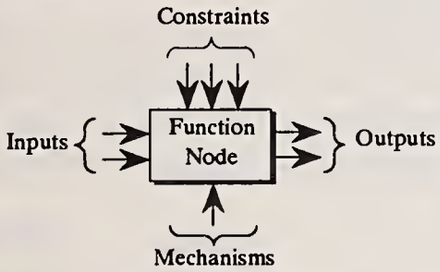
Another important concept of IDEF0 is its explicit use of *context, viewpoint and purpose. Context* establishes the subject of the model as a part of a larger whole. It is accomplished by a textual description and by defining the bounds within which a model exists using a high level node diagram. *Viewpoint* specifies the perspective from which the model is constructed, and *purpose* establishes the reason why the model is created.

In updating IDEF0 for use in the CIM/GEMS project, we expanded the use of viewpoints to require multiple models with specified viewpoints. We model the manufacturing system from a process flow viewpoint and from a documentation viewpoint. The process flow viewpoint models the activities that are needed for the physical production of goods. The documentation viewpoint models the documents used to manage and control the activities shown in the process flow model. In addition, the documentation viewpoint models the relationship between all the documents through its information flows. The documentation model is connected to the process flow model through the mechanism approach described earlier. The connection of these two models allows explicit representation of what documents are needed at each activity in manufacturing. Together with the IDEF1 model, these two models can be used to identify the data, transactions, and screens that are needed to support each manufacturing activity in a computerized system.

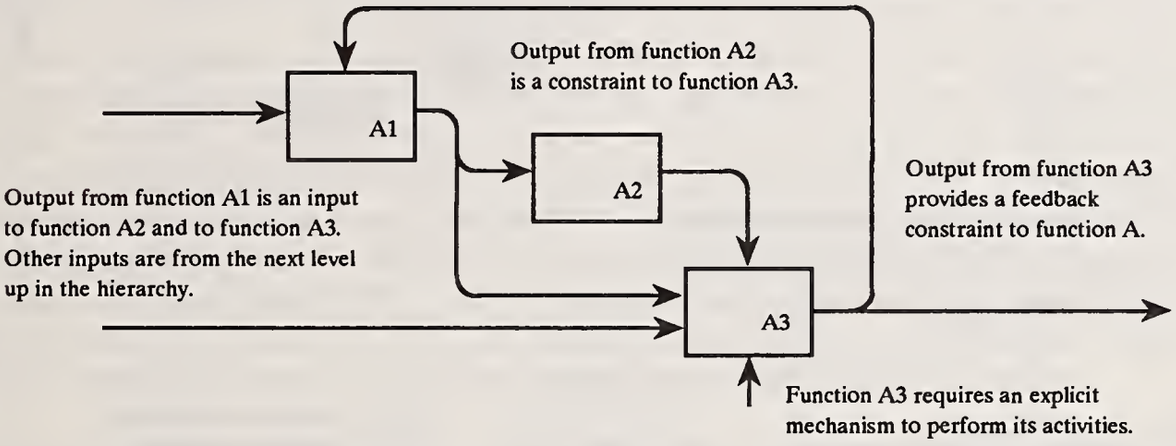
The manufacturing system is modeled for different purposes as we move through the lifecycle shown in Figure 1. In the A2 level of Figure 4, two purposes are identified for the modeling efforts -- the *AS-IS system* and the *TO-BE system*. The purpose of the *AS-IS system* models is to model the existing manufacturing system from the two viewpoints previously discussed. These models represent a definition of the current system. The purpose of the *TO-BE system* is to model the manufacturing system as it should exist after computerization. As before, two viewpoints are used.

Figure 3. The IDEF0 methodology

IDEF0 Basic Syntax



A node is an activity or function that transforms inputs into outputs under constraints using mechanisms. The inputs, outputs, constraints, and mechanisms are directed information flows that connect functions in IDEF0 models. Often the mechanisms are not shown because they are understood by the audience. The information flows to/from a function in a parent diagram at one level in the hierarchy match the information flows to/from its decomposition at the next lower level.



IDEF0 Hierarchical Structure

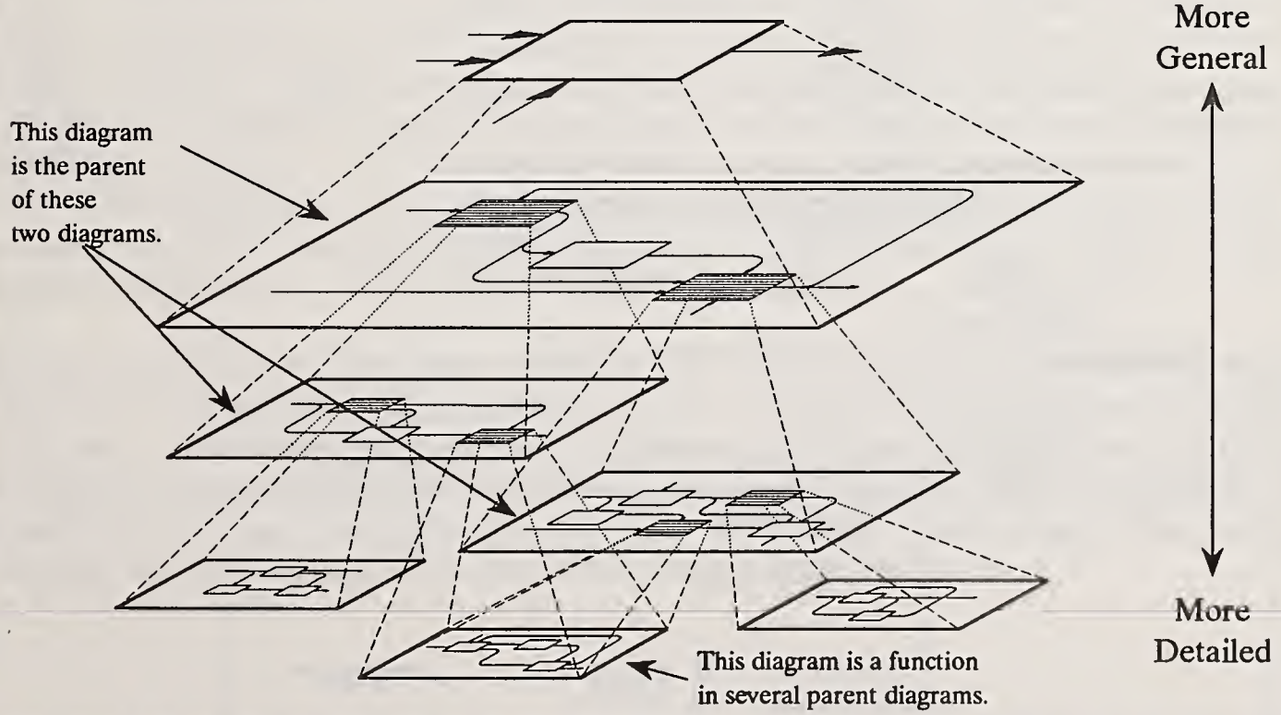
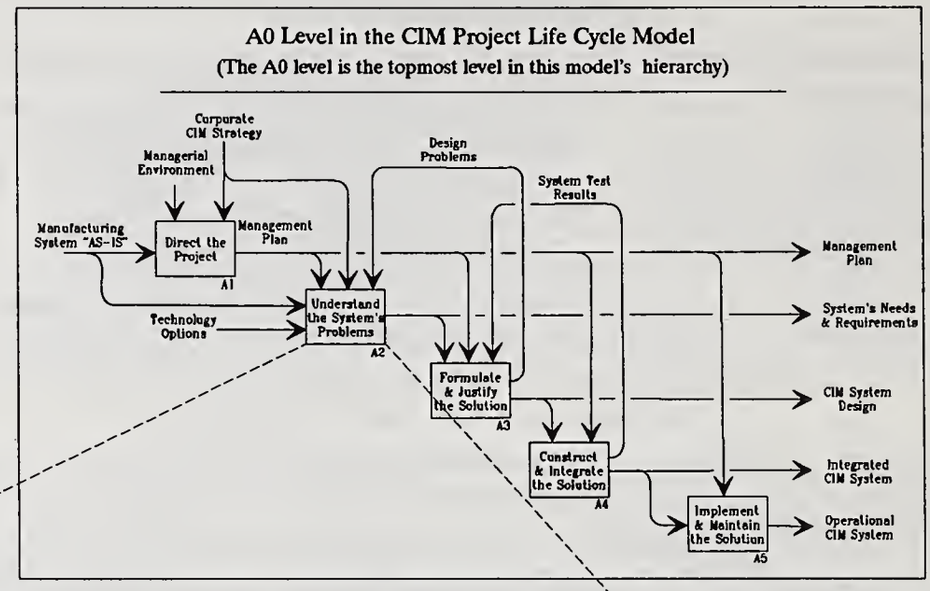
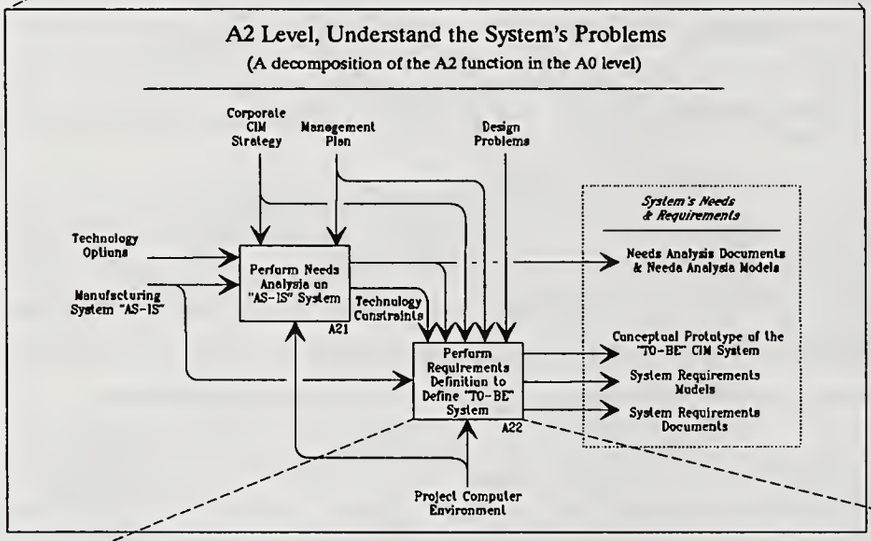


Figure 4. A breakout of activities for specifying system requirements in the CIM/GEMS project

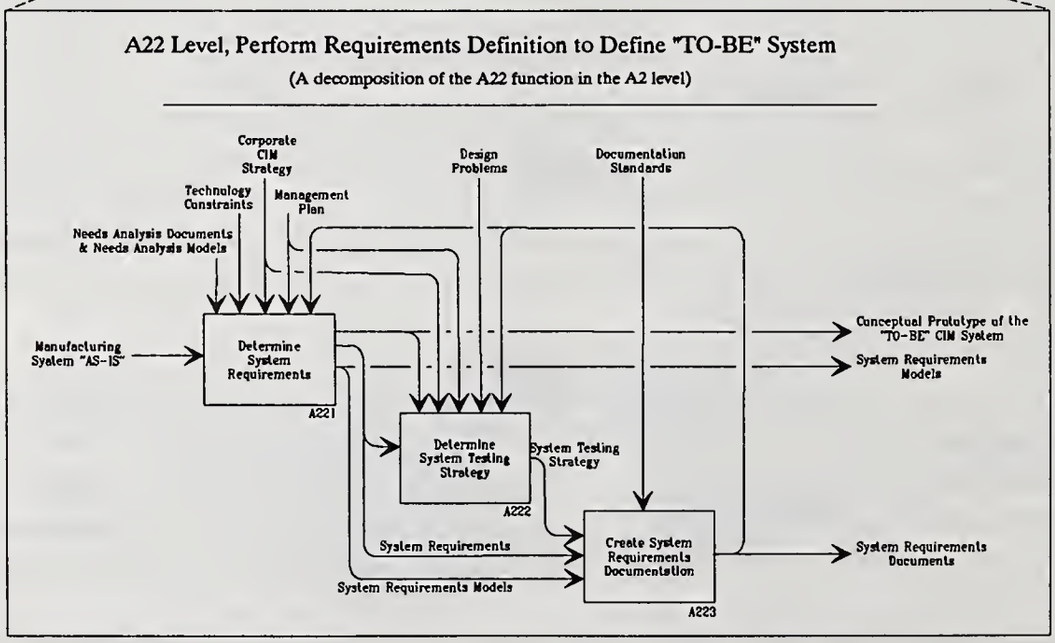




Note: Mechanisms are specified at this level but are implied at other levels and thus not shown.

More General

More Detailed



However, in this case, the documentation model represents a model of the screens to be used in the computerized system and the information flows between them. Together the TO-BE models represent a definition of how the system should operate. They can be used to define requirement specifications as shown in the A2 and A22 levels of Figure 4.

Throughout this discussion we have emphasized the use of IDEF0 and not its syntax. The IDEF0 syntax is identical to SADT and is quite simple. By itself it is of limited value. Only through its use in a specified manner and combined with other techniques does it become a powerful tool. The next section discusses one of the other techniques, IDEF1.

3.2 IDEF1

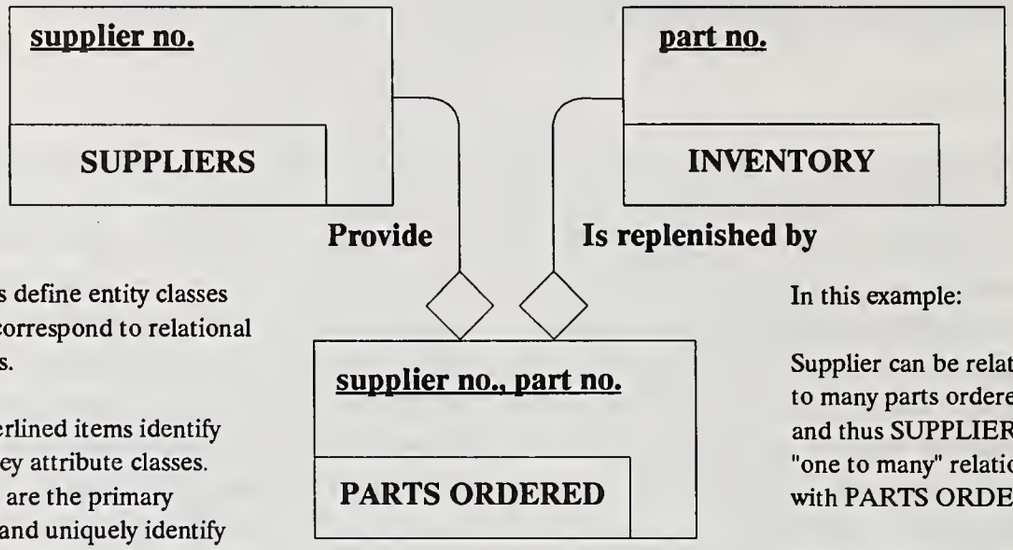
The ICAM program actually developed two versions of IDEF1 information modeling tool, IDEF1 [IDEF181] and IDEF1X [IDEF1X85]. Both versions are Entity-Attribute-Relationship models and evolved from Chen's ER technique [CHEN76] and Nijssen's ENALIM technique [NIJ79]. IDEF1 was initially developed only for requirements specification. It was designed to then be translated into a design for a hierarchical, network or relational database. As relational database technology began to dominate manufacturing users realized that with a few extensions an IDEF1 model could explicitly define a relational database's logical structure. IDEF1X was developed to meet these needs by using a different graphical syntax with some improved semantic richness. Although it met these goals, IDEF1X did so through increasing the completed model's complexity. After evaluating IDEF1 and IDEF1X , we decided that with a few simple extensions IDEF1 was equivalent to IDEF1X and that models built using it were considerably less complex.

In contrast to IDEF0, IDEF1 is <u>not</u> a flow diagram. Although like IDEF0 it uses a node/arc construct, unlike IDEF0 its nodes and arcs are passive. The nodes and arcs do not represent transformations and information flows -- an IDEF1 model is a static structure that defines information groupings and relationships among groupings. It is tool to organize information so that it can easily be located and referenced or changed. Because it can define information groupings and relationships among groupings, it can model the business rules that define how an organization operates. Figure 5 shows the basic syntax of an IDEF1 model and lists the five phases of the modeling procedure. The phases are grouped into two stages and produce a completed IDEF1 model. Consistent with Zachman's architectural concepts[ZAC87], Stage 1 results in a business model that defines the manufacturing system in the terms of the users and Stage 2 results in a logical definition of the relational database that may be difficult for users to interpret. Figure 6 shows a mapping between a Stage 2 IDEF1 model and table definitions written in SQL.

Although Rouf [ROUF84] notes that a completed IDEF1 model is in approximately third normal form, it can be shown that an IDEF1 model completed through Stage 2 will always be in third normal form. In addition, by incorporating determinant analysis as defined by Howe [HOW83] with extensions from Smith's dependency diagram technique [SMITH85], each entity class can be analyzed and its normal form explicitly determined. This level of detail is useful when writing transactions and as documentation for future maintenance work on the database.

Figure 5. The IDEF1 methodology

An example of the IDEF1 syntax



* Boxes define entity classes and correspond to relational tables.

* Underlined items identify the key attribute classes. They are the primary keys and uniquely identify a row in a relational table.

* A connection between two boxes defines the relationship between them.

In this example:

Supplier can be related to many parts ordered and thus SUPPLIER has a "one to many" relationship with PARTS ORDERED.

Inventory can be related to many parts ordered and thus INVENTORY has a "one to many" relationship with PARTS ORDERED.

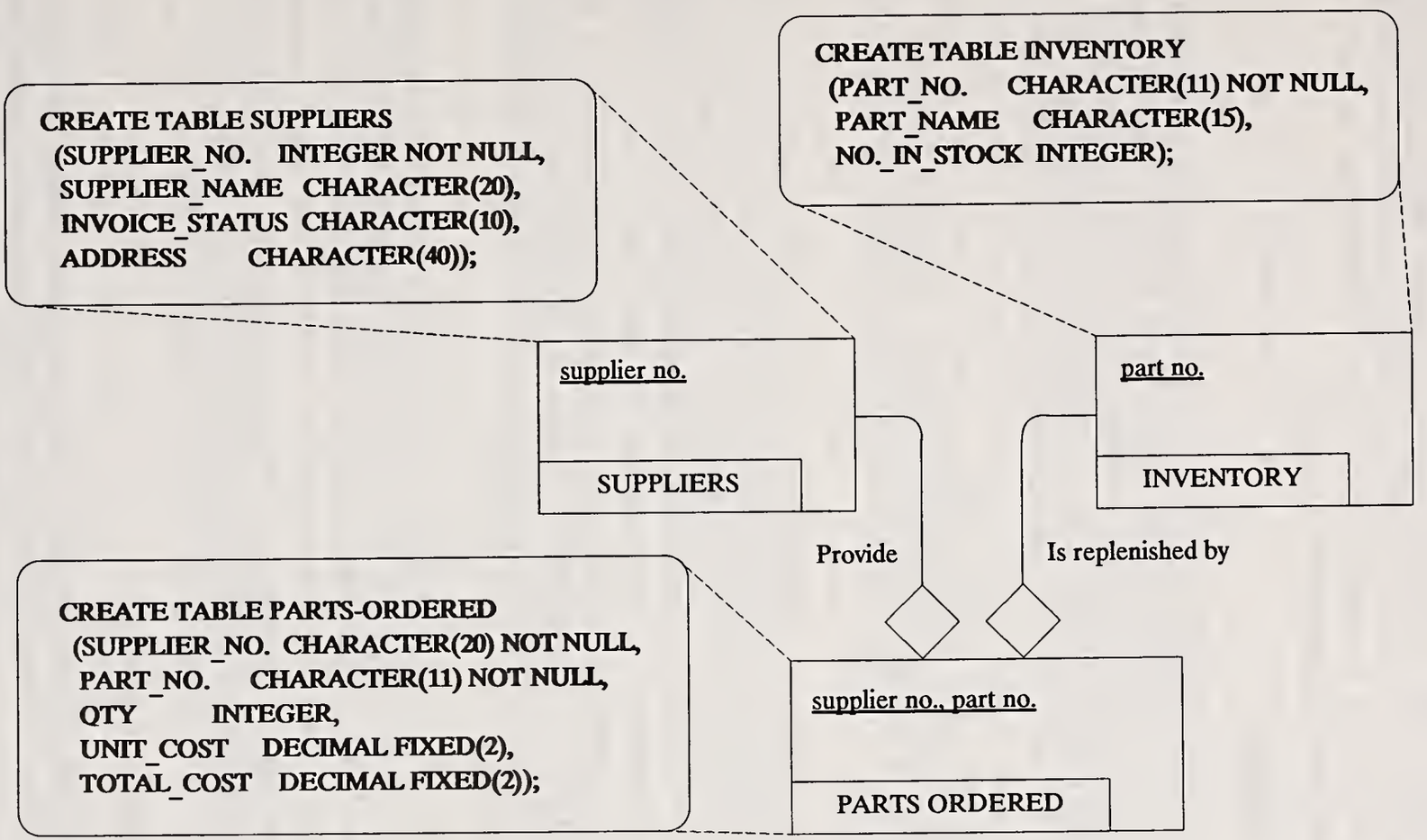
The five phases of the IDEF1 modeling procedure

Stage 1 - Results in a business model of the manufacturing system
- Phase 0 - Establish context, viewpoint & purpose
- Phase 1 - Determine the entity classes
- Phase 2 - Determine the relationships among entity classes

Stage 2 - Results in a logical definition of the relational database
- Phase 3 - Determine the key attribute classes and entity class dependencies
- Phase 4 - Determine the non-key attribute classes

Figure 6. Creating a relational database from an IDEF1 model

Each "box" has detailed information associated with it that
can be used to define a table in a relational database.



The completed Stage 2 IDEF1 model is used to identify the entity classes needed by each low-level function in the IDEF0 models. For each function, a separate IDEF1 submodel is extracted from the IDEF1 CIM system model. Each submodel shows only the entity classes and their relationships that contain the information flowing to and from the IDEF0 function. Each IDEF1 submodel defines the *external schema* for an IDEF0 function and identifies the relational tables that will be used to computerize the function.

3.3 Simulation

Examining the time-varying behavior of a system is crucial in assessing its current behavior and in predicting its behavior as a CIM system. Even though we focused upon computerizing manual operations and not replacing the manual system with process automation, it is quite possible that replacing paperwork with computers will degrade system performance. Our objective in using simulation was to determine the number of transactions per second that the database management system must process to maintain the current manufacturing throughput rate. Detailed descriptions of SIMAN and CINEMA [SMC] as well as descriptions of comparable tools such as SLAMSYSTEM [PA] are available elsewhere. Instead of describing a particular tool, we will describe how we used simulation to specify the database performance necessary to support the CIM system.

We first modeled the manual system and established a baseline model that matched the actual system in structure and performance. We created a scenario for the CIM system by estimating when transactions would be made during the operation of each station and then using the simulation model, we determined the maximum time in which a transaction must occur to maintain production at its current rate. We constructed a CIM system model by including in the simulation model the database management system, and transactions to and from the database at each station in the manufacturing system. By exercising the simulation model, we were able to determine the minimum transactions per second the database management system must process to maintain paroduction at its current rate.

The SIMAN simulation program was used to drive a CINEMA-based animation of the AS-IS and To-BE manufacturing systems. Animation was used to visually verify the correct sequencing of activities in the manufacturing system and to demonstrate to our industrial coalition the concept of simulation. The data analysis capabilities of SIMAN were used to analyze the empirical results and establish the database performance specifications. Simulation analysis ensured that the database software and hardware we selected would meet the CIM system's needs. The completed CIM system performed as predicted by the simulation analysis.

3.4 Development approach

The CIM development life-cycle as shown in Figure 1 defines the high level tasks that must be accomplished to develop a CIM system. Using the System Engineering Methodology (SEM), we developed IDEF0 models of the tasks by decomposing them into subtasks. The decomposition created a work-breakdown structure for our CIM system development project. Its information flows defined the task sequence and its feedback loops defined the task overlap structure, and

from analyzing its hierarchical structrue and its information flows a project schedule was developed. The model provides a roadmap of how to organize a CIM development project. Figure 3 shows the life-cycle represented as an IDEF0 model with an example section decomposed into more detail.

The System Engineering Methodology as originally conceived viewed the development process as a sequence of activities following the traditional waterfall model from software engineering. Because of the price/performance of todays PCs and their highly interactive interface, we modified the basic SEM approach to reflect a highly iterative process driven by successively more sophisticated prototypes. By using a prototyping approach, early in the development life-cycle: 1) users have a sense of the system's *look -and -feel,* allowing them to influence the development; 2) developers gain insight into the technical issues ensuring that the CIM system is technically feasible; and 3) problems and issues in the manufacturing system are uncovered early that would otherwise late in the project.

Using the SEM approach, IDEF0, IDEF1 and simulation models of the AS-IS system are constructed. The process of constructing these models identifies inconsistencies and structural errors in the existing system and is the problem analysis phase. This is followed by correcting the problems and incorporating new capabilities into the manufacturing system through modifying the models. The models then become the definition of the proposed CIM system's functional and information structure, and its expected performance. Prototypes of key areas validate the models. The prototypes also provide a definition of the interface's look-and-feel and a sense of how the completed system will operate. The prototypes include implementation of the database, transactions with the database, and user screens.

The models and the prototypes become part of the CIM system's requirements specification, completing the first phase in the CIM development life-cycle. At this point in the life-cycle, manufacturing personnel have sufficient insight into the manufacturing system's complexity and into the technical issues associated with computerization to decide whether to continue the development on their own or to contract the development to an outside group. If they contract the development then the system requirements can be used to specify the contractual requirements. If they decide to continue development then the requirements become a roadmap of what must be done.

The key element in our approach is to use the database to decouple the functions in the manufacturing system. All information is passed between functions through transactions with the database. This allows the functions to be implemented in isolation from the bottom up and then integrated together through the CIM system database when they are brought online with the with the rest of the CIM system. From the TO-BE models, functions at the lowest level can be implemented as a set of database transactions in conjunction with screen interfaces implemented using systems such as SQLForms [ORA]. The relational tables that will be accessed by a database transaction are already known and defined in each function's external schema.

3.5 Caveats

There are two areas in which caution should be exercised -- 1) the use of this approach when systems have time-critical tasks, and 2) the need for computer expertise in establishing a local-area network. First of all, our approach assumes that the manufacturing system does not have time-critical tasks. Such tasks severely restrict the amount of time available for determining the system's current state and deciding what to do. Such tasks are commonly associated with realtime control and require a different analysis and design approach than used in our work, and require significant computer expertise to be implemented correctly.

The second area in which we urge caution is in establishing a local-area network. We recommend implementing CIM systems using a distributed computing approach built around a local-area network. Initially, we believed that this tool could be designed and build by computer inexperienced manufacturing people. Our own experience and that of our industrial coalition indicates that the physical design and implementation of the network can be done without any prior experience. However, setting up the network software and tuning it requires sophisticated knowledge and is almost impossible for inexperienced people to do correctly. If the expertise is not available inhouse, then manufacturing personnel will have to obtain outside help to ensure the system will work properly.

4. Coordinating CIM development and Transfer

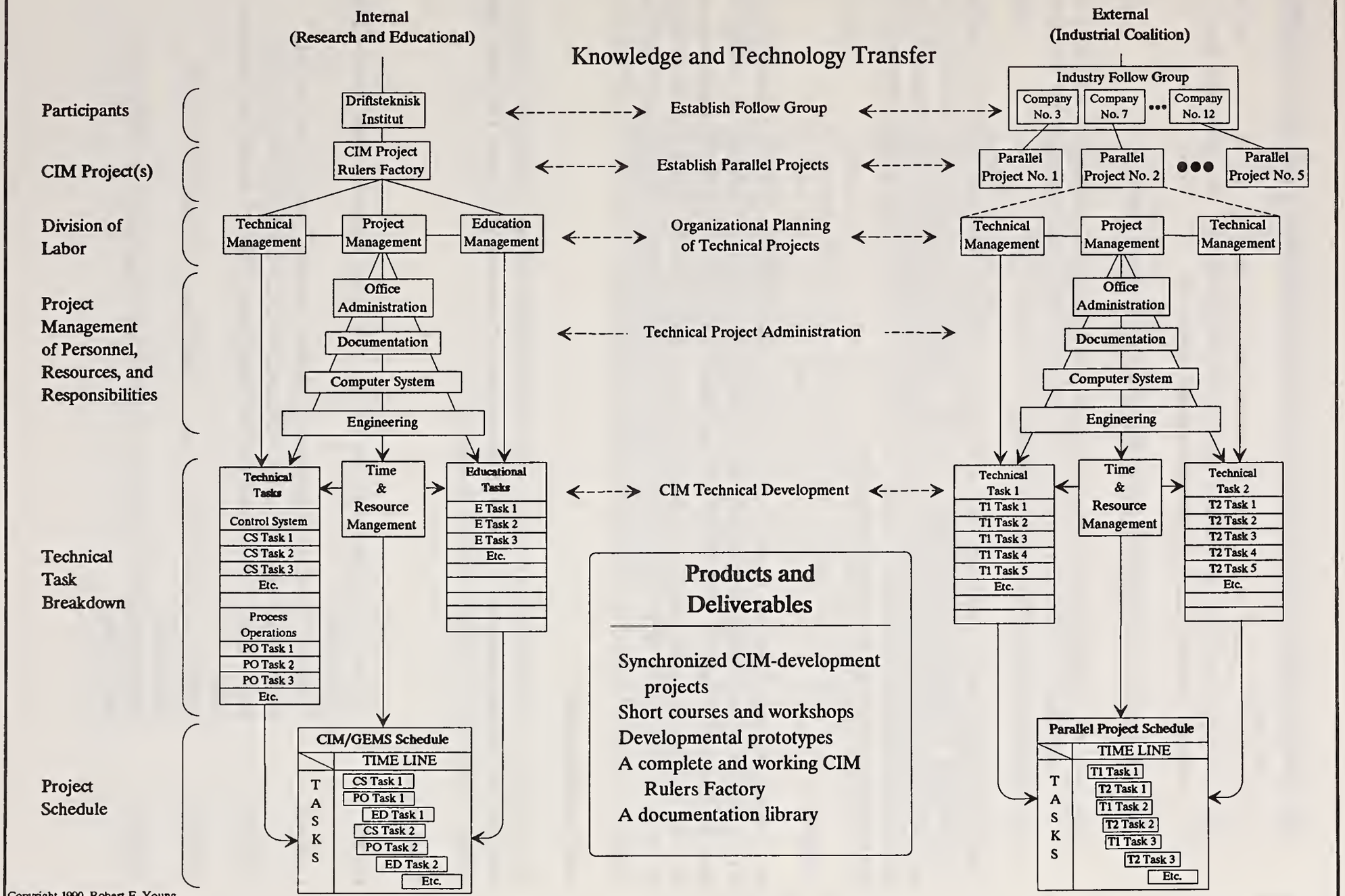
Although the above approach satisfactorily defined the CIM system development, it did not address the other obligation of the project: knowledge and technology transfer. Though more abstract, this effort also had to be specified as concrete activities and deliverables so that the results of the project's technical development could be acquired and used by participating companies.

To do this, the CIM/GEMS project first established a program of "parallel projects," in which six of the twelve companies in the follow group agreed to undertake their own in-house CIM projects in parallel with the CIM/GEMS project. They were to follow the same CIM-development life cycle, staying one phase behind the CIM Rulers Factory development in order to learn from the project's activities. Thus, the companies had both a developing laboratory system to observe and a system to build themselves. Additional assistance came from working with other developers in industry engaged in the same CIM activities at the same time. This organization of the CIM/GEMS project can be seen in Figure 7. The technical tasks of the university group on the left and the parallel project(s) on the right are coordinated and run in parallel with each other.

The chart also shows on the left the educational activities that the university group coordinated with its technical activities. These two avenues of activities came together in a common project schedule that yielded the products and deliverables that the parallel projects used to guide their own system development. These deliverables are:

(1) *synchronized CIM-development projects* (described above), which we called the "on-going CIM workshop";

Figure 7. CIM/GEMS project organization and structure



(2) *short courses and workshops* on tools, methods, and technology appropriate to each life-cycle phase;

(3) *developmental prototypes*, which were built throughout the life cycle and used to develop, test, and demonstrate system components;

(4) *a complete and working CIM Rulers Factory* at the university; and

(5) *a documentation library*, which included system-development documents describing the construction of the CIM Rulers Factory and educational materials for the short courses and prototype demonstrations.

The CIM/GEMS "on-going workshop" was a cooperative university/industry effort in two areas: (1) empirical CIM-system development, and (2) educational exchange of knowledge and experience about CIM tools and methods. The organization of the life cycle into phases made it possible to isolate specific technical activities for each phase and learn about appropriate tools and technology for undertaking and supporting them.

5. Summary

In this paper we have described results from the Danish CIM/GEMS project that demonstrated it is possible for manufacturing people with little or no prior computer experience to design and build their own CIM systems. The approach is applicable to small and medium size companies and is a means to overcome the software backlog that currently exists so that the projected growth in the CIM area can occur. In addition, we developed a novel approach to knowledge and technology transfer that allowed industry to receive technology and begin using it as soon as it is developed.

6. Acknowledgements

We would like to acknowledge the other people who participated in the CIM/GEMS project and provided the technical expertise that allowed the project to be successful: Kjartan Bergsson, Søren Jensen, Finn Jørgensen, Ellen McDaniel, and Erik Tvedt.

7. References

BW86 "Retool or Die: Job Shops Get a Fix on the Future," *BusinessWeek*, June 16 1986, pp. 105-106.

CHEN76 Chen, P., "The Entity-Relationship Model: Toward a Unified View of Data," *ACM Transactions on Database Systems*, Vol. 1, No. 1, Mar. 1976, pp. 9-36.

HOW83 Howe, D.R., *Data Analysis for Data Base Design*, Edward Arnold Ltd, London, 1983.

IDEF081 *IDEF0 Function Modeling Manual*, ICAM Program Office, WPAFB, Oh 45433,1981.

IDEF181 *IDEF1 Information Modeling Manual*, ICAM Program Office, WPAFB, Oh 45433, 1981.

IDEF1X85 *IDEF1X Information Modeling Manual*, ICAM Program Office, WPAFB, Oh 45433, 1985.

IDEF281 *IDEF2 Dynamics Modeling Manual*, ICAM Program Office, WPAFB, Oh 45433, 1981.

NIJ79 Nijssen, G. M., "Modeling in Data Base Management Systems," *Proc. Euro IFIP Conference*, Sept. 1979.

ORA SQLForms, Oracle Corporation, 20 Davis Dr., Belmont, CA 94002, 800/672-2531.

PA SLAMSYSTEM, Pritsker & Associates, 1305 Cumberland Ave., W. Lafayette, IN 47906-0413, 317/463-5557.

ROSS85 Ross, D.T.,"Applications and Extensions of SADT," D.T. Ross, *IEEE Computer*, April 1985, pp. 25-34.

ROUF84 Ruoff, K. L., "Practical Application of IDEF1 as a Database Development Tool," *IEEE Conference Proceedings CH2031-3/84/0000/0408*, pp. 408-415, 1984.

SEM85 *Systems Engineering Methodology Manual*, ICAM Program Office, WPABF, OH 45433, 1985.

SHEM87 Shemer, I., "Systems Analysis: A Systemic Analysis of a Conceptual Model," *Communications of the ACM*, Vol. 30, No. 6, June 1987, pp. 506-512.

SMC SIMAN and CINEMA, Systems Modeling Corporation, 504 Beaver St, Sewickley, PA 15143, 412/741-3727.

SMITH85 Smith, H.C., "Database Design: Composing Fully Normalized Tables from a Rigorous Dependency Diagram," *Communications of the ACM*, Vol. 28, No. 8, Aug. 1985, pp. 826-838.

ZAC87 Zachman, J. A., "A Framework for Information Systems Architecture," *IBM Systems Journal*, Vol. 26, No. 3, 1987, pp. 276-292.

HIGHLY EXTENDABLE CIM SYSTEMS BASED ON AN INTEGRATION PLATFORM

R H WESTON, A HODGSON, I COUTTS, I S MURGATROYD AND J D GASCOIGNE

ABSTRACT

CIM systems are still typically implemented in a "hard-wired" form, the resulting islands of integration taking on the external appearance of a larger machine. Such systems are difficult to modify, to expand incrementally or to interface to other islands of integration as part of an enterprise scheme.

This paper contrasts the "hard" and "soft" approaches to integration and presents the concept of a flexible integration shell to enable the implementation, debugging, management and modification of CIM systems as part of an enterprise plan.

Such a flexible integration shell is used to establish soft integration schemes in a batch manufacturing environment.

1. Introduction

The 80's have seen the emergence of a wide variety of CIM programmes aimed at creating more responsive manufacturing systems. Major advances in computer hardware and software systems have provided such initiatives with an armoury of enabling tools for this development work [CIMOSA88, JON86, WES89a, PDES89, NOF89, MAP/TOP89].

In contrast to these initiatives, examination of the situation regarding industrial CIM implementations would appear to indicate that there has been little fundamental change during this period. Industrial CIM implementations are still almost invariably "hard-wired", not only in terms of physical equipment, but also effectively in terms of their systems engineering - the glue that sticks the manufacturing entities and their applications together.

Under the hard-wired "fixed integration" approach, the required interactions between CIM entities must be anticipated at the specification stage, otherwise major upheaval will result [YOU89]. As the manufacturing environment is typically highly dynamic, it is most unlikely that all potential changes over a system's economic life can be predicted in this way. A further problem of this fixed integration approach is that it cannot cope easily with the requirement to allow for incremental, self-funding growth. This capability is particularly important for small and medium manufacturing companies, both to allow them to take advantage of the internal benefits of CIM, and to enable them to be included effectively in increasingly integrated supplier/manufacturer/customer chains.

It is only now being fully appreciated that, as systems have expanded from free-standing or simply linked pairs of machines to become computer inte-

grated manufacturing systems, the range of potential interactions (and the associated communications problems) between heterogeneous computer systems has increased out of all proportion. The enablement of these interactions represents the major problem area still requiring solution.

The scenario is complicated further by the current work on a range of standards relevant to CIM, for example MAP/TOP, PDES/STEP, SQL [ANSI86]. Although offering potential pathways towards a simplification of (rather than a solution to) the problems associated with interaction, they do not represent a panacea in themselves [WES88].

The aims of this paper are to introduce the concept of "soft" integration, its associated methodologies and tools and to indicate the roles of relevant standards. Examples of applications in both research and industrial environments are presented and conclusions are drawn based on these experiences.

The paper is split into five sections. Section 2 introduces the ideas behind soft (or flexible) integration and describes the concepts which led to the work of the Loughborough Systems Integration Group on the AUTOMAIL "flexible integration shell". Section 3 describes some research applications using the AUTOMAIL flexible integration shell. A comparative analysis of a fixed integration and a flexible integration example is presented. This section also includes work on the flexible integration of vision into CIM, and investigation into the use of PDES-like modelling approaches. Section 4 details an industrial application utilising the AUTOMAIL flexible integration concepts, the rationalle behind the work and the potential further developments in this particular environment. Finally section 5 concludes with an assessment of associated future developments in the CIM arena (including standards) and the planned work of the Systems Integration Group.

2. Soft integration: the AUTOMAIL approach

It is now widely accepted that more responsive product realisation can result from facilitating high speed access to and updating of information in machine readable form. However, as yet there is not a widely agreed upon set of generally applicable methods and tools which can be used to efficiently establish such CIM systems*. The absence of the above tools

has meant that the scope of systems integration projects has normally been limited and as a result so too have the benefits. CIM systems established today fall somewhere between the two situations depicted by Figures 1(a) and 1(b), i.e. normally only a few carefully selected product-realising processes or entities will be connected together electronically so that they can (i) share "global" information and (ii) automatically accomplish interaction between processes or entities. The majority of interaction is performed by humans and is thus poorly defined for implementation in a computer integrated system.

* Here it is assumed that a CIM system could encompass the complete range of product realising processes through marketing, inception, design, manufacture, sales, re-engineering and field support/maintenance.

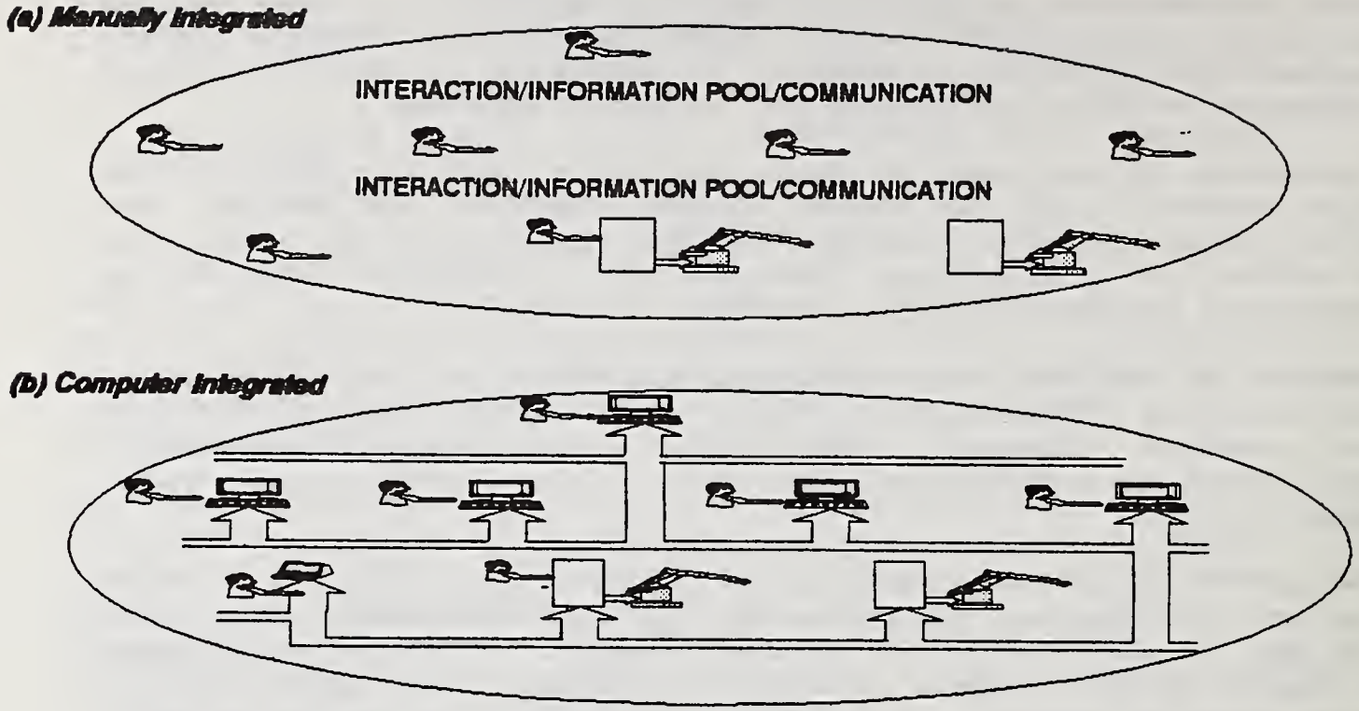


Figure 1. Computer and manually integrated systems

With existing CIM systems it is highly likely that the techniques used to achieve integration of the selected processes or entities will have led to relatively inflexible (or hard integrated) solutions, inasmuch that change in scope or function of the CIM system cannot be easily accomplished. Hard or inflexible integration will ultimately lead to the creation of a "bigger machine" which itself cannot easily be integrated into a wider scheme of things. Soft integration provides the capability to establish a programme of integration projects, each of manageable complexity, leading to CIM systems of much wider scope and a significant increase in competitive edge.

How then can "soft" or flexible integration be achieved? Finding answers to this question has been the subject of a major research study at Loughborough University of Technology. Integration scenarios of different scope and aimed at various target applications have been established through, where possible, using emerging standards and associated tools. From the findings of the Systems Integration Group so far, it is clear that:

(i) To establish soft integration it is essential that "open" mechanisms are utilised, to establish communication between manufacturing entities, govern access to and update of shared information and to establish meaningful interaction between entities.

(ii) Although tools are emerging which address specific aspects of the soft integration problem, there is as yet no unified set of tools. In aiming to minimise the cost of creating (and subsequently

expanding the scope of) CIM systems, a unified set of tools is required to enable configuration using the open mechanism referred to in (i) above.

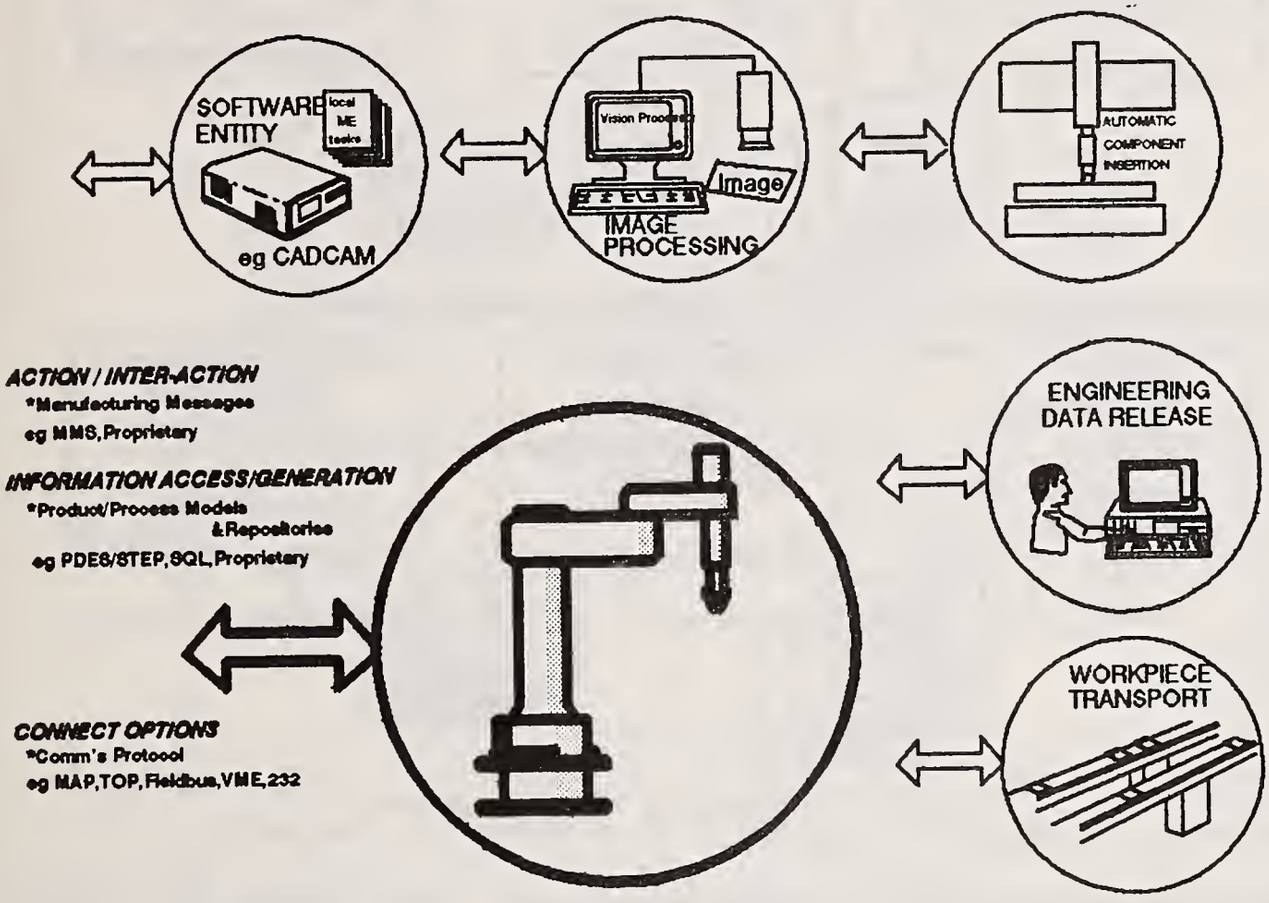


Figure 2. Generalized Models of inter-action

The above work on integration scenarios has led to a characterisation of interacting manufacturing entities and the necessary mechanisms needed to accomplish integration, as depicted by Figure 2. To date various classes of machine entity have been studied including robots, computer vision/image processing systems, workpiece transport elements and printed circuit board (pcb) component sequencing, assembly and test machines. The integration mechanisms required to establish "links" to a number of proprietary software packages (performing CAD/CAM, kinematic simulation, robot off-line programming and production planning and control) have also been investigated and various alternative solutions proposed and implemented. This work has led to an increased understanding of the integration problem and the creation of various cell and shop control systems which demonstrate integrated product realisation on an incremental basis. The schemes implemented have also encompassed interaction with entities comprising a combination of person and computer system. In such a scenario, the computer system typically provides an information/decision support capability which supports the person in a variety of roles (eg. as an operator, supervisor, manufacturing engineer, designer or manager).

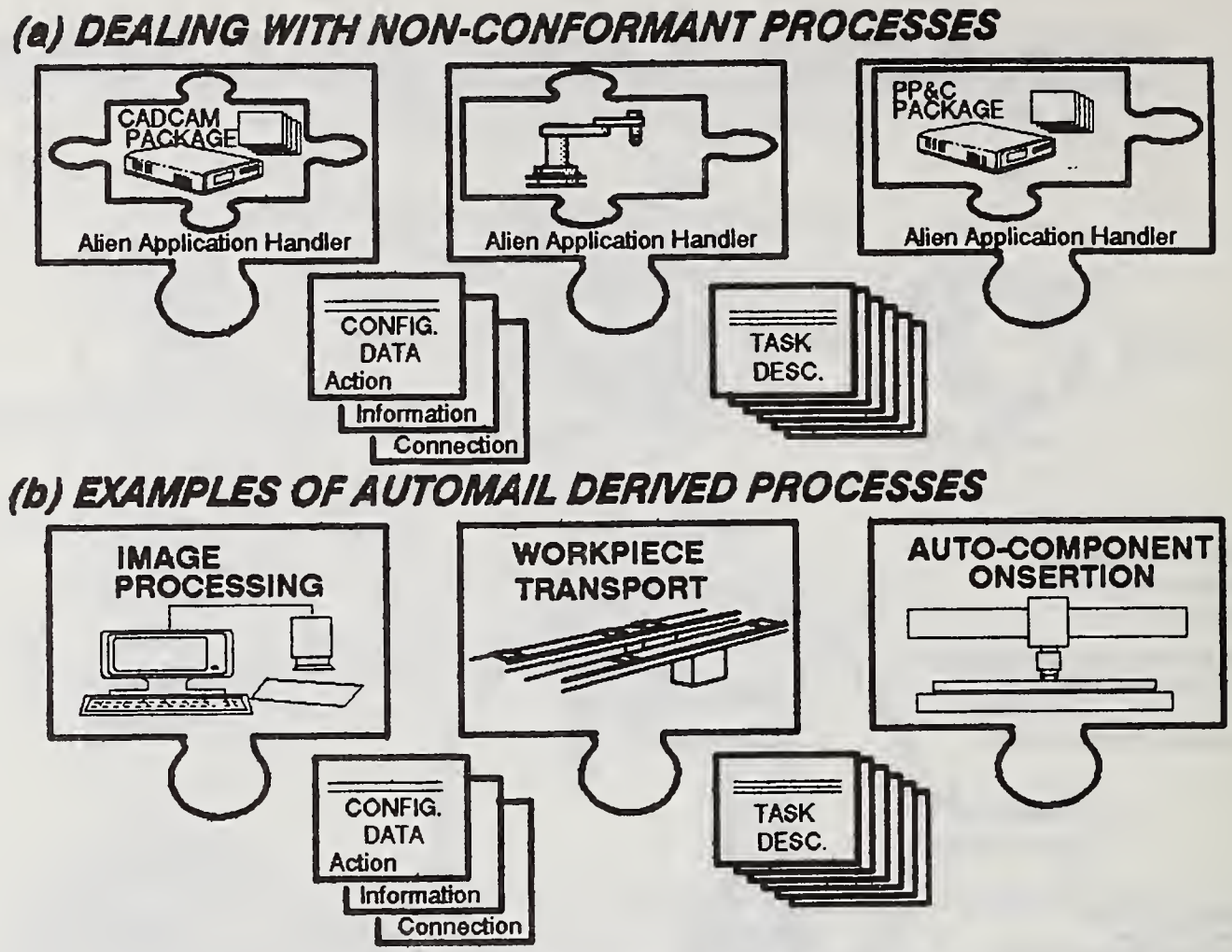


Figure 3. Creating "open" processes

From the above work an understanding has been gained of various "interaction models" and of the scope/limitations of current integration tools.

A further important theme has been the need to establish flexible integration given an existing installed base of "non-conformant" machines and software packages. This situation is representative of reality, except where a "green field" situation occurs, as most product realising enterprises will have already invested heavily in automation and/or information technology and will need to progress along a migration path towards "open" CIM. The methodology used by the SI group to deal with existing proprietary or "closed" entities is depicted by Figure 3(a), where external processing is used to establish conformance with the interaction model for the class of device concerned. Clearly in certain instances the proprietary entities will not have been designed with "data visibility" in mind. This typically implies a low level of functionality in terms of entity interactions, or the need for potentially costly "shadow" processing in the "Alien Application Handler". Figure 3(b) contrasts the approach taken when entities are designed and implemented in a form which is conformant with the SI group's "open" interaction model or "rule set".

The "open" methodology derived is highly flexible as both configuration and task descriptions are data driven, as illustrated in figure 3. The task description data defines the dynamic or run-time interactions between entities, whereas the configuration data relates to the more static relationships between entities.

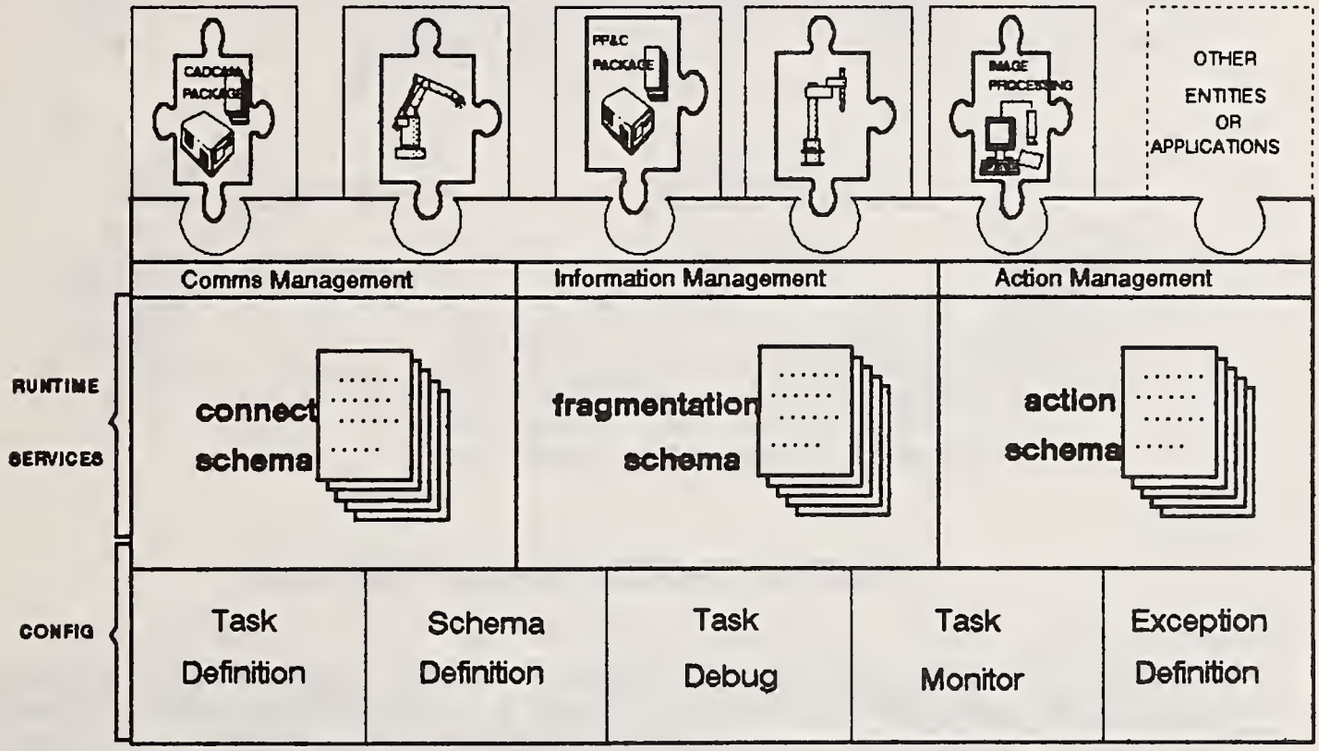


Figure 4. Automail as a platform

As indicated earlier, the SI group's "open" integration methods have evolved from extracting the generic component parts of prototype integration schemes. Initially, the work focused on the provision of task programming/configuration tools to enable the definition of entity interactions and this led to the original AUTOMAIL (AUTOMAtion Integration Language) concepts and title. Subsequently AUTOMAIL has evolved to encompass "open" debugging and runtime services. Figure 4 is a schematic of the current AUTOMAIL platform which, as depicted, provides a consistent set of services for interacting entities. In any given system a number of AUTOMAIL instances can exist, essentially providing a shell of services which can be duplicated and distributed as required in any given factory hierarchy, see figure 5. It should be emphasised that AUTOMAIL does not impose a hierarchy per se, being itself heterarchical, but it can be used to create soft integration schemes across the various organisational levels commonly found within product realisation structures.

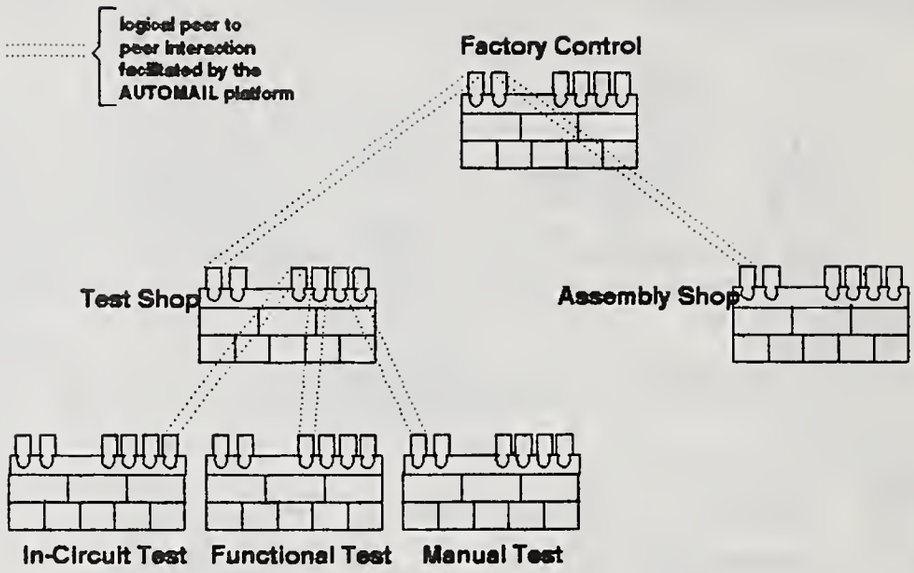


Figure 5. Multiple AUTOMAIL platforms

Previous SI Group publications [HOD88, WES89a, WES89b] have reported a significant increase in flexibility and a resulting greater resilience in the face of change can be enabled by separating the action, information and connection functions into individual architectures, see Figure 6. The AUTOMAIL methodology is based on such a decomposition with standardised access mechanisms and services incorporated to support each of these architectures.

The AUTOMAIL connection architecture provides an application process with a uniform communication interface, i.e. the underlying protocols (MAP/TOP, TCP/IP, RS232, etc.) need not be known to the application. Application processes use a standard set of communication services and perceive a flat homogeneous network. They do not require specific network address information for associated processes as this is stored by the AUTOMAIL platform. This address information can easily be accessed and configured via an engineering interface.

The action architecture provides applications with a uniform message set (historically based on OSI message protocols) to enable a consistent method of interaction, i.e. applications do not interact directly in a customised fashion. This enables the systems engineer to replace one manufacturing entity and/or its application processes without ramifications on other processes within the integrated system. Clearly, if the replacement or alternative systems do not provide an appropriate functionality, then performance will suffer irrespective of the integration approach.

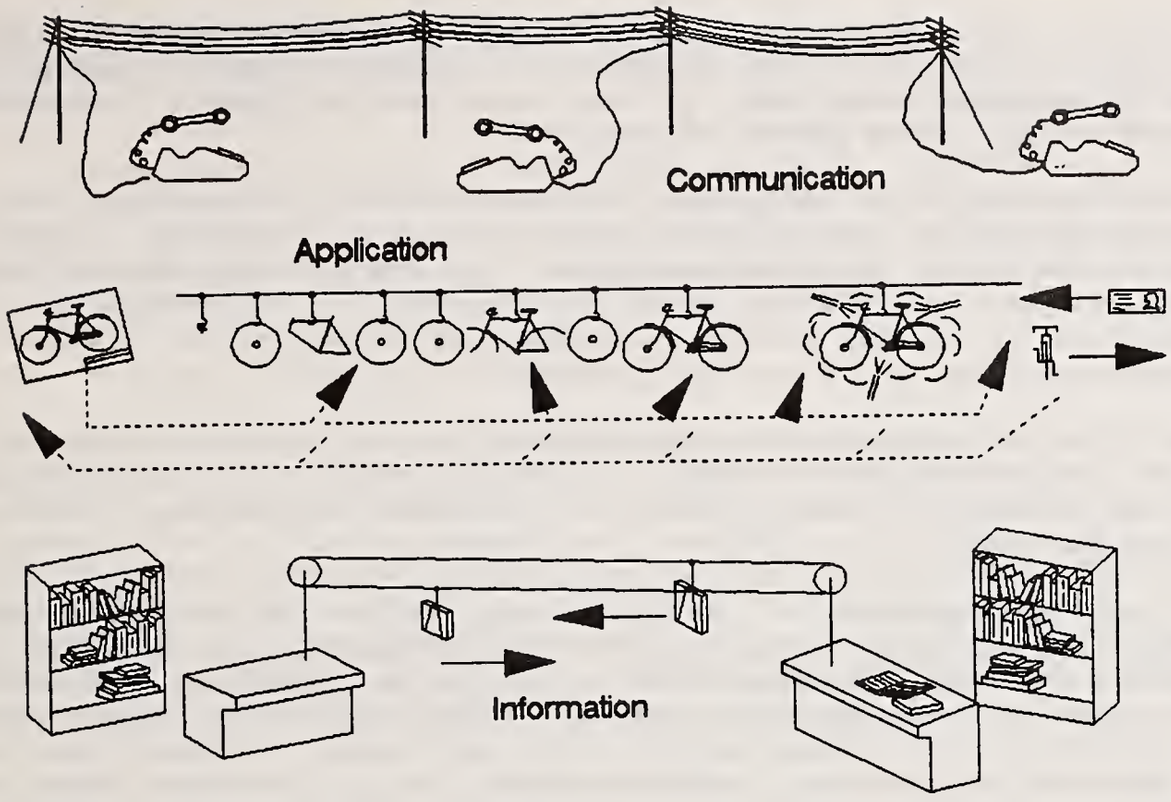


Figure 6. The three architectures

In an analogous fashion, the AUTOMAIL information architecture provides each application with a standard interface, currently SQL, into the available information resources. Thus a consistent access approach can be applied to a variety of information resources, including flat files and databases.

3. A research application

In order to take advantage of the potential benefits of CIM, a company has to go through a learning and familiarisation process. In almost all situations, the only feasible approach is to decompose the overall integration problem into a number of simpler sub-problems, each of containable complexity. Subsequently, it may be possible to integrate together the "islands of integration" so formed. However, this will typically not be the case, and even if it is possible, it is likely that severe development constraints will be imposed.

To illustrate this situation, we shall consider an industry-representative solution to a specific containable integration problem. This is one of the series of integration problems tackled by the Loughborough Systems Integration Group, with the aim of extracting generalised integration methodologies.

3.1 System description

The integration problem investigated was a relatively small-scale application. It was concerned with the provision of information support facilities

to improve functionality when accomplishing the automated optical inspection (AOI) of printed circuit boards. Two different types of machine vision systems were to be used. In both cases several types of unpopulated printed circuit boards were to be inspected.

At the commencement of the project the main functional improvements envisaged related to (i) reduced training cycles (i.e. a reduction in the design-to-manufacture time for new boards) and (ii) reduced machine setup times (relating to subsequent batch manufacture of all board types). The concept was to utilise during inspection operations product information created previously during the design process.

Both inspection stations to be incorporated were to be capable of carrying out the complete inspection task, thus another objective of the investigation was to compare the differences in information support requirements of the two stations.

The hardware components of the system comprised an IBM personal computer on which the CAD system resides, a SUN 3/60 used as the cell controller and general file store, an Adept SCARA robot equipped with Adept area vision and an additional IBM personal computer equipped with Matrox vision system. Two proprietary CAD systems were investigated, namely the Racal REDAC and the Personal Cad System Inc. PCAD packages. Only one of these, the PCAD package, was to be integrated into the system. PCAD was chosen because its output format, PDIF, was formally defined and was based loosely on the evolving Electronics Data Interchange Format (EDIF) Standard [EDIF90].

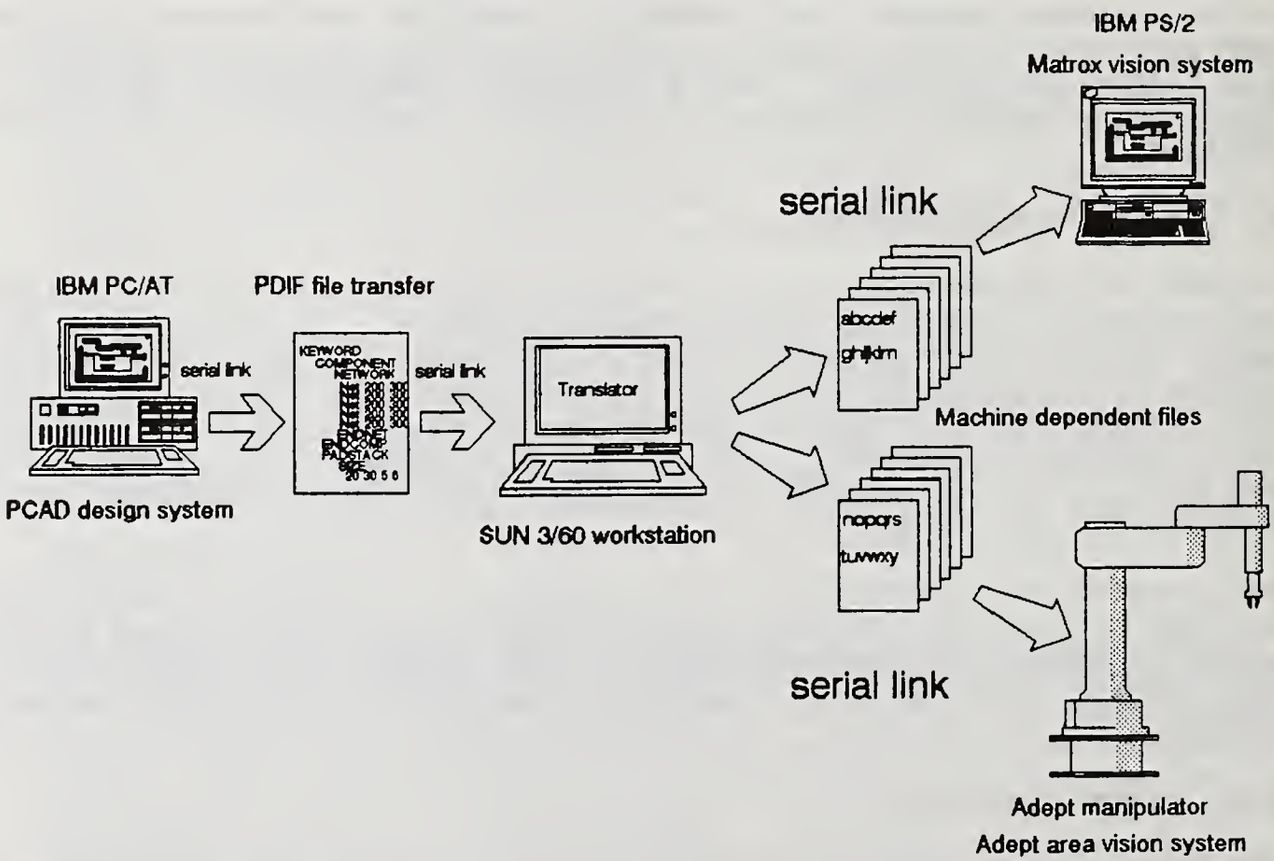


Figure 7. PCB inspection cell 'hard' solution

Section 3.2 describes the first approach used by the Systems Integration Group to accomplish integration of the bare board inspection system. Figure 7 illustrates this solution, which can be considered to correspond to a relatively hard or fixed integration approach. Subsequently the integration scheme was re-implemented in soft form, based on the use of the AUTOMAIL platform. Section 3.3 describes the soft solution, thereby enabling comparison of the two approaches and their associated characteristics with regard to expandability.

3.2 A "hard integration" solution

PCB layouts are designed on the PCAD package which produces a proprietary (PDIF) output file. This data file includes information relating to board geometry, electrical connectivity maps, component types and their locations, and drilling information. The UNIX tool YACC (see figure 7) is used to parse the PDIF file to produce two machine-dependent files, each of which contains the information necessary for its target inspection station. These files are downloaded to their respective stations in order that the inspection tasks may take place. All links between hardware devices are serial RS232, with KERMIT being used as the file transfer mechanism.

A system such as this, although integrated, must be considered a "hard" solution since its ability to accommodate internal change and to be integrated with other ad-hoc integration schemes factory-wide is limited. For example, where one of the inspection stations changed (eg. for one providing increased functionality) the translator would require a considerable amount of re-engineering to provide the new device with its particular machine-dependent file format. Another likely requirement for change in integrated systems concerns the need for ongoing functional enhancements as they are identified. For example, it became clear in this case that two new main areas of enhancement could result from:

(i) Calibrating and re-utilising (in subsequent assembly and test operations) a data model which corresponds to a specific (calibrated) instance of the CAD-generated model of the board. In this way, many manufacturing tolerance problems could be overcome, hence offering potential for much-improved product quality and reduced reject rates.

(ii) Using fragments of the information measured during optical inspection for purposes of work-in-progress monitoring and to generate trend/-quality/productivity/traceability information, as required.

This type of enhancement could be described as widening the scope of the problem.

With regard to necessary architectural changes to the hard integrated system so that enhancements of this type could be made, it should be clear that considerable systems engineering effort would be involved. It is probable that little of the original hard integration software could be reused. There is a square law relationship between the number of interacting devices and the number of potential interactions. Hence, as we widen the scope, the systems engineering problems can rapidly overwhelm the available resources and the capabilities of the personnel involved. This, allied

with the probability of discarding much of the previous expensive software development work each time enhancements are required, makes changes very difficult to cost-justify.

3.3 A "soft integration" solution using the AUTOMAIL platform.

An alternative soft solution, which draws upon recent Loughborough Systems Integration Group research is presented. The AUTOMAIL methodology is used, and reference is made to relevant evolving standards.

The processes required to achieve the example manufacturing task have been discussed in the previous section, but here their implementation is facilitated using the AUTOMAIL methodology described in section 2.

An AUTOMAIL platform was configured and installed on the SUN 3/60 as illustrated in figure 8, and the translation (YACC) process described earlier was re-implemented as an AUTOMAIL process. As this had originally been written in-house, it was considered preferable to re-implement it as an AUTOMAIL process, rather than to produce an alien application handler which would achieve a similar result but would not offer the same level of data visibility.

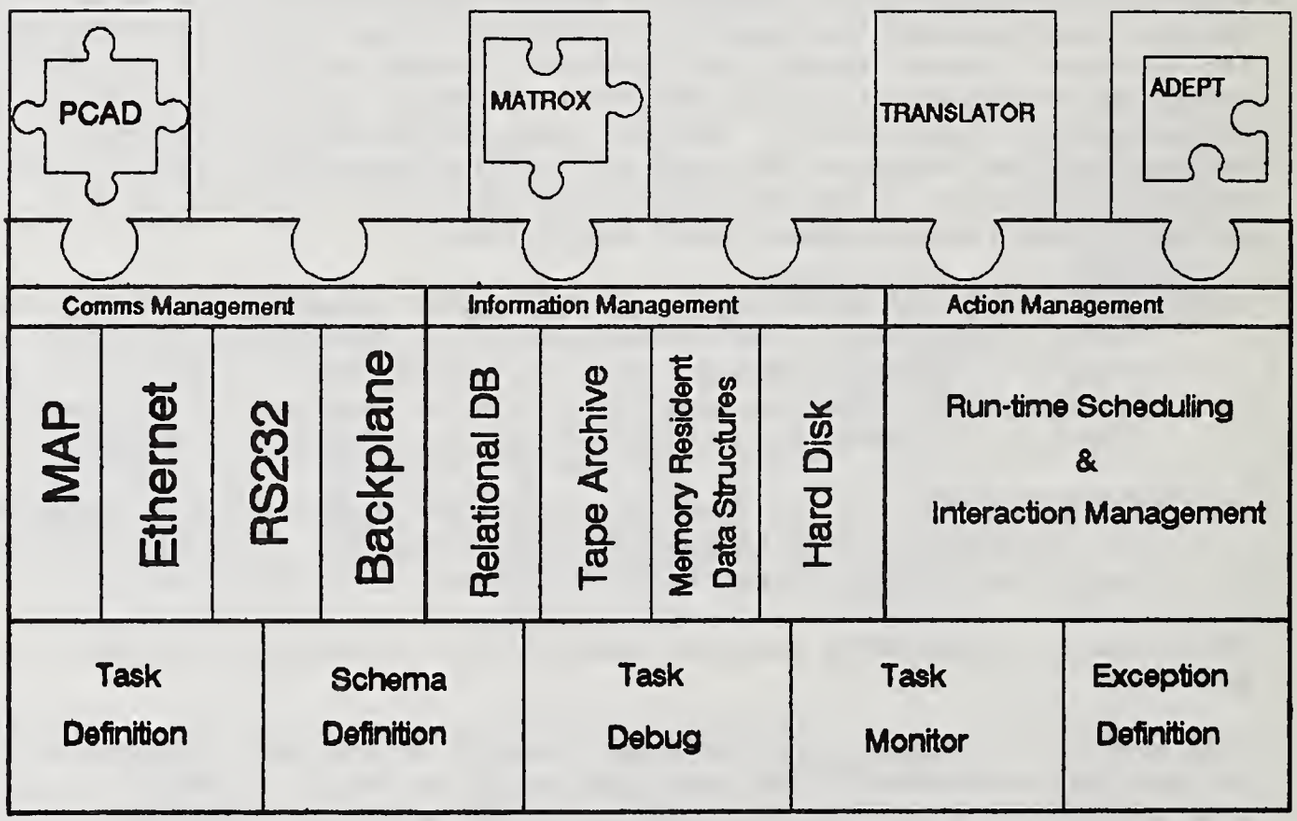


Figure 8. 'Soft' solution

Alien application handlers were implemented for both inspection stations and the CAD personal computer station. This allowed them to be viewed ex-

ternally as AUTOMAIL-conformant processes. To achieve this required not only the standardisation of their communications interface, but also the way in which they interacted with other processes and viewed the information services.

An immediate benefit of using a soft integration approach can be seen when considering increasing the functionality of an integrated system. Additional AUTOMAIL processes can be accommodated by the platform without any major re-engineering of existing integrated components. Thus the scope of an integrated system can be incrementally increased as need dictates.

Another major benefit is the ability of a system designed with limited functionality and scope to be integrated with other similar systems because of the use of a consistent underlying integration methodology. This is seen as not only desirable, but fundamental to the implementation of large integrated projects, i.e. this ability initially to decompose problems into small containable sub-problems and subsequently to build the corresponding implemented solutions into larger CIM systems

As stated in the introduction, the use of evolving standards offers simplification of the problems associated with systems integration. Significant work has already been carried out by the Systems Integration Group on the use of MAP (MAP/TOP 89) as a standard communications protocol. Currently, an EDIF sub-committee is producing a conceptual model for PCB layout for which it is intended to include data useful for manufacture, an area not currently addressed by the EDIF standard. This conceptual model will be produced in the EXPRESS data modelling language [EXP89] which is also the language adopted by the PDES/STEP initiative. This model will be used at Loughborough as the basis for a number of PCB inspection application scenarios over and above the simple bare board inspection described earlier, for example, populated board inspection. In addition to the provision of data for PCB inspection, the model will contain information for other PCB manufacturing tasks (eg. manual assembly, automated assembly, test, etc.). Each application will require a specific partial view of the global product model. Future work at Loughborough will include the investigation of both the information requirements of each application, and the management of its presentation.

The availability of evolving standards does not in itself enable improved integration solutions. Such standards only contribute when incorporated into appropriate methodologies and tools. As appropriate standards evolve, they will be taken on board and become part of the platform of the successors to the AUTOMAIL system.

4. An industrial application

As a result of collaboration between the Loughborough Systems Integration Group and a major UK computer systems company, ICL, a decision was made by that company to transfer some of the AUTOMAIL flexible integration concepts and methodologies to its manufacturing environment. This development, the SEFIMA project (Support Environment for Flexible Integration of Manufacturing Applications), was planned as a two-stage implementation:

(i) Proof-of-concept demonstrator cell

(ii) Operational cell

The second stage would be dependent on successful performance of the first stage demonstrator. In addition to these two stages there was obviously a prior stage involving the specification and initial development of the SEFIMA software.

4.1 The SEFIMA system

The initial specification for SEFIMA includes mechanisms for the control of manufacturing application execution and the routing of inter-application messages. It does not provide the functionality of its AUTOMAIL ancestor in terms of information storage and retrieval mechanisms, but does provide for a limited functionality information service. The initial version of SEFIMA has now been produced.

Applications run either above the support environment (SEFIMA-conforming applications) or in separate machines (as alien applications). The SEFIMA manufacturing application interfaces control initiation, execution, information access and the means of application interaction. A database of currently executing applications is maintained. The host's inter-process communications service is used to pass the formatted command blocks which make up the SEFIMA/application interface mechanism.

The major aim of the communication services interface is, given a potential wide diversity in existing functionality and type, to bring each communication service up to a consistent level of functionality.

Communication with other SEFIMA systems involves the straightforward transference of data packets. Communication with alien machines involves additional interpretation steps by the SEFIMA system.

SEFIMA invokes and maintains data on communications service drivers and interface processes as it does on manufacturing applications.

4.2 Proof of concept demonstrator cell

The demonstrator represents the first "visible" stage in the transfer from the research environment to a working manufacturing support product for use by the company's manufacturing systems specifiers and implementors. The cell has been built, SEFIMA installed and the system is now being operated and evaluated within the company's Advanced Systems Development Department. The objectives in building this cell are:

(i) To demonstrate and promote an understanding of the main SEFIMA features.

(ii) To prove the internal software mechanisms and principles of operation.

(iii) To enable an assessment of the system's potential and applicability in the ICL manufacturing environment, including its impact on other aspects of the company's manufacturing systems integration policies.

(iv) To provide an indication of timescales and systems engineering effort for future implementations.

The cell consists of three Sun workstations, each with a SEFIMA implementation. There are three simple SEFIMA-conforming applications and one alien application (which represents a typical non-SEFIMA-conforming pre-existing application). The alien application runs on a Zevatech surface mount station simulator (provided by the Zevatech company) which is considered reliably to represent the Zevatech system reponses.

The SEFIMA demonstrator cell is now at the system test and evaluation stage. A range of graphic system representation tools has been produced to assist in the evaluation exercise.

4.3 Operational cell

Three main options currently under consideration for a live SEFIMA implementation concern (i) the integration of various system elements in the test shop (as depicted earlier in figure 4), (ii) to provide information support facilities in the surface mount assemply area and (iii) to underpin the release of engineering data, thereby integrating and enhancing existing issuing and archiving data systems in the company. The first of these options is very interesting from the point of view of requiring multiple SEFIMA instances arranged hierarchically, whereas the other options raise interesting issues with regard to potential three-schema information services and the AUTOMAIL/SEFIMA support of open access to heterogeneous distributed databases.

Ultimately, it is hoped that all three options will be implemented in an incremental, consistent fashion, leading to levels of sophistication and benefit not achievable via the use of hard integrated solutions.

5. Conclusions and future work

The soft integration approach is the most suitable way forward for small integration schemes. For larger, long term integration schemes, it is the only viable solution. The underlying methodology allows integration by "containable complexity", with a subsequent path to the integration of the resulting islands of integration into a total enterprise scheme.

Standards are an essential part of the foundations on which a common soft integration methodology is agreed and a set of conformant toolkits is built. At present there is a patchwork of standards, some developing "bottom-up", others "top-down". Some rationalisation of these standards will inevitably be required before wide progress is made on the general acceptance of soft integration methodologies.

The work of the Systems Integration Group based on AUTOMAIL will expand, both in terms of the incorporation of further emerging standards, and in terms of the addition of facilities to cope more effectively with further manufacturing-orientated systems such as engineering data release and production planning and control.

References

AZAR 88, Azar I and Weston R H, A Vision Reference Model for Systems Integration", Int. J. CIM. Vol. 1, No.4, 234-244.

ANSI 86, "ANSI American National Standard Database Language SQL" (New York: American Standards Institute Inc).

CIM-OSA 88, ESPRIT Project No. 688,CIM-OSA strategic management and design issues. CIM-OSA/AMICE, 489 Ave. Louise, B14-B-1050 Brussels.

EDIF 90, Electronic Data Interchange Format, Version 2.0.0. Obtainable from CADETC, Leeds Uni., UK.

EXP 89, Express Language Reference Manual, ISO TC184/SC4/WG1, Doc.No. N442, Dec, obtainable from CADETC, Leeds Uni., UK.

HOD 88, Hodgson A, Weston R H, Sumpter C M, Gascoigne J D, Rui A, "Planning and Control of Information Flow in CIM", IERE Int Conf. on Factory 2000, Cambridge, UK

JON 86, Jones A T and McLean C R, "A Proposed Hierarchical Control Model for Automated Manufacturing Systems. Journal of Manufacturing Systems, 19, 15-25

MAP/TOP 89, Manufacturing Automation Protocol/Technical Office Protocol Specification Version 3.0, Obtainable from the Society of Manufacturing Engineers, Dearborn, Mi 48121.

NOF 89, Nof S Y and Moodie C L, Editors, "Advanced Information Technology for Industrial Material Flow Systems", Springer-Verlag, Berlin.

STEP 89, STEP Preliminary Design Document & Introduction Document, ISO TC184/SC4/WG1, Obtainable from CADETC, Leeds Uni., UK.

WES 88, Weston R H, Gascoigne J D, Rui A, Hodgson A, Sumpter C M and Coutts I,"Steps Towards Information Integration in Manufacturing", Int J. CIM, Vol. 1, No.3, 140-153.

WES 89a, Weston R H, Gascoigne J D, Sumpter C M and Hodgson A, "Robot Integration within CIM", Int. J. Prod. Res.,Vol. 27, No 3, 515-528.

WES 89b, Weston R H, Hodgson A, Coutts I, Murgatroyd S and Gascoigne J D, "Integration Tools Based on OSI Networks", AUTOFACT Conf. Proc. ,SME, Dearborn, Mi 48121.

YOU 89, Yourdon E, Modern Structured Analysis, Prentice Hall Int., 1989.

SERVER NETWORKS:
A CIM ARCHITECTURE DESIGN ENVIRONMENT

L.E. ZEIDNER

Abstract

Computer-integrated manufacturing (CIM) offers the benefits of flexibility; however, many CIM systems are quite inflexible, due either to inflexible architectures or inflexible software implementations. This paper presents the Server Network Generator (SNG), a new high-productivity CIM architecture design environment. The SNG is a large-system design tool enabling the CIM architect to test the flexibility of his designs. It is also a distributed-software development environment enabling the CIM system implementor to create a flexible software implementation. The SNG provides application-level, graphically programmed, transparent access to a distributed VM/370-based computing platform networked across the CIM enterprise, ranging from IBM's new 7437 VM/SP Technical Workstation to the IBM 3090 vector-processing mainframe.

1. Introduction

> The "architecture" of a manufacturing system -- which includes its hardware, its material and information flows, the rules and procedures used to coordinate them, and the managerial philosophy that underlies them all -- largely determines the productivity of the people and assets in the factory, the quality of its products, and the responsiveness of the organization to customer needs [HAY88].

The goal of flexible manufacturing is to design flexibility into the architecture of a manufacturing enterprise. This goal is motivated by growing competition, shorter product cycles, increasing customer expectation, and market instability [FAR86] [MIT89]. The success of flexible computer integrated manufacturing (CIM) systems relies on their ability to change their product mix and their process technology fundamentally, often in previously unanticipated directions [GUP89]. Many early flexible CIM systems were only superficially flexible, offering limited pre-planned product variability. Some of these systems suffered from insufficiently flexible CIM architectures, while others relied upon inflexible software implementations [DUP82]. These early inflexible systems represent a costly software version of hard automation.

Changes in process technology, part design, materials, and required tolerances are necessary as products evolve. As the rate of product evolution continues to increase, these changes and others are increasingly likely to occur within the expected

lifetime of CIM systems. The CIM architect must be able to subject each of his designs to various forms of change, to evaluate its flexibility. The design and implementation of successful, flexible CIM systems will depend upon the availability of large-system design tools that enable the CIM architect to explore the advantages of alternate designs, through simulation, early in the design process.

1.1 Flexibility

Much of the opportunity for flexibility in automation stems from the use of computers and software systems to control manufacturing processes, replacing previous hard-automated control systems. If the changes made to software-controlled systems are less costly than similar changes made to hardware-controlled systems then flexibility is likely to be enhanced.

Unfortunately, the traditional manual programming technology used to implement most CIM system software is rather inflexible. It requires many programmers working for a considerable length of time. Recent studies show that software-system maintenance may constitute up to 75 percent of a system's cost over its lifetime [NRC90]. As the CIM enterprise grows in size and complexity, so do the software systems that control it. CIM systems built by these traditional manual programming techniques become increasingly inflexible as they grow in size and complexity, while the size of the programming staff grows to handle the task, becoming less likely to converge on a solution [BRO75].

Due to the promises of advanced functionality made possible through software control, CIM software systems have taken on an increased burden of complexity. It is often more costly to change them than to change hard-automated systems. For major system modifications there is no guarantee that there would be convergence to a successfully modified functioning CIM system. High-productivity software-development methods are needed to make CIM systems flexible, reduce the cost of changes, and guarantee the convergence of system modifications.

While the size and complexity of a CIM system depends largely upon the size and complexity of the CIM enterprise, its complexity is compounded by a variety of other factors. CIM systems consist of a collection of software distributed across a network of computing hardware. This computing network is typically spread throughout the enterprise. The distributed nature of CIM software systems is a major source of complexity and inflexibility. High-productivity distributed-software development methods are needed to create flexible distributed CIM software systems.

The diversity of computing hardware found throughout the CIM enterprise as corporate mainframes, departmental systems, personal computers, cell controllers and machine numerical

controllers adds significantly to the complexity of CIM systems. The increasing speed with which the network of computing hardware becomes obsolete, and must be replaced for the enterprise to remain competitive, dramatically emphasizes the need for flexibility [DEM84].

1.2 Server networks

This paper presents a large-system design tool that is intended specifically for the design of CIM architectures and the resulting CIM systems. This design tool is a new high-productivity distributed-software development environment. It dramatically simplifies the development of distributed cooperative CIM architectures and systems by separating two of the sources of their complexity, so that these sources can be addressed individually. It allows the complexity of the individual CIM architectural components to be isolated from the complexity of their interconnection.

Three software-development technologies are then combined for system development: a) graphical programming [ZEI88], b) automatic code generation [ZEI87A], and c) data-driven application-independent software [ZEI87B]. The result is a system modelling, simulation, and implementation environment. The Server Network Generator (SNG) is a prototype of this high-productivity distributed-software development environment [ZEI89].

The SNG drastically reduces the number of CIM system implementors necessary to develop a CIM system, by separating the macroscopic and microscopic sources of complexity and by providing a highly productive distributed-software development environment. By drastically reducing the size of the CIM system implementation team, the development effort becomes a manageable task that is likely to converge.

A server network is a set of asynchronous concurrent software processes distributed across a computing network and cooperating to model, simulate, or control a manufacturing system. The SNG is used to develop and modify server networks. The CIM architect can use the SNG to graphically configure a server network for his CIM system. He identifies the CIM system's major architectural components and their interfaces by sketching a functional block diagram, using the SNG. The SNG automatically constructs the operational framework of the server network from the block diagram. Then the CIM architect can address the complexity of each architectural component individually using the SNG's high-productivity distributed-software development environment.

The server network framework consists of automatically code-generated data that drives application-independent network communication software. This drive data can easily be changed by graphically modifying the architectural block diagram sketch in the SNG. As a result, the server network framework is changed automatically. This ease of flexibility enables the CIM architect to experiment freely with his designs, testing their flexibility.

1.3 Implementation

The SNG software-development environment is implemented using multiple VM's within the IBM VM/370 operating system. The use of the VM/370 operating system provides three major benefits. First, the computing environment in each virtual machine is quite rich, is fully configurable, and allows us to benefit from the past two decades of software-development progress in this arena. Second, modularity complete with assumption- and data-hiding is provided inherently because software within a VM is isolated from software in other VM's, except through intentional connections. Third, server networks operating within communicating VM's on a single host computer can be extended in a straightforward manner to server networks distributed across a network of VM/370 computer hardware, via a suitable interprocess communication platform.

Server networks are positioned well to make use of recent advances in desktop-mainframe workstation technology that have led to the introduction of IBM's new 7437 VM/SP Technical Workstation [IBM88]. This PS/2-based workstation provides a most desirable distributed-computing platform for flexible automation. It addresses a major source of complexity in CIM systems: the diversity of computing hardware and software. Personal computers (PC's) have been used to front-end a diverse spectrum of device controller hardware in many CIM systems, to reduce this apparent diversity. The 7437 workstation offers the next generation solution while providing the advantages of mainframe computation, communication and robustness. The 7437 workstation is a complete VM/370 mainframe that runs the same software as large VM/370 mainframe systems. Its communication is based upon the PS/2 component of the workstation. Through local area network (LAN) technology these workstations can become part of a distributed computing network of computer hardware ranging in power all the way from the 7437 up to the largest IBM 3090 vector-processing mainframes. The most important advantage of this approach to CIM system computing hardware, as compared with previous PC networks, is that the use of the VM/370 operating system offers transparent access to a broad range of distributed computing power coupled with robust powerful mainframe system software tools.

Server networks are intended to be distributed across this sort of mainframe computing network, with 7437 workstations interfacing both to manufacturing hardware and to people within the CIM enterprise. Each server is a software process that occupies a single VM in the IBM VM/370 operating system context. Many servers may coexist in a timesharing mode on each individual host computer, or a single server may run standalone on any of the networked hosts. This enables server networks to focus the full power of mainframe computing anywhere that it is necessary in the CIM system, at any time. The distribution of servers across the computing network can be changed dynamically because server network communication software is data-driven and application-independent. Furthermore, server network distribution across the computing network can be changed automatically to respond to changes in server computing needs, because server network configuration and communication drive data is automatically code-generated.

This paper describes a) the difficulties inherent in the development of distributed cooperative CIM software, b) how the SNG is used to design CIM architectures and implement CIM systems, c) the structure of server networks, d) a sample CIM system server network, e) implementation issues, and f) conclusions.

2. Distributed-software development

Distributed cooperative processing software systems are substantially more difficult to build than traditional single-threaded software elements. There are four primary contributing factors that complicate the distributed cooperative software-development process: 1) the logistics of distributed-software management, 2) the interdependencies between the cooperating processes, 3) their asynchronous operation and communication, and 4) "deadlocks" that must be prevented.

From a very practical perspective, the logistics of coordinating a distributed set of cooperative software processes pose problems of remote vs. local access to software, version conflicts, and completeness of software-change distribution. The SNG provides a mechanism for automating the software management process so that it becomes dependable.

Distributed cooperative software processes have interdependencies related more to the interpretation of their interface than to the actual data that they exchange. For example, consider two servers that communicate with an assumed application-level protocol. The implementation of the interface protocol is embodied in the software, on each side, that receives data and formulates responses. However, these two sets of software must be built apart and then tested together across the interface.

The asynchronous nature of the separate processing streams makes it impossible to predict precisely when an expected message will arrive from another process. When several messages are due from several processes it is impossible to predict the order of their arrival. Distributed software must be able to tolerate this asynchronous behavior. Repeated testing using identical test data is a technique that is central to traditional software development. It ensures that the only element of variability is the change made to the software between consecutive tests. It is impossible to provide the same level of consistency between consecutive tests in a distributed software-development environment due to the asynchronous behavior. While identical test data may be used, the order of asynchronous events is not predictable or controllable, and thus the processing paths taken, through the software, will be different each time it runs.

Situations can arise, called "deadlocks," in which two or more processors are stalled because they are each waiting for a message from one another. None of these waiting processors can break the deadlock, nor can any outside processor. These deadlocks must be prevented because they cannot be resolved remotely, once they occur. If communication status information is provided as a service of the interprocess communication platform, it can be used to guarantee that no deadlocks will ever occur. Each process never needs to enter a state in which it waits exclusively for a message from a subset of other processes.

3. Designing CIM architectures and creating CIM systems

To design a CIM architecture the CIM architect uses the SNG to sketch a system block diagram. In this subdivision of the CIM system the blocks correspond to major architectural components and the links correspond to component interfaces, which are specified parametrically. As a result, the SNG automatically builds the framework of a server network. A set of virtual machines (VM's) is organized, one per functional component, and the interprocess communication links are established between these servers to implement the interfaces between the components. The communication software within each server is automatically provided. All that is required of the CIM architect or CIM system implementor is that he develop the software, within each server, that models the architectural component's functionality. The SNG provides a highly productive distributed software-development environment within which to accomplish this task.

This functional decomposition of a CIM system into its architectural components and their interfaces is the basis for design with server networks. It is intentionally a "contextual" subdivision of the system. The CIM architect describes the system according to his understanding of its operation. He distinguishes between the complexity of his architectural design and the complexity of the architectural components. This design approach provides modularity so that the complexity of each

functional component is localized within that server and can often be changed with minimal impact on the rest of the CIM system. From a software-design perspective this design approach facilitates assumption- and data-hiding [PAR72]. The allotment of one virtual machine to each server provides a rich computational environment within which to model the associated functional component.

4. Server network structure

4.1 Interserver communication

The interprocess communication between servers is performed by application-independent data-driven software. This software is completely generic, i.e., independent of the details of any particular CIM system. The communication software implements the details of each individual CIM system by processing data in which the details are encoded. The SNG converts the information gleaned from the sketched block diagram into "drive data," by a process of automatic code generation (ACG). The generic communication software is actually "driven" by this drive data. This approach to software development is called the data-driven control-flow (DDCF) methodology [ZEI87B]. The DDCF methodology provides enormous flexibility in the modelling of CIM systems and extremely high productivity for CIM system development by combining application-independent software, built once for use in all server networks, with graphical programming and automatic code generation of drive data specific to each CIM system.

Interserver communication is based on the "shared-variable" model [LAT73], complete with optional data-arrival interrupts for the receiver, and data-use interrupts for the sender. Each shared variable is bilateral, connecting two "share partners." Any server can share any number of variables with any number of other servers.

The shared-variable model enables servers to enjoy the best features of message passing and those of remote procedure calling, without the usual implementation complexity [BIR84]. The software within each server is organized into response functions and the software that they in turn invoke. There is a response function associated with each shared variable that can cause an interrupt when data arrives through it. The application-independent communication software in the server, driven by the graphically programmed and automatically code-generated drive data, handles the interrupt. The interrupt causes the associated response function to be invoked, thus using the newly arrived shared-variable data. The data could be a message passed from another server or it could be thought of as the argument to a remote procedure call. In the remote procedure call model, the share partner has caused this server to invoke the response function and whatever software it invokes, possibly

returning a result or causing other actions in the server network.

Within the SNG, the CIM architect indicates the interfaces between servers. He uses "shared-variable macros," a highly productive hierarchical approach to specifying the hundreds or even thousands of shared variables interconnecting a complex CIM system server network. Each shared-variable macro is a bundle of shared variables and their associated interrupts. A very simple analogy to a shared-variable macro is an electrical cable consisting of many strands of wire. Beyond this simple analogy, macros group shared variables contextually in bundles organized for a specific purpose. Within the SNG, shared-variable macros are considered as attributes for the block diagram links. There is no limitation on the number of macros per link.

Shared-variable macros also facilitate both top-down CIM architectural design and a highly productive method for CIM system modification. CIM architectures can be designed macroscopically with the details of macro definitions deferred to the CIM system implementation stage. The architectural design structure is more easily understood due to this contextual grouping which allows a bi-level view of the component interfaces. During CIM system implementation or later, when an existing CIM system must be modified, when the need arises to change a macro's shared-variable makeup, the change is automatically replicated in every instance of the macro usage throughout the CIM system, during the automatic code generation phase.

4.2 The SNG and the dispatcher

The SNG is an expert system, a customized interactive graphical and parametric environment, with which the CIM architect and CIM system implementor can a) graphically configure the network while network-integrity rules are automatically enforced, b) monitor its performance, and c) perform remote distributed software development. The SNG helps them to conceptualize the operation of sets of communicating asynchronous CIM system components. It helps them to remotely analyze particular component/component software and timing interactions.

A special server called the "dispatcher" is used to establish and maintain contact with each VM under its control and coordinate network configuration. The dispatcher directs each VM to load software appropriate for its role as a server modelling an architectural component in the newly configured CIM system. It then distributes the drive data, built by the SNG, to each server, sending only the portion pertaining to that individual architectural component. The dispatcher is used for any subsequent reconfiguration of the CIM system, including minor changes to the system connectivity, or relocation of servers to different VM's.

Figure 1 shows the dispatching subnetwork. It consists of the SNG, the dispatcher, and all of the VM's controlled by the dispatcher. The CIM system subnetwork is created when the SNG and dispatcher send appropriate drive data through the dispatching subnetwork. The dispatching subnetwork is retained and used to reconfigure the CIM system subnetwork as necessary, either incrementally or completely. Figure 2 shows the relationship between the dispatching and CIM system subnetworks, for three tiny CIM system networks. Notice that although the dispatching subnetwork is a simple two-level tree, the CIM system subnetwork can take on any structure whatsoever. This flexibility is due to the DDCF approach to inter-process communication. It enables server networks to be used to model complex CIM systems, regardless of their structure. The communication software is no more complicated for a densely connected general network structure than it is for a simple ring, because the network structure is completely encoded in the drive data. The same generic data-driven communication software is used for every CIM system structure.

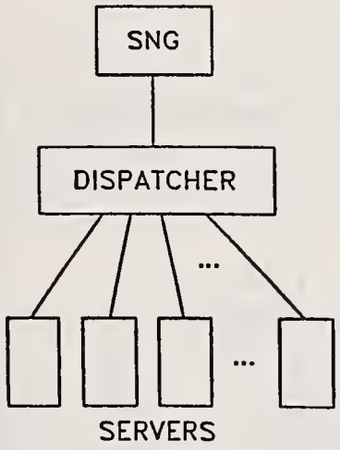


Figure 1. The dispatching subnetwork consists of the SNG, the dispatcher, and the VM's controlled by the dispatcher.

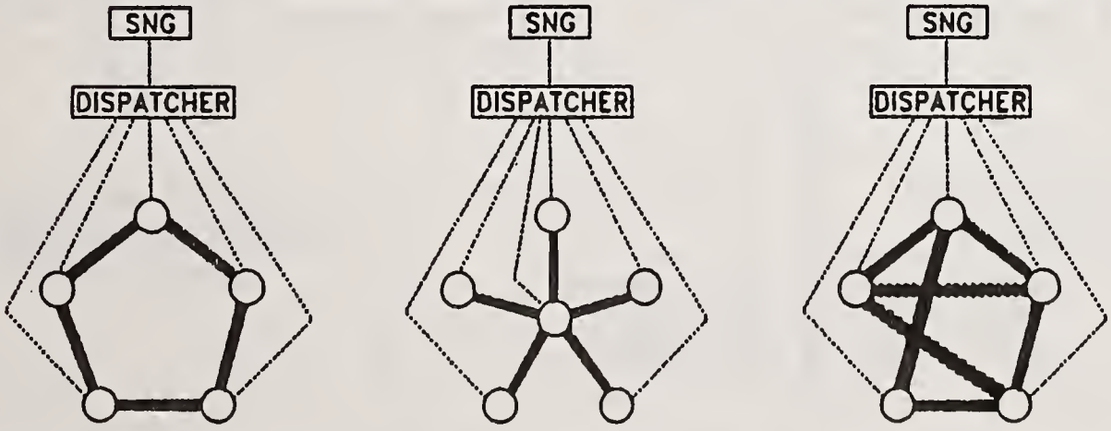


Figure 2. Three very simple CIM system subnetworks indicated using bold links, and the associated dispatching subnetworks, indicated using dotted links.

4.3 The software development process

Figure 3 illustrates the relationship between the SNG and the resulting CIM system server network software. The top of figure 3 portrays the process of using the SNG to capture the CIM architecture in the form of a block diagram, the interfaces between architectural components, and the distribution of the CIM system across the computing network. This information is gathered through graphical and parametric programming.

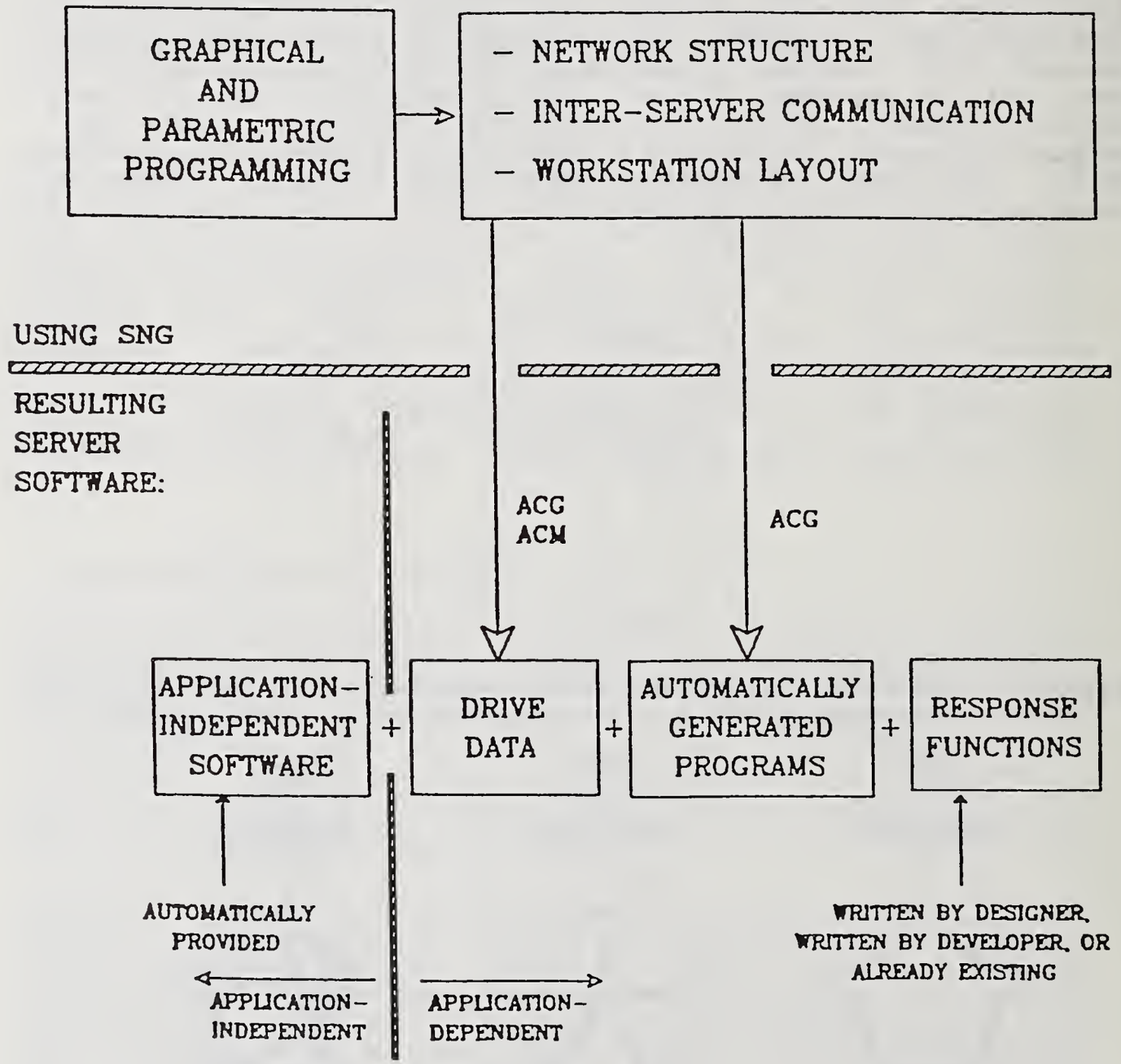


Figure 3: The structure and development of server network software.

The bottom half of figure 3 shows the resulting CIM system software and data. On the left is the generic application-independent data-driven communication software. This software is driven by the drive data which is built by the SNG through a process of automatic code generation (ACG), and is distributed by the dispatcher. The SNG also automatically generates some software, such as the initial versions of the response functions. These response functions are generated with only the bare essentials to perform inter-component communication. CIM software must be added to them to make them perform their architectural functionality. This addition of CIM software to the response functions is performed by the CIM architect or CIM system implementor, or the necessary software may already exist and merely need to be invoked. When changes to the CIM system structure, inter-component communication, or distribution layout are needed, the SNG is used to modify the server network. Then automatic code maintenance (ACM) is used to revise the drive data and the dispatcher distributes the changes.

4.4 Utility services

Most CIM system server networks include at least a few servers that do not correspond to components of the CIM architecture. These servers provide support services in the CIM system, for server network development, operations, validation, or maintenance. Typical support servers are clocks, input consoles, event loggers, status displays, data managers, and performance monitors.

Throughout this research we have attempted to isolate a minimal set of generic servers that could be reused in successive applications. Support servers have yielded the best results in this endeavor. The concept of "provision of" and "subscription to" utility services has emerged. A server can provide a utility service to any number of subscribers, and a server may subscribe to any number of utility services. Note that these are utility services and <u>not</u> utility servers. A server may provide a utility service in addition to modelling a functional component of the engineering system, or may provide more than one utility service. From an implementation perspective, this requires the server to have access to a set of utility service software; however, additional software may be added. Because of the SNG's graphical programming and automatic code generation, it is straightforward to identify a set of macros of shared variables that must connect the server to subscribers, and a set of response functions and other associated software all of which constitute the ability to provide the utility service. All of these macros and the associated software can be automatically provided once the CIM architect graphically specifies that a server subscribes to a utility service.

The concept of utility services differs from that of the "client/server" model [CRI88]. The client/server model is most often associated with print servers and file servers. The motivation of the client/server model is to provide access to a limited valuable resource to a variety of clients. Often the related concept of "location transparency" is a goal in the client/server model. The concept of utility services is different because it carries no such association with a resource, valuable or otherwise, nor with the location of the service provided. Utility services can be any type of computation that we commonly prefer to do remotely, rather than locally within a server. Often this preference is specifically due to the existence of an appropriate utility service provider within the network. Using remote utility service providers can simplify a server's software development considerably. The tradeoff in increased inter-server communication traffic, and resulting level of utility service provided must obviously be weighed. A typical alternative to the use of a remote utility service is the use of a library approach to distributed utility software.

The mechanisms of support services required in different CIM systems are fairly consistent, however, the details of their implementation requirements vary considerably. Our approach to this variability is to create parametrically tunable utility services, in which the implementation details are tunable from within the SNG, typically through the data-driven control-flow (DDCF) methodology of software development. The result of this tuning process is drive data that ultimately drives the generic utility service software within the provider.

4.5 Performance monitoring and reconfiguration

A particularly useful utility service is that of performance monitoring. A server can be constructed to monitor the event logs kept by a server providing the event-logging utility service. If servers post events when they have been expecting a response from their share partners and it is seriously overdue, these complaints can be found by the server monitoring the logs. When the performance monitor notices sufficient evidence of a bottleneck in the network an appropriate solution is to move that struggling server onto another virtual machine in the distributed computing network having more computer power. Another alternative may be to remove other busy servers from the bottlenecked VM's host computer, leaving it more computer power that it would no longer have to share.

Either of these alternatives can be accomplished by the server providing the performance monitoring service. It can take advantage of the data-driven nature of server network configuration. By roughly the same process of automatic code generation (ACG) that the SNG uses to build the network configuration drive data, this server can modify the drive data and pass it to the dispatcher for network reconfiguration.

Reconfiguration may be a multi-phase effort, since the state of the bottlenecked server must be captured as well as the states of its shared variables and their interrupts, and all of this must be recreated at the server's new location, before it can continue operation. This entire process can be performed automatically due to the DDCF software-development methodology.

5. A Sample CIM system server network

This server network is an example of how a CIM architecture can be designed. This CIM system was modelled using the methods outlined above, so that the system's behavior could be simulated. This sample server network simulates a manufacturing enterprise. Figure 4 shows the block diagram as it was sketched in the SNG.

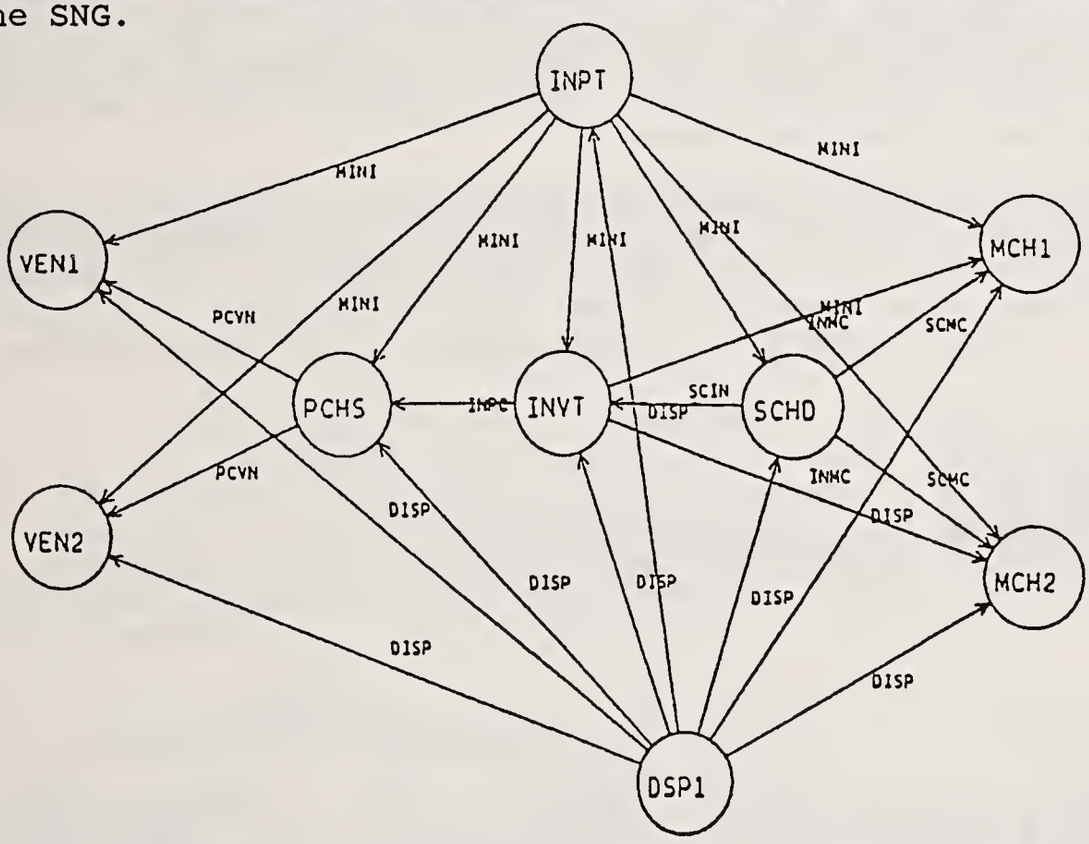


Figure 4: The block diagram sketched in the SNG corresponding to an existing CIM system server network.

Each circle represents an architectural component modelled by a server. These are numerical control machines (MCH1 & MCH2), the production scheduler (SCHD), the inventory control system (INVT), the purchasing department (PCHS), and the external vendors (VEN1 & VEN2). The dispatcher is shown below (DSP1) and a utility service of input console is provided by the server at the top of the block diagram (INPT). The input console is used to construct and archive simulation scenarios. The inventory control system (INVT) server also provides the graphical status display utility service. Each link is labelled with one or more shared-variable macros as link attributes.

Note that several of the macros are used repeatedly, replicating the associated shared-variable connections. Note also that both the machines (MCH1 & MCH2) and the vendors (VEN1 & VEN2) are sets of replicated servers. The server network software is able to function correctly, regardless of how many duplicates are sketched in the SNG. For example, the purchasing department server software is able to flexibly base its strategy on however many vendor servers it sees in the network. Similarly, the production scheduling server is able to flexibly accommodate as many NC machines as are present in the network, and dynamically adjust if new machines appear or disappear.

The shared-variable macros in this server network are PCVN, the vendor/purchasing department interface; INPC, the purchasing department/inventory control interface; SCIN, the inventory control/production scheduler interface; INMC, the inventory control/machine interface; SCMC, the production scheduler/machine interface; MINI, the simulation initialization data; and DISP, the dispatcher/server interface.

Figure 5 shows the graphical distribution of the CIM system server network over the computing network. The CIM system block diagram is shown in the lower right quadrant. The computing network nodes are shown in the other quadrants, divided by category. The assignment is quick and easy to do graphically.

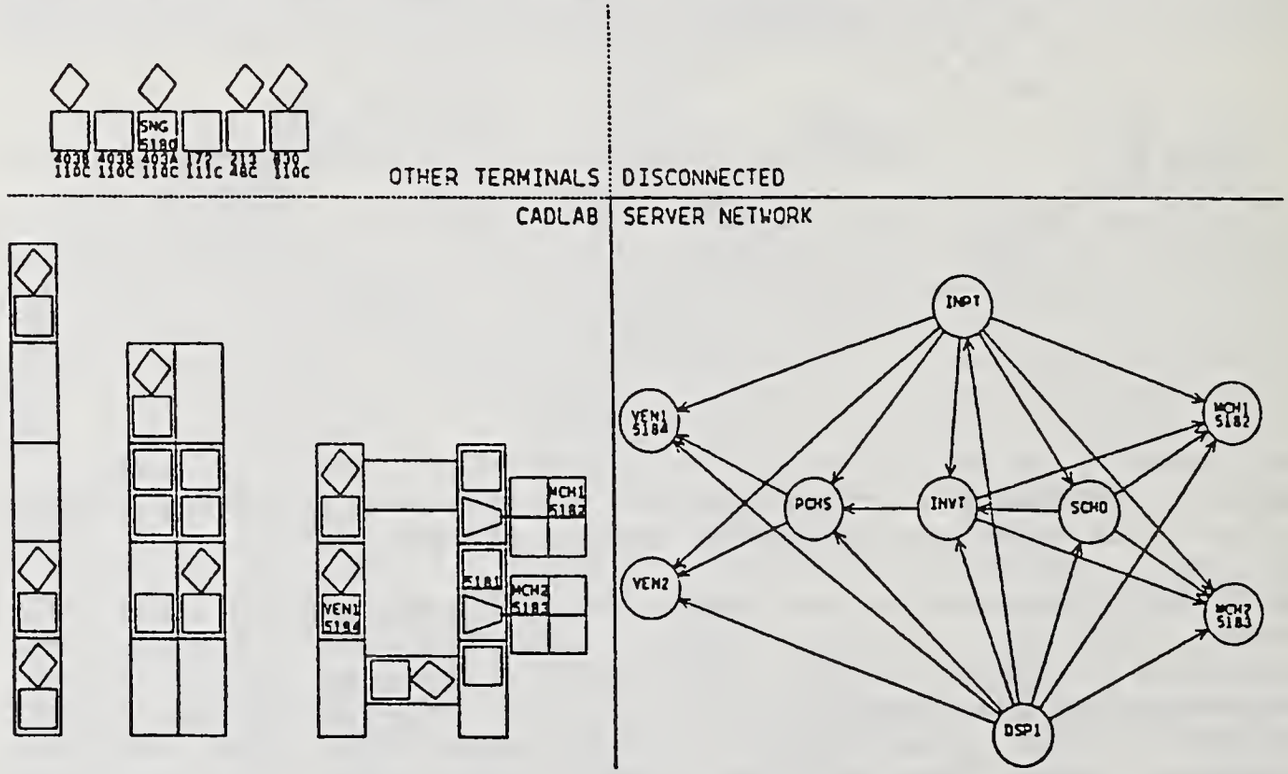


Figure 5: The graphical facility, within the SNG, used to distribute the server network across the computing network.

The CIM system exists once the network is dispatched. By using the input console service in the INPT server, a set of manufacturing jobs is specified, each consisting of assembling a set of parts. The quantity of each part type used in each job is specified. Then the initial stock levels are set, and the reorder quantities can be adjusted. A keystroke starts the CIM system by sending this information out through the MINI shared-variable macro to each server. The job scheduler (SCHD) examines each job's requirements and compares them with current inventory levels. When a job is ready for production it is sent to an available NC machine. If a stock level is depleted, inventory control (INVT) notifies the purchasing department (PCHS) to reorder. The purchasing department uses whichever vendor/purchaser interface policy is being tested, to order the part. Later, if the vendor is busy and has not yet filled the order, the purchasing department may retract the order, if the vendor has not yet begun processing it. Ultimately, another vendor may be approached. Many different policies could be implemented for comparison.

While the CIM system is operating, the inventory control system (INVT) server provides the status display service, displaying a constantly refreshing bar chart of each part's stock levels.

6. Implementation

The SNG software-development environment has been implemented using multiple virtual machines, within the VM/370 operating system, on an IBM 4341 small mainframe computer. It is currently being transferred onto a distributed network of IBM's new 7437 VM/SP Technical Workstations. This involves extending the interprocess communication platform to span a distributed computing network.

6.1 The interprocess communication platform

The interprocess communication platform used in this implementation is APL2's inter-user shared-variable facility [IBM87]. This platform provides reliable, rapid, flexible communication and is directly and productively accessible to the CIM architect and CIM system implementor. The availability of this platform greatly enables CIM system development because of the direct high-level language accessibility. It allows the direct sharing of CIM data without any need to invoke external function calls using this data as arguments to the functions, or to move the CIM data through file transfers.

We are currently extending server network technology to a distributed computing platform by extending the inter-user shared-variable platform to span across a distributed network of VM/370 host computers. This will provide the concurrent-processing speedup, flexible transparent access to different

strengths of computing power and levels of performance consistency, and physically distributed access to CIM resources such as people and numerical control (NC) machines spread across an organization or an automated factory floor.

The Computer Integrated Design Analysis and Manufacture (CIDAM) Laboratory at Boston University was recently awarded a research contract involving a testbed of IBM's new 7437 VM/SP Technical Workstations [IBM88]. These desktop mainframes connect to one another, and to our IBM 3090 and 4341 mainframes, via IBM's token ring local-area network (LAN) and various controllers. When connected in this way they communicate with one another at high speed using traditional mainframe-to-mainframe channel-to-channel communication. They are VM/370 machines in every way and run all of the software we currently use on our 4341 including the SNG.

We are currently transferring server network technology onto this research testbed of 7437's and connecting them to the 4341 and 3090 mainframes. The 3090 is equipped with vector-processing capability. (This capability is employed automatically by the APL2 interpreter running on the 3090 without any need for special attention by the software developer. Any APL2 software that can benefit from vector processing is automatically performed as vector operations, while computations on smaller arrays are automatically performed as scalar operations.) To move server network technology into the distributed computing arena we are extending the inter-user shared-variable platform to provide communication among distributed VM/370 processes, based upon the VM Pass-Through facility [IBM85].

6.2 Inter-server communication protocols

As a result of the experience we have gained while building several server networks to model and simulate engineering systems, we have observed certain usage patterns. We are in the process of identifying a minimal set of shared-variable macro structures and associated sets of response functions that accomplish application-level communication protocols that are used frequently. For example, maintaining a queue between two servers, broadcasting data to a set of servers, and collecting data from a set of servers and acting only when all has arrived, are typical protocols that are used frequently. We are incorporating this minimal set of protocols into the SNG so that they can be indicated graphically to describe the interfaces between the functional components in the system block diagram. Using data-driven automatic code generation, it is straightforward to establish these protocols between the appropriate servers. This will further reduce the quantity of software that the CIM architect or CIM system implementor must develop textually by traditional software-development methods. The response functions responsible for effecting these protocols are general-purpose application-independent software customized through application-dependent drive data.

7. Conclusions

Server network research has progressed to the point that a prototype SNG software-development environment exists, and has been used to develop several server networks for modelling and simulating CIM systems as well as other types of engineering systems. This work has demonstrated the validity of the underlying data-driven distributed software-development concepts. With the availability of our new Research Testbed we have an opportunity to move this research into the distributed cooperative processing domain. Transferring server network technology onto the Research Testbed serves two purposes. First, it provides a truly distributed computing environment for CIM architecture design and system implementation. Second, it creates the first high-productivity, distributed-software development environment featuring application-level, graphically programmed, transparent access to a complete dynamic range of computing power from the IBM 7437 all the way up to the IBM 3090.

8. References

[BIR84] Birrell, A.D., and Nelson, B.J., "Implementing Remote Procedure Calls," ACM Transactions on Computer Systems, vol. 2, no. 1, pp. 35-39, 1984.

[BRO75] Brooks, F.P. Jr., The Mythical Man-Month, Addison Wesley, 1974.

[CRI88] Crichlow, J.M., An Introduction to Distributed and Parallel Computing, Prentice Hall International, Hartfordshire UK, pp. 112-135, 1988.

[DEM84] Dempsey, P., "Why designers of FMS ought to consider integration," The FMS Magazine, April 1984.

[DUP82] Dupont-Gatelmand C., "A Survey of Flexible Manufacturing Systems," Journal of Manufacturing Systems, vol. 1, no. 2, pp. 1-16, 1982.

[FAR86] Farnum, G.T., "FMS: The Global Perspective," Manufacturing Engineering, pp. 59-60, 1986.

[GUP89] Gupta, D. and Buzacott, J.A., "A Framework for Understanding Flexibility of Manufacturing Systems," Journal of Manufacturing Systems, vol. 8, no. 2, pp. 89-97, 1989.

[HAY88] Hayes, R.H., Wheelwright, SC., and Clark, K.B., Dynamic Manufacturing: Creating the Learning Organization, The Free Press, NY, pp. 185-191, 1988.

[IBM85] IBM, VM/Pass-Through Facility: Guide and Reference, SC24-5208, 1985.

[IBM87] IBM, APL2 Programming: System Services Reference, SH20-9218-2, 1987.

[IBM88] IBM, 7437 VM/SP Technical Workstation: User's Guide and Reference, SA23-0351-00, 1988.

[LAT73] Lathwell, R.H., "System Formulation and Shared Variables," IBM Journal of Research and Development, vol. 17, no. 4, 1973.

[MIT89] The MIT Commission on Industrial Productivity, Made in America: Regaining the Productive Edge, MIT Press, Cambridge MA, 1989.

[NRC90] The National Research Council; Computer Science and Technology Board, Scaling Up: A Research Agenda for Software Engineering, National Academy Press, 1990. (Excerpted in Communications of the ACM, vol. 33, no. 3, pp. 281-293, 1990.)

[PAR72] Parnas, D.L., "On the Criteria to be Used in Decomposing Systems into Modules," Communications of the ACM, vol. 15, no. 12, pp. 1053-1058, 1972.

[ZEI87A] Zeidner, L.E., "Server Networks for the Design and Analysis of Factory Communication Systems," Proceedings of the Workshop on Factory Communication, IEEE/NBS, NBSIR 87-3516, pp. 171-177, 1987.

[ZEI87B] Zeidner, L.E., et al., "An Expert-System Generator," IASTED Journal of Control and Computers, vol. 15, no. 1, pp. 22-33, 1987.

[ZEI88] Zeidner, L.E., "Server Networks: Software Integration Tools for CIM," Proceedings of the International Conference on Computer Integrated Manufacturing (CIMIC), IEEE Computer Society Press, Washington, D.C., pp. 226-235, 1988.

[ZEI89] Zeidner, L.E., "Server Networks: Distributed Simulation and Modelling," Proceedings of the IASTED International Symposium APPLIED SIMULATION AND MODELLING-ASM'89, IASTED, pp. 138-142, 1989.

9. Acknowledgements

This research was funded during the past several years in part by the IBM Corporation, by the General Electric Corporation, and by National Science Foundation grant #DMC-8615560. The CIDAM Laboratory's new Research Testbed of IBM 7437's was provided through IBM contract #MHVK-UO51. The development of server networks enjoys continual support and encouragement from Yehonathan Hazony, CIDAM Director. Many of the concepts

expressed in this paper grew out of discussions and implementations in my graduate course on server networks. The sample CIM system server network was designed and built by Junjie Hu in that course.

RIA: REFERENCE MODEL FOR INDUSTRIAL AUTOMATION

MICHEL BÖHMS, FRITS TOLMAN

TNO, The Netherlands

ABSTRACT

It is clear that we need CIM. It is also clear that everybody involved in CIM uses his own definitions, concepts, models and terminology. Therefore we have to take one step back and try to model the CIM modeling itself by the development of a CIM Framework which identifies and relates existing results of CIM projects/activities and gives guide-lines for doing things better in the future. Two major models within the CIM Framework are worked out: a CIM Reference Architecture, a blueprint for applying CIM in discrete parts manufacturing and a CIM Base Model, the language used to express the CIM Reference Architecture.

1 Introduction to this paper

In this paper we present a CIM Framework which is used for two main purposes:
- to understand and interrelate existing projects/activities which deal with modeling CIM and,
- to propose a new way of modeling CIM

We also develop two main elements within the CIM Framework:
- A CIM Reference Architecture and,
- A CIM Base Model used to express the Reference Architecture

We will concentrate on the manufacturing of discrete parts although some results, especially the framework itself, are also valid for a wider scope. This paper is mainly based the earlier papers [BOH89] and [TOL89] and on discussions about CIM in three groups:
- ESPRIT II IMPPACT project
- ISO TC184/SC5/WG1 'reference models' and,
- CEN/CENELEC AMT WG-ARC

We hereby want to thank the people in these groups who indirectly contributed to this paper. I hope that this and many other CIMCON papers will be the start of interesting discussions about the new work item in ISO TC184/SC5/WG1 on a Framework for Modeling CIM.

2 CIM

2.1 Why CIM?

Through a bottom-up approach in industrial automation, we see islands of automation everywhere. Typical islands in industrial enterprises are the 'product modeling' island, the 'process modeling' island and the 'production' island. The first island is typically dealing with fast specialized hardware (workstations) for the manipulation of geometrical information of parts of products. The second island is concerned with more general, less capacity critical hardware (minicomputers, mainframes and personal computers) manipulating bills of materials, planning data, commercial data etc. The third island is dealing with specialized computer controlled machines (e.g. punching, cutting, bending, milling and drilling machines) and things like automatic transportation -, storage - and positioning systems (e.g. robot systems). Integration of these islands of automations means in the first place a global optimization of the business functions involved, instead of a sub-optimization for every island.

The relation between the different functions in the enterprise is shown in figure 1-1. This figure divides the industrial enterprise in two parts (see also [SOL88]): a Real System (RS) and an Information System (IS). The real system covers the realization of the product (it is the 'shop floor' or 'factory'), the information system takes care of the planning and control of the real system.

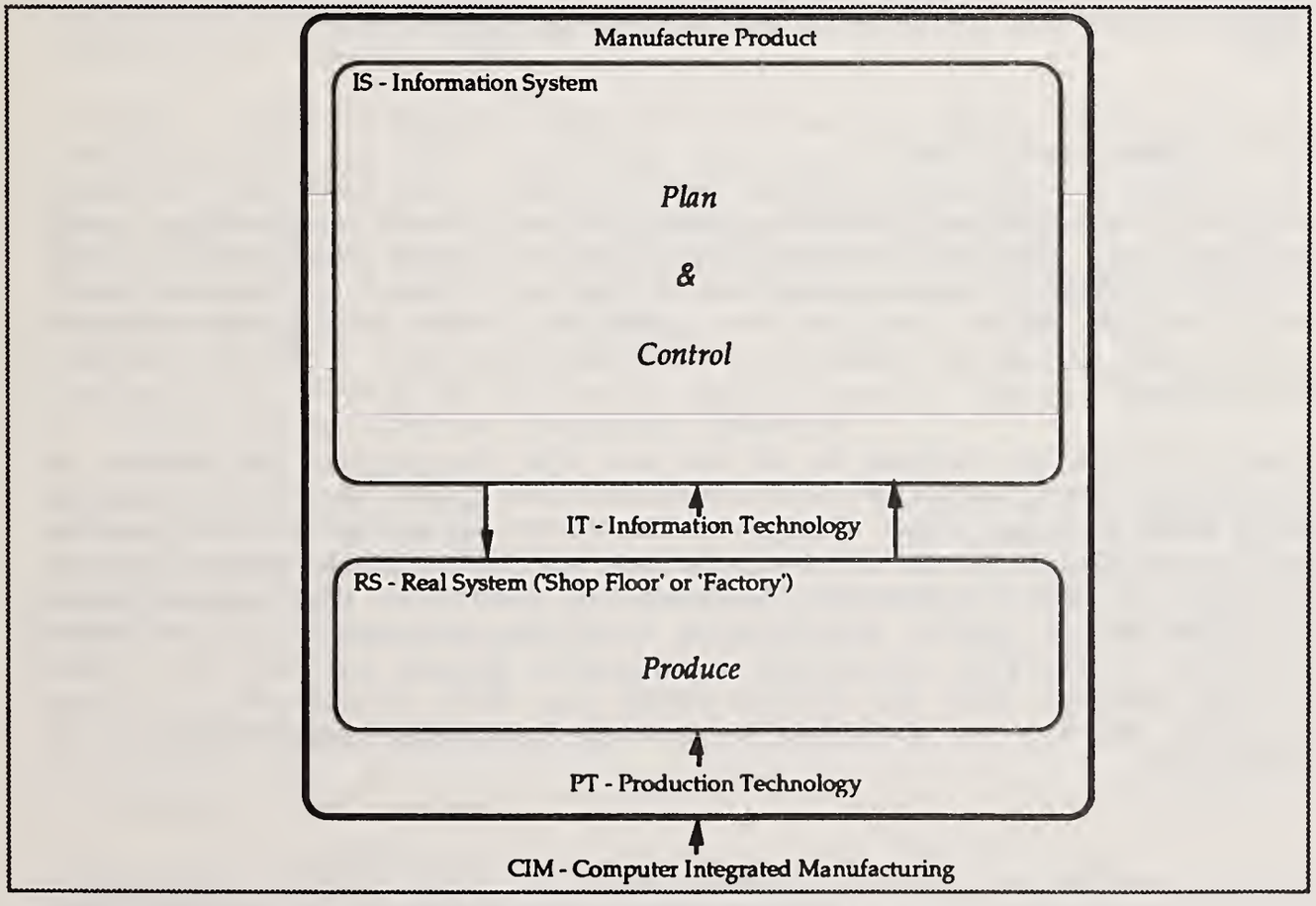


Figure 1-1. Industrial Enterprise seen as a RS-IS Combination

CIM in general can then be defined as:

The application of information technology components and production technology component in such a way that these components are supporting the enterprise functions in an effective and efficient way with a high degree of automation.

'Information Technology Components' here means: all the software based mechanisms performing or supporting information processing functions in the 'information system' of the industrial enterprise. Concrete examples are product modeling software (CAD), process planning software (CAPP, MRP) and production control software.

'Production Technology Components' here means: all the machines performing or supporting the realization functions in the 'real system' (the shop floor or factory) of an industrial enterprise. Concrete examples are NC controlled milling -, punching -, cutting, bending and measuring machines which are controlled by the information system.

The enterprise functions of interest are the functions that deal with the complete product life cycle from the product specification, via the part modeling (design), the process modeling (planning) and the time scheduling to the production control and the actual realization (production) of the product.

Future trends in Industrial Enterprises like Co-Makership, Just-in-Time Management, Design for Economic Manufacturing will only increase the need for CIM.

2.2 Problems with CIM

Hundreds of committees, platforms, standardization institutes and working groups are developing CIM models, standards and concepts dealing with CIM for the description of CIM Components and the application of them in industry. Almost every research project uses its own vocabulary: What is e.g. the difference between CIM (Computer Integrated Manufacturing) and CIE (Computer Integrated Enterprise)?

Many companies are confused by all the new CIM Components, CIM Models and organization principles (figure 1-2). This and the typical bottom up approach (there is often no top down strategy), is the reason for the islands of automation. Partial processes in enterprises are optimized but the partial solutions cannot be integrated. 'Design for Economic Manufacturing' cannot be implemented because it's impossible to transfer manufacturing knowledge automatically to the design department. Just-In-Time management is impossible because we don't use a standard for automatic order and delivery control (e.g. EDI). Co-makership is difficult to realize because we don't use a standard for product data exchange (e.g. PDES/STEP).

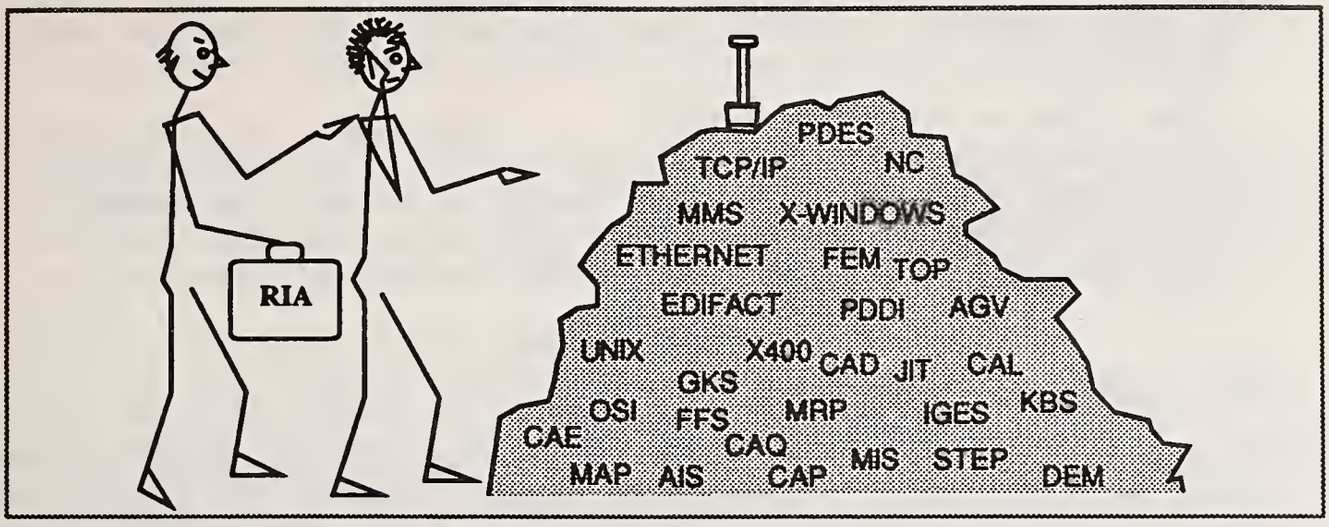


Figure 1-2. Concepts in CIM world

In short: the potentials of the existing CIM Components are not utilized because of a lack of integration.

Application of CIM in industry needs support in the form of a CIM Framework and CIM Reference Architectures in which business functions Information structures, CIM Components and their interrelations are described in a clear, open (vendor independent) and modular way, to provide a migration path for an industrial enterprise to an integrated CIM System. This paper will present such a Framework for modeling CIM.

Looking at results of projects dealing with CIM one can identify all kind of models that have been developed. Analysing these models we first develop a CIM Framework that helps us to understand the different approaches, to relate the different results and to identify overlaps and gaps. This Framework will be a collection of dimensions where each dimension represents a certain model property. This Framework is descriptive because it can be seen as a conceptualization of all kind of existing ideas about modeling CIM.

The results of the projects can then be analyzed with the help of this framework by considering the different dimensions identified.

Based on these analysis results and some new ideas (hypotheses) we come up with a new prescriptive telling what models to consider when modeling CIM in the future. The dimensions themselves will be the same but the points relevant for the dimensions will differ from the descriptive framework where sets of points are given for existing projects.

3 A descriptive framework

3.1 Existing CIM activities

Projects and activities which have been studied to identify the different modeling dimensions for the descriptive CIM Framework are:

- ESPRIT I CIM-Open Systems Architecture
- Esprit I Open CAM Systems

- ISO Open Distributed Processing
- Esprit I CAD-I
- NIST - AMRF
- CAM-I ATPC, An Architecture of CIM
- Danish Principal model for CIM
- ISO TC184/SC5/WG1 'Reference Models'
- ISO TC184/SC4/WG1 'STEP'
- CEN/CENELEC AMT WG-ARC
- Esprit II IMPPACT

3.2 Dimensions identified

We identified nine different dimensions for modeling CIM: two basic dimensions which are mostly implicit available in the projects identified, the modeling level dimension and the language level dimension and seven other dimensions which can be seen as an elaboration of the CIM Framework:

Modeling Level Dimension

This dimension distinguishes between different meta levels of modeling. Meta levels are: Reality, Models of Reality (what most project are dealing with) and models of models or Frameworks. Depending on the domain we can have all kinds of frameworks. Some existing frameworks are e.g.:

ISO-OSI (Open Systems Interconnections) for computer networks
ANSI-SPARC Framework for databases
OSF for operating systems

These are more or less well-defined and well-structured frameworks which are or have the potential to become world-wide international standards. Many frameworks however are just described in natural language or even hidden (implicit) in lower modeling levels. In this paper we will of course address a Framework for Modeling CIM.

Language Level Dimension

For every model on every level we need (a) language(s) to express them. Such a language is often referred to as the 'set of modeling constructs' or 'base model'. Just as with the first main abstraction mechanism, the modeling level, we have in general more language levels because we need also languages to express languages.

Aspect Dimension

Every projects models its own set of aspects (views, viewpoints) which they think are relevant for modeling CIM. Examples are 'functions', 'information', 'resources', 'responsibilities', 'management structure' etc.

Composition Level Dimension

We see different levels of composition of models used from very global models to very detailed models.

Scope Dimension

Different models have different ranges of applicability. Some are applicable to every industrial enterprise, some only for discrete parts manufacturing, others

are even more restricted to some special class of manufacturers and in the extreme case, a model is applicable for one specific manufacturer only.

Representation Dimension

Models might share the same content but this content may be represented in different ways (because a different language is used to express the model). Different representations are often necessary to suit the needs of different uses of the model (e.g. human understanding or computer interpretation).

Product Life Cycle Stage Dimension

We see often different sub-models for different stages in the complete product life cycle. E.g. information models for process modeling or function models for the realization of the product. In fact this dimension can be seen as as a fixed first (global) decomposition for a 'manuafcturing function' aspect. So the *first* decomposition of the manufacturing function is not orthogonal with this dimension, however when starting from the identified product life cycle stages the decomposition dimension is orthogonal for the manufacturing function aspect and thus also for the aspect dimension in general.

Actuality Dimension

Some model describe existing situations (AS-IS) while other prescribe a future situation (TO-BE) which does not yet exist. In the extreme they will describe an ideal far future solution which will never exist. We are now talking in a sense about a CIM model life cycle, modeling the CIM evolution in reality.

Specification Level Dimension

This dimension has to do with the amount of choices which are still open in a model. Some models are very generic (e.g. parametrized) while others are completely fixed. Note that this dimension is still orthogonal with the 'scope' dimension.

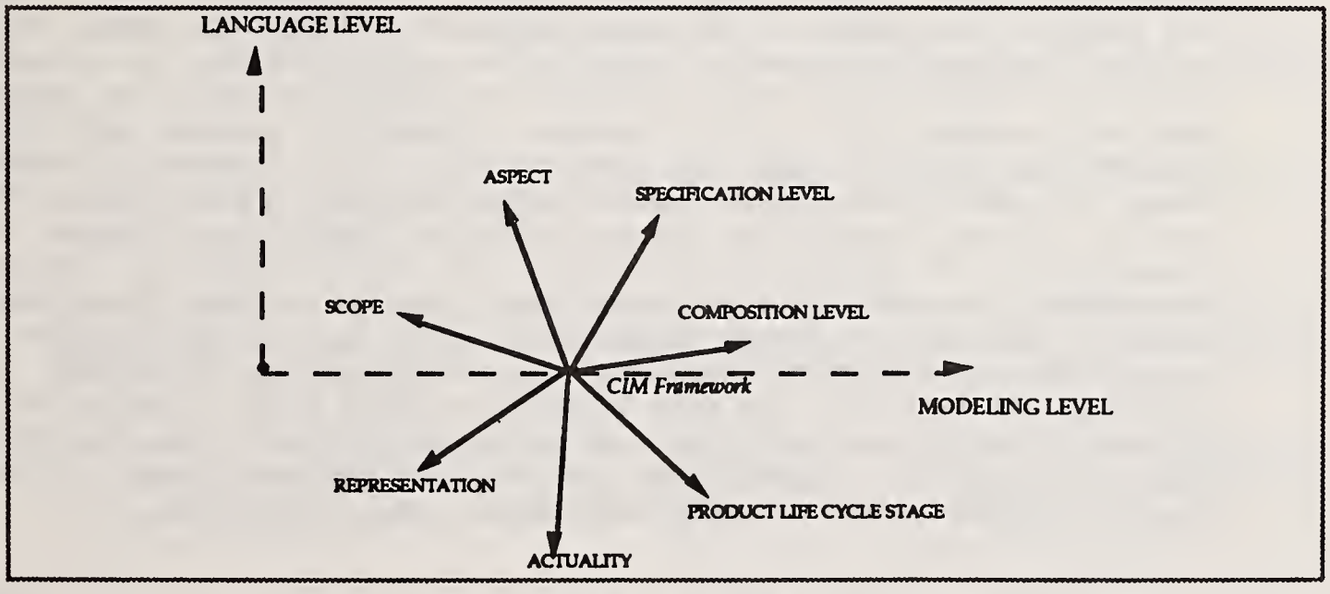


Figure 3-1. Framework for Modeling CIM

Summarized we have a nine-dimensional Framework, containing two very basic dimensions which encompass the scope of the CIM Framework itself and seven

dimensions within the CIM Framework (figure 3-1). Note that in principle all nine dimensions are othogonal.

3.3 Two examples: CAM-I CIM ARCHITECTURE and ESPRIT I CIM-OSA

In this section we will apply the descriptive framework to two specific projects dealing with modeling CIM and see how they fit in.

CAM-I CIM ARCHITECTURE ([CAM88])

The *Modeling level* for this project is the level of models of CIM. A framework is not made explicit. For the *language level* only one level is considered mostly in the form of human-understandable graphics and sometimes more formalized languages such as IDEF0 (SADT) or variants. There are no alternative languages used to express the same content so every model is *represented* only once. The *scope* for the models addressed is 'discrete parts manufacturing' and every *product life cycle stage* is covered. The *actuality* of the models is typically TO-BE (prescriptive). Five important *aspects* (called views here) are identified: management structure, information structure, function/activity structure, computer systems structure and physical structure. For none of the aspects we see models for different *composition* levels. Models for all views are elaborated and related to each other. Finally the models considered are always fixed and are therefore fully *specified*.

ESPRIT I CIM-OSA ([AMI89])

This project is typically a 'framework' project. The *modeling level* is therefore mainly the first meta level or framework level, identifying and relating CIM models. Also one *language level* is addressed, although this dimension is combined with the *scope* dimension. This combined dimension is called 'architectural levels' which range from 'generic' (first language level), partial and particular (two points on the scope dimension). Although not made explicit in their framework the complete *product life cycle* is covered. The *actuality* dimension is fairly covered by the CIM-OSA modeling levels (not to be confused with our modeling level). This dimension differentiates between requirements for CIM (enterprise modeling), best solutions for TO-BE (intermediate modeling level), and the actually chosen solution (implementation modeling level). Four *aspects* (called views) are selected: function, information, resource and organisation. It is stressed by the project that there is only one computer-interpretable *representation* for every model that is developed. Some models considered may contain choices (modelled by rules), so also different levels of *specification* are allowed (e.g. sort of parametrized function models with decisions). For most views and most CIM-OSA modeling levels (actuality) we see different *composition* levels (levels of detail) identified. E.g. for the function aspect we start on a global level with business processes going via more detailed enterprise activities to the most detailed level of functions.

The results of the analysis of the two projects are graphically shown in figure 3-2.

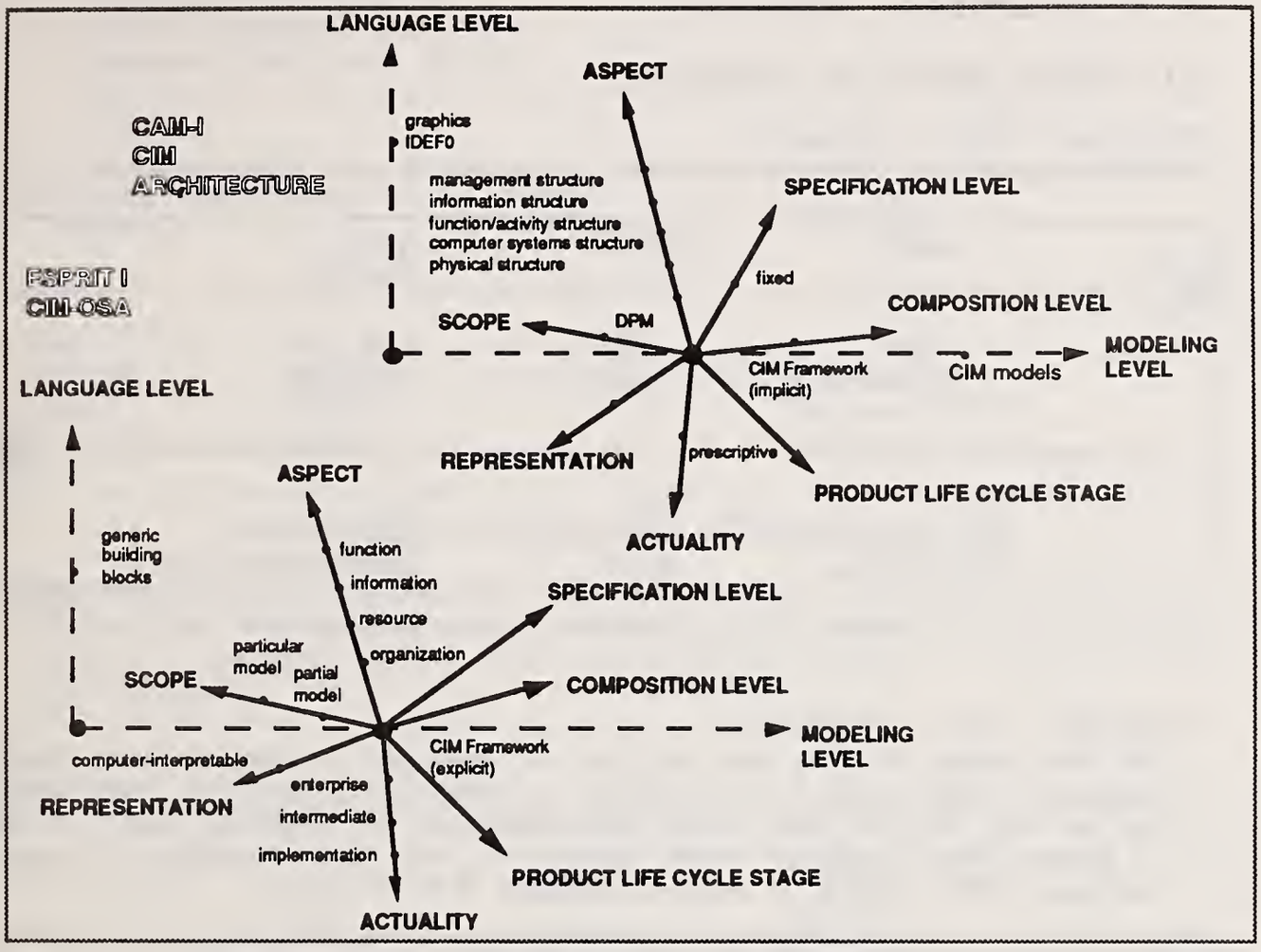


Figure 3-2. The descriptive CIM Framework applied

As shown the descriptive framework helps us to understand, compare and inter-relate the results of different CIM projects. The differences can now be handled and discussed on the right level:

- Which dimension are covered by the one and not by the other (implicitly or explicitly)? E.g. 'composition level' is not covered by CAM-I's CIM Architecture, 'multiple representation' is not covered by CIM-OSA.

- If both cover the same dimension, which points for the dimensions are covered by the one and not by the other? E.g. both projects cover the aspect dimension (they even call them both 'view') but they have different sets of aspects

- How are the same points for the same dimension worked out? E.g. both projects cover the first language level for the function aspect, but CIM-OSA uses a self-developed language while CAM-I uses $IDEF_0$.

4 A prescriptive framework

4.1 Chosen values for dimensions

Modeling Level Dimension

The series of meta-levels in modeling is depicted in the following figure:

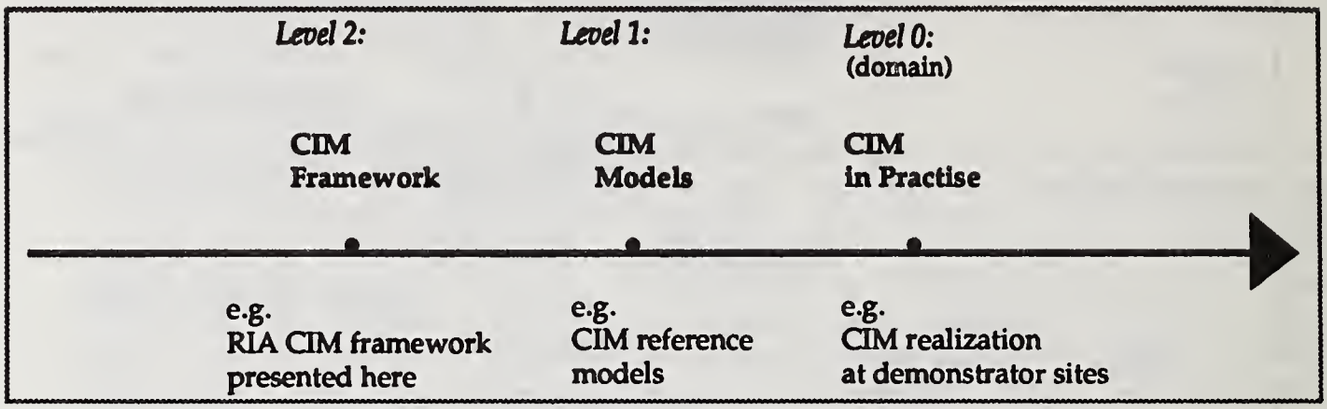


Figure 4-1. Modeling Level Dimension

Language Level Dimension

For every model on every level we need (a) language(s) to express them. Such a language is often referred to as the 'set of modeling constructs' or 'base model'. Just as with the first main abstraction mechanism, the modeling level, we have in general more language levels because we need also languages to express languages. This situation is depicted in figure 3-2.

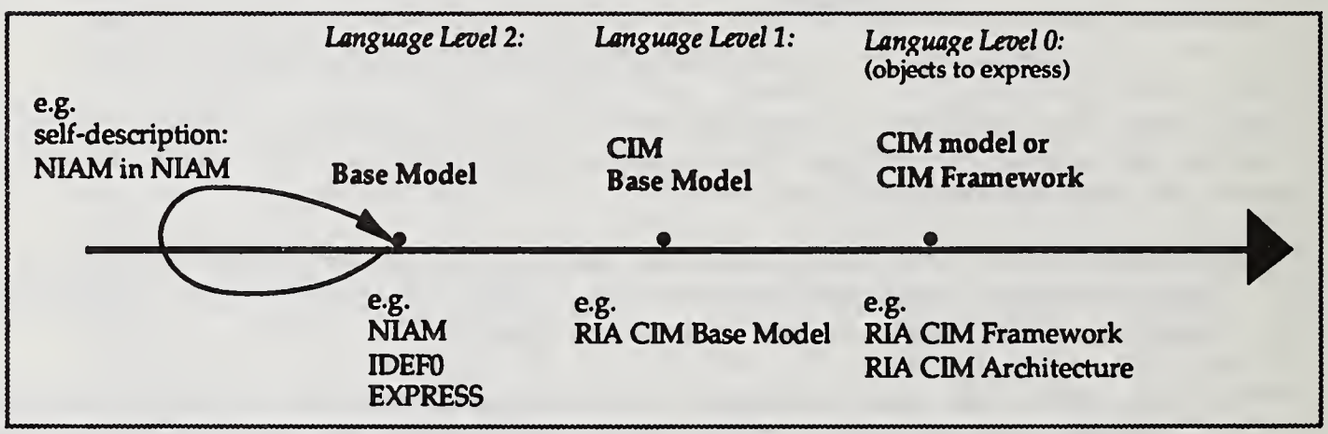


Figure 4-2. Language Level Dimension

Aspect Dimension

When modeling CIM we have to solve different sub-problems:

- a Systological problem: why CIM ?
- a Infological problem: what CIM ?
- a Datalogical problem: how CIM ?
- a Technological problem: with what CIM ?

We therefore have to model these four aspects when modeling CIM, resulting in four kinds of models:

- Manufacturing functions (why CIM?)

- Information (what CIM ?)
- Data & Programmes (how CIM?)
- Technology (with what CIM?)
  (Information Technology Components and Production Technology Components)

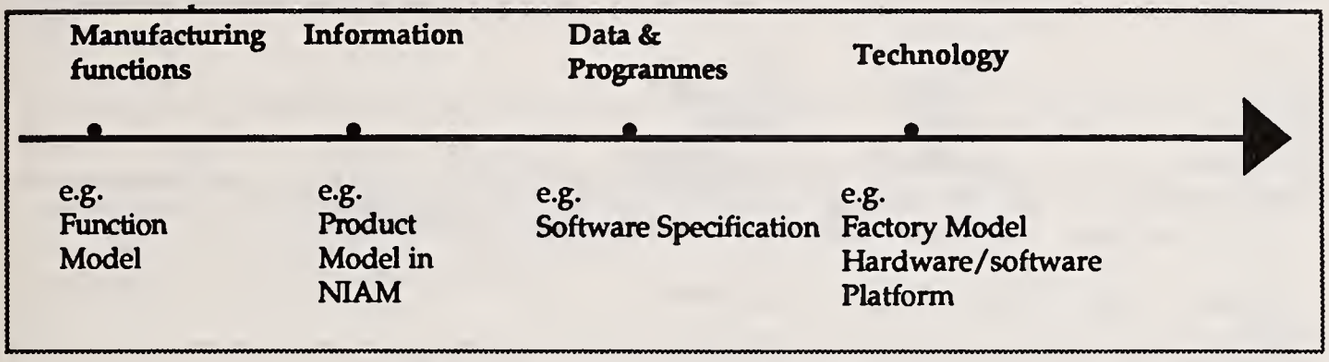


Figure 4-3. Aspect Dimension

Composition Level Dimension

We will not have fixed points for this dimension because the dimension can be seen as a continuum starting from *global* models and going to more and more *detailed* models.

Context Dimension

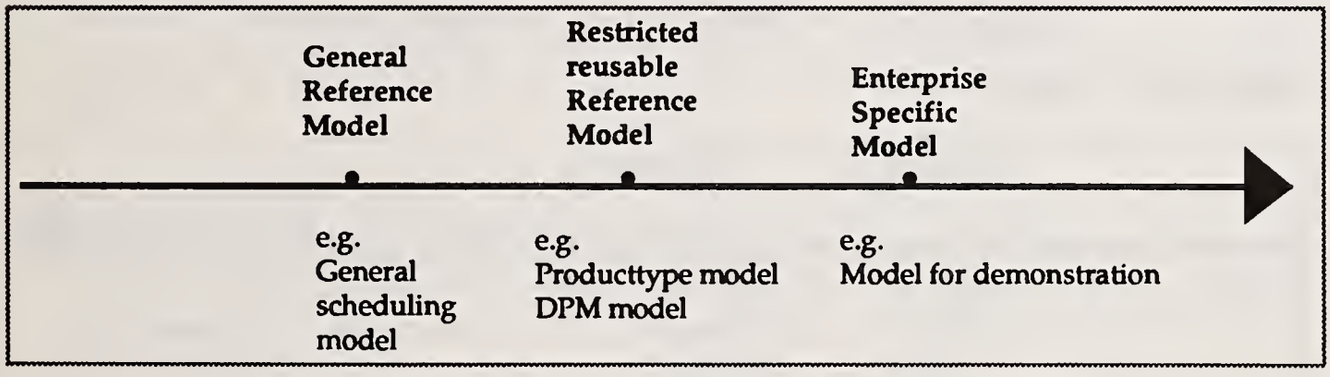


Figure 4-4. Context Dimension

Representation Dimension

Here we will distinguish between models which representation is only human-interpretable or also computer-interpretable. In the first case models are often intended to be able to understand the thing what is modeled. In the second case models are used for the future information processing. The representation of a model will change when a different language is used to express it. So the interpretability of the model depends of the interpretability of the language used. *Conversion* plays an important part in this language/model translation.

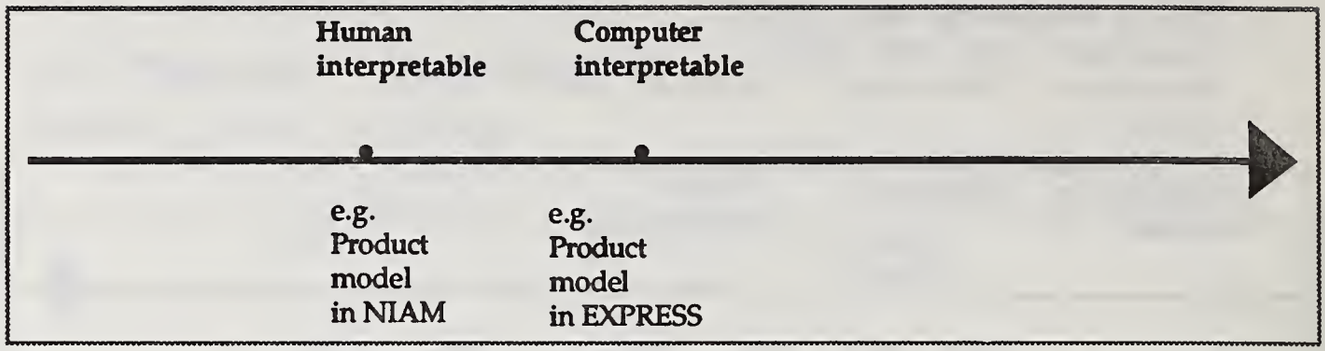


Figure 4-5. Representation Dimension

Product Life Cycle Stage Dimension

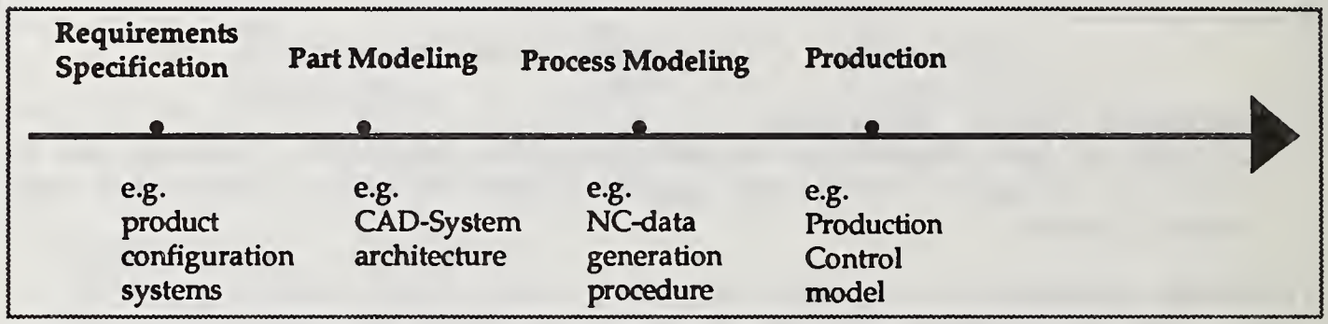


Figure 4-6. Product Life Cycle Stage Dimension

Actuality Dimension

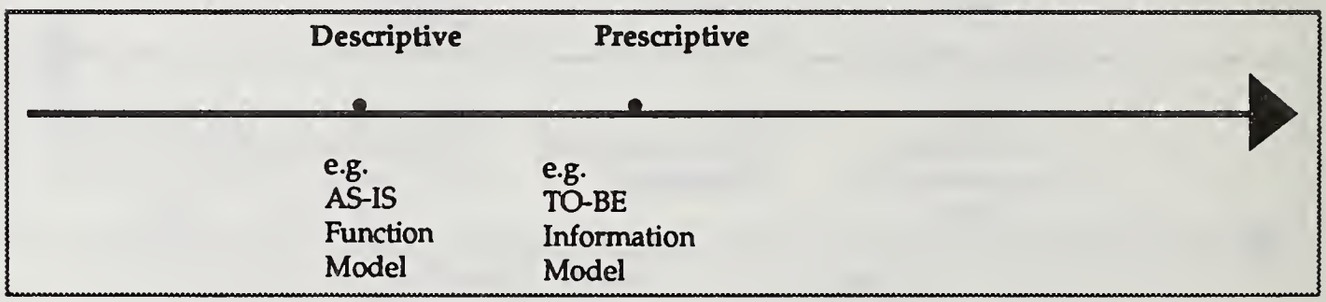


Figure 4-7. Actuality Dimension

Specification Level Dimension

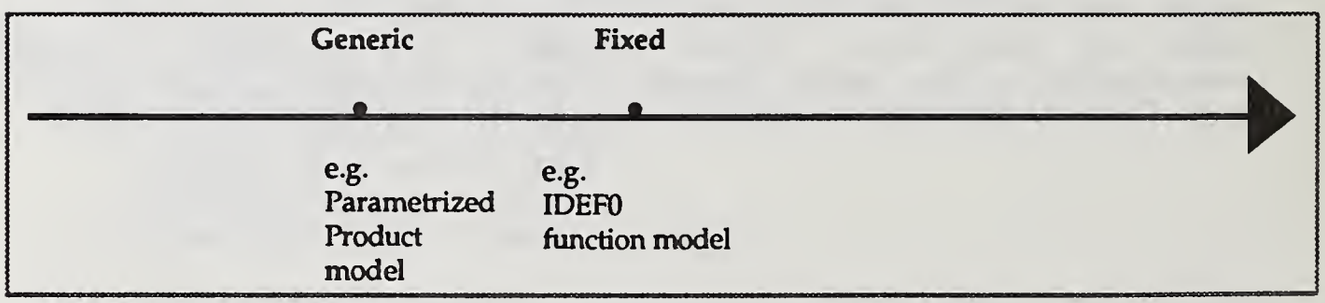


Figure 4-8. Specification Level Dimension

4.2 A new CIM definition

Basic assumption for the improved CIM definition is the idea that integration of any kind is needed to provide a better communication between different parts of an industrial enterprise. This communication is vital for the right coordination between the different parts which will now be called integration of business functions.

So the next question is: what kind of integration is further necessary to make business function integration possible or as we saw above makes an effective communication possible. Effective communication especially in a computer environment (with databases as buffers) is only possible if we can agree upon and fix the pragmatics, the semantics, the syntaxes and the medium for the information that is communicated.

The first two aspects (pragmatics and semantics) are typically issues for the CIM Architecture, whereas the last to aspects (syntaxes and medium) are issues for the CIM Infrastructure. It is clear that we need integration on all four levels. We now relate this observation to our prescriptive CIM Framework. In the Framework we identified three mayor aspects to be modelled:

Manufacturing Functions
Setting the requirements for the information needed.

Information
Covering the logical contents of the information which are used by or flow between the manufacturing Functions and also the specification of the manipulation of this information on a logical level (implementation-independent) and,

CIM Components
Covering Information Technology in the IS (the Information System) and Production Technology in the RS (the Real System that is controlled by the IS). In both cases we have hardware and software (applications, data structures, knowledge bases etc.). These CIM Components can be seen as the nodes in the CIM Infrastructure.

The I (of CIM) now deals with all three aspects:

Integration of Manufacturing Functions
Integration on this high level means the possibility of (computer supported) manufacturing knowledge transfer from one end of the product life cycle, requirements definition, via design, process modeling and production to the other end, the delivery and even the maintenance of the product.

Examples of backward knowledge transfer are the designers access to the information about what can be made (e.g. standard parts or features) and the process modellers access to information about machines, tools, manufacturing rules etc. Especially the backward knowledge transfer in the form of production data feedback is very important to continually improve the design and process modeling processes.

Integration of CIM Components

The basic integration needed to make all other forms of integration possible is the integration of the CIM components performing or supporting the manufacturing functions and generating, using and manipulating the information. It should be possible for the CIM Components (both IT and PT) to 'talk' to each other. As identified by the ISO-OSI (International Standards Organisation, Open Systems Interconnection) this communication takes place via several levels from the physical level to a level which offers services to communicate all kinds of information.

Integration of Information

Integration of Information means that we have one 'company-wide' conceptual schema for all information generated, used and manipulated by the manufacturing functions. This logical schema should cover all kind of product and process data and describe the information in a structured, unambiguous, non-redundant, consistent and if possible computer processable way.

However there is still a fourth form of integration: aspect integration. CIM not only deals with the separate integration of Manufacturing Functions, Information or CIM Components but also with the integration of these different aspects to one another. This is why the third kind of integration, the information integration, plays such an important role. It is the link between on the one hand the ultimate business integration and on the other hand the technical integration. Business integration sets requirements for the logical content of the information and this information should be represented, used and manipulated by CIM components. Or, said in another way: for Integration of Information we need Integration of CIM Components and for Integration of Manufacturing Functions we need Integration of Information.

This leads us to the following definition of CIM:

Computer Integrated Manufacturing is the Integration of CIM Components (Information - and Production Technology), together with the Integration of Information which make the Integration of Manufacturing Functions possible.

The overall goal is Integration of Manufacturing Functions. This integration sets also the requirements for the Information Integration which on its turn sets the requirements for the Integration of CIM Components which process this information. The Integration of these aspects is also part of the CIM concept.

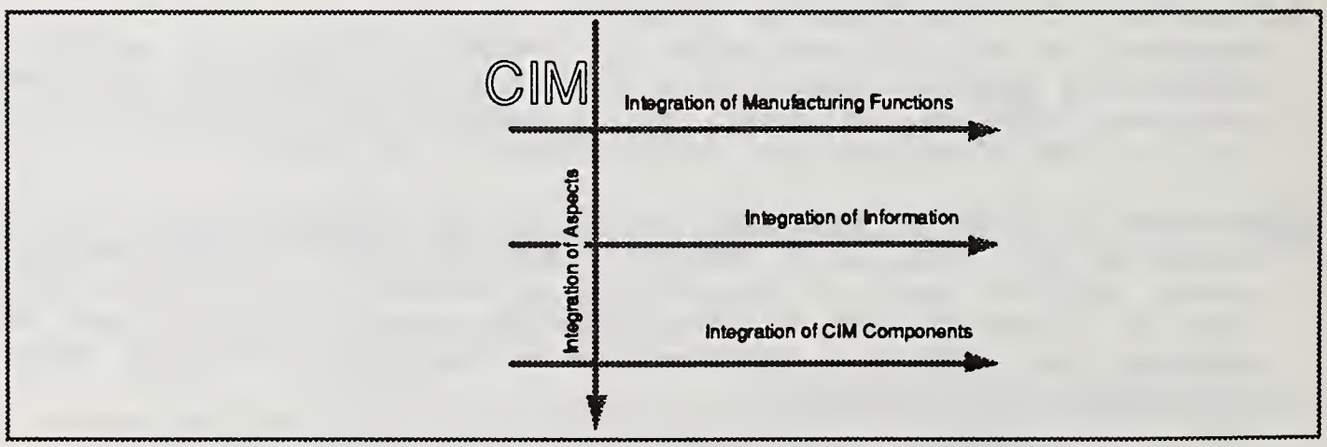


Figure 4-9. Different kinds of Integration covered by CIM

5 CIM Reference Architecture

5.1 A CIM Base Model

As you can see in the framework, there may be different languages to model the different aspects of CIM. The idea is that modeling CIM is best be done in an integrated way. We would like to model over the boundaries of the model types identified in the framework, instead of having a different language for every model type. Some groupings are already quite clear, e.g. you will not use different languages for different contexts (this will not favour the reuse). You will also not use different languages for different product life cycle stages, if possible. What we need is one language to model CIM, at least for our primary representation of the model. This means that all dimensions identified in the framework should be 'embedded' in this language.

For this moment we will concentrate on the modeling of a CIM Architecture (the aspects 'manufacturing functions' and 'information'). To be able to illustrate the idea of a CIM Base Model we will focus on two important dimensions: 'Composition Level' and 'Context' apart from the partial aspect dimension. Looking at the integration of aspects you could say that we follow some high level object-oriented approach and develop a language specialized for expressing CIM architectures, or as we call it a 'CIM Base Model'. We will describe this language in the next paragraphs and then give an image of a global CIM Reference Architecture in section 5.2.

The CIM Base Model consists of a basic construct, here called CADU (Cim Architecture Definition Unit). We use for this illustration only two abstraction mechanisms[1] which operate on this CADU: 'specialization' (representing the context dimension) and 'decomposition' (representing the composition level dimension). The CADU we use in this example[2] is a combination of related entities relevant for the CIM Architecture level:

- Manufacturing Function
- Output (Real object or Information)
- CIM Component

This CIM Base Model is represented in a NIAM-like way (see for pure NIAM [SOL82]) in figure 5-1.

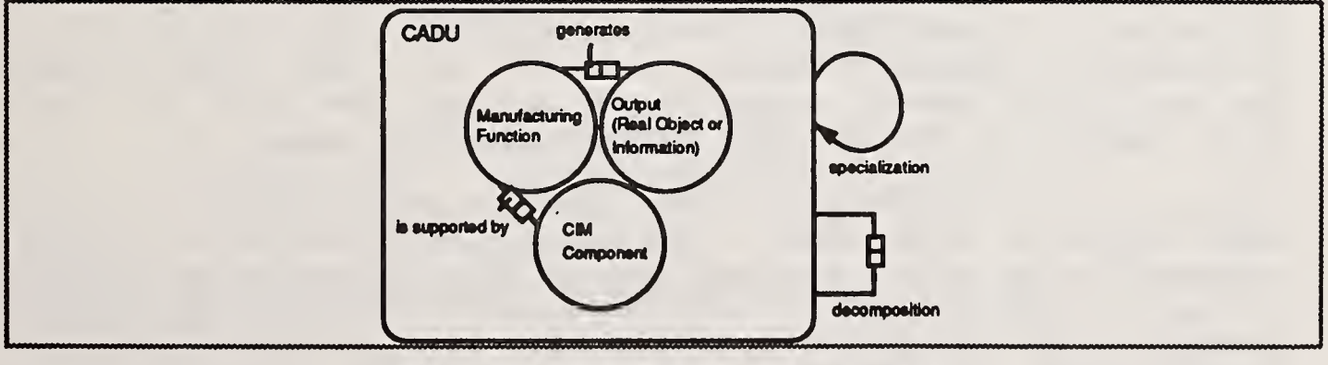


Figure 5-1. The CIM Base Model

---

[1] The complete CIM Base Model covers all the dimensions of the framework

[2] Other entities, like organizational unit, with other relations, can be added

The idea now is that when a CADU is specialized or decomposed, Function, Output and CIM Component are specialized or decomposed all together. While specializing a child CADU inherits the properties of its parent CADU and when decomposing extra relations (connections) between the parts of the whole appear (see figure 5-2). When specializing decomposed CADU's also the connections to other CADU's specialize.

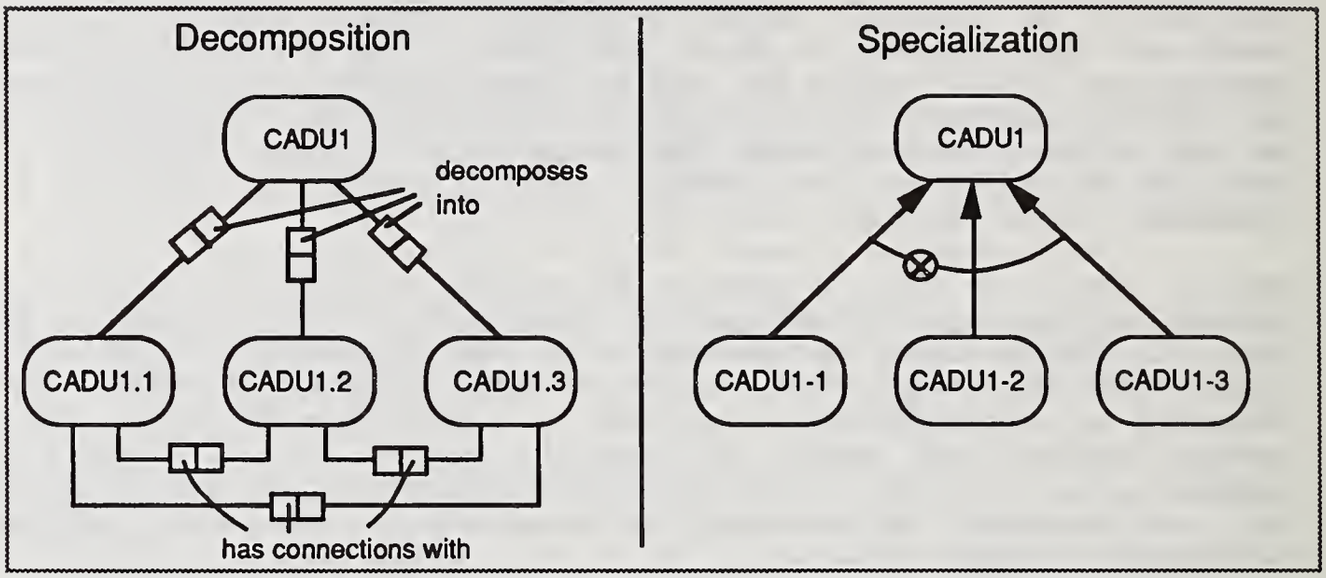


Figure 5-2. Applying the two abstraction mechanisms

With the CIM Base Model we can build up CIM Architectures. We will show here an example in the form of a part of a Reference model (see the Context dimension in the Framework). The model is partly 'general' for discrete parts manufacturing and partly 'restricted reusable' for complex shaped parts and sheet metal parts. The example we have chosen deals with the determination of 'how' a product can be produced. This activitivty is often referred to as 'production planning' or in this case 'process' modeling.

In figure 5-3 we see six CADU's: CADU1 to CADU4 are valid for discrete parts manufacturing in general. CADU1 decomposes into CADU2 to CADU4. At the same time all components of a CADU (function, output and CIM components) decompose. With this decomposition new relations between the sub-CADU's arise (the 'is input for' relationship). Then CADU3 is specialized into CADU5 and CADU6 which are valid in two more specific context (complex shaped parts and sheet metal parts respectivily). Again we see that all CADU components specialize together.

Althoug not shown in figure 5-2, it is also possible to specialize CADU1 for CSP and SMP, because the decomposition - and specialization dimensions are completely orthogonal. Because of the inheritance property, the specialized version of CADU1 for e.g. CSP will also have the same decomposition of CADU1. One specialized component of this decomposition is shown in figure 5-2 in the form of CADU 1.2-1. In other words: the specialization of the decomposition is the same as the decomposition of the specialization (because of the orthogonality).

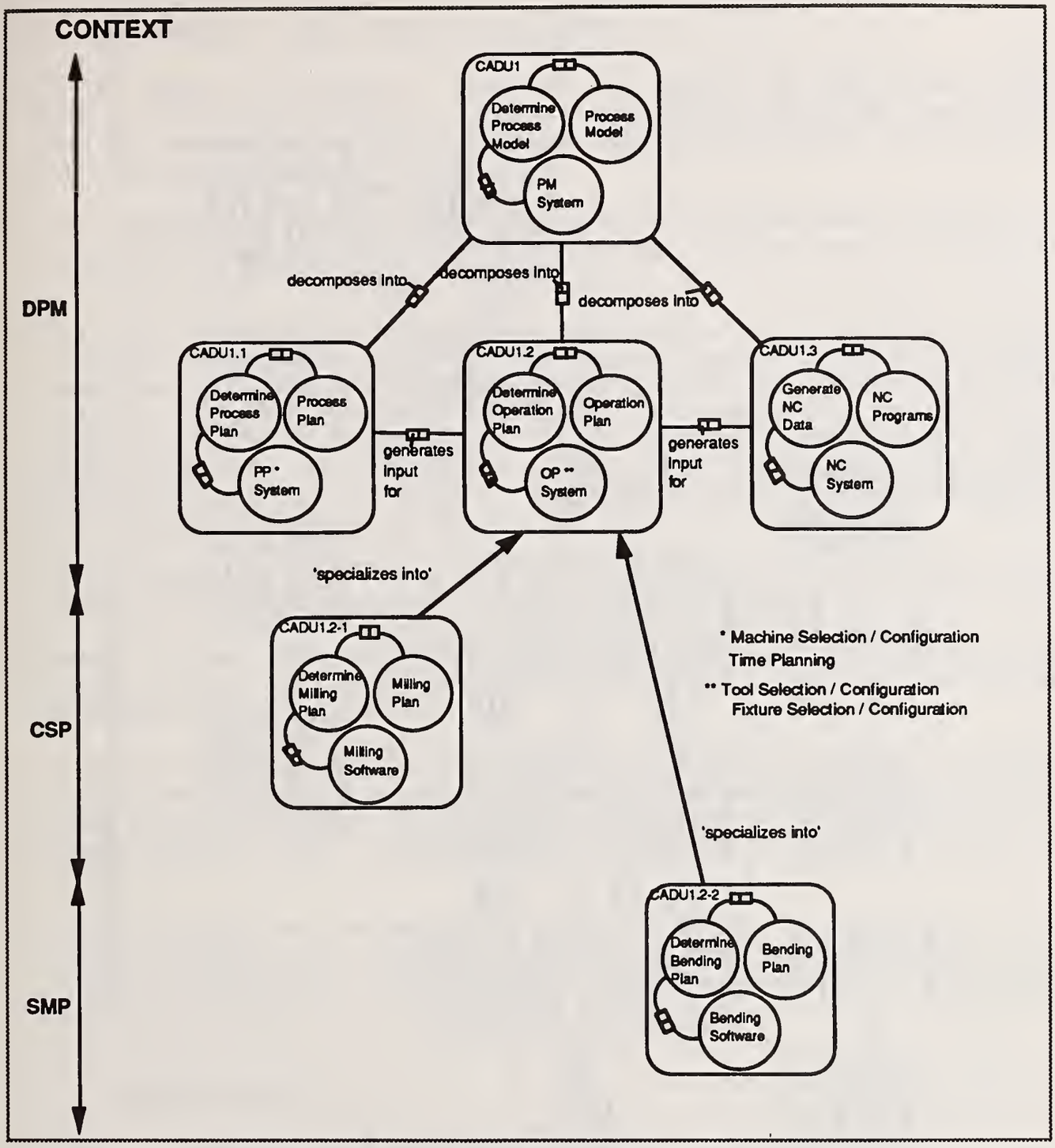


Figure 5-3. Example of the CIM Base Model

5.2 Using the CIM Base model: example of a CIM Reference Architecture

Finally we present a global CIM Reference Architecture for discrete parts manufacturing (figure 5-4). This model is of course not complete (the CADU's can be specialized and decomposed further) but it gives an idea of building up a CIM Reference Architecture and the way we can indicate the place of standards like MAP, TOP, EDI(FACT) and STEP in this architecture.

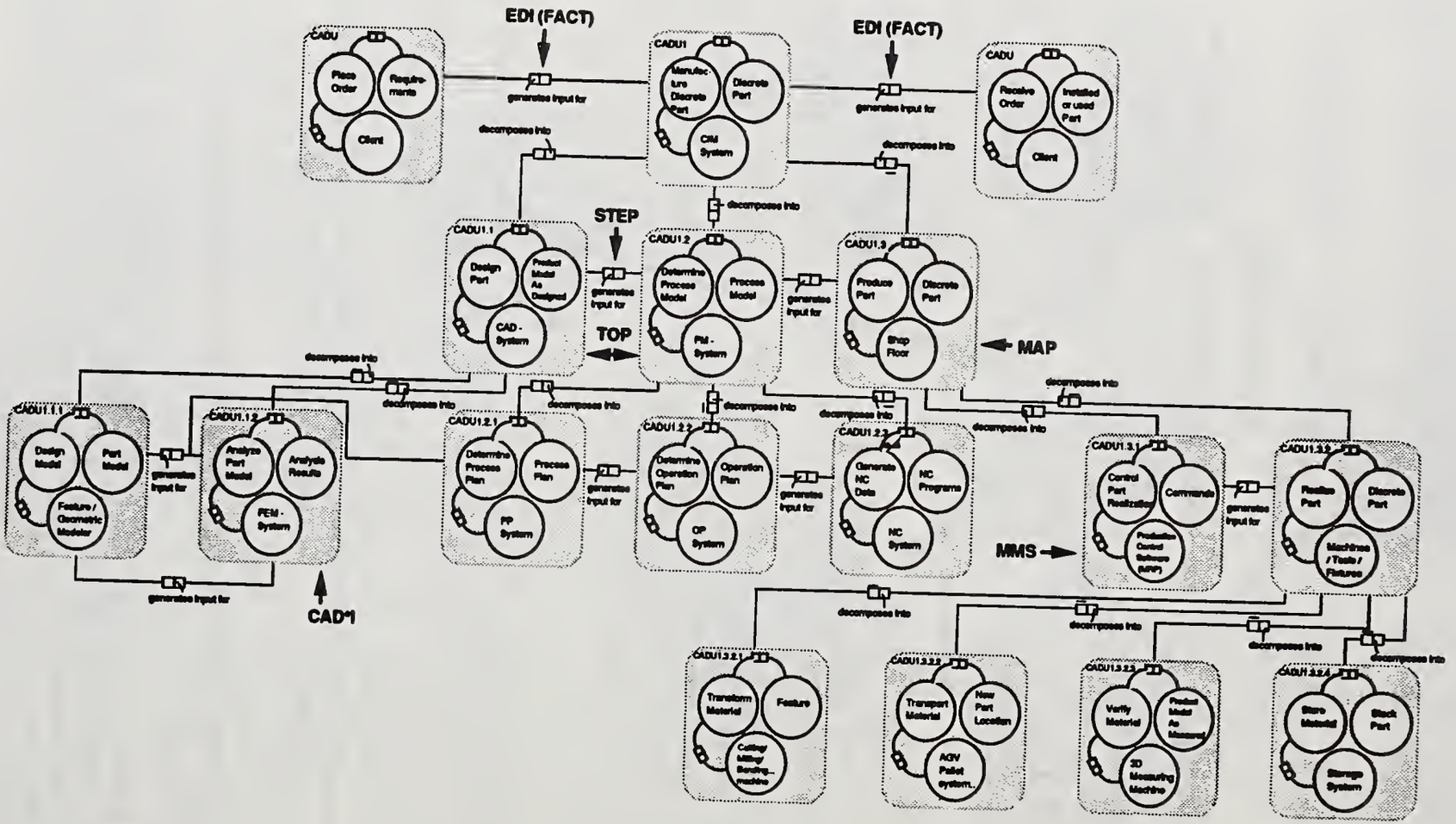


Figure 5-4. Global CIM Reference Architecture

6 Conclusions and further research

In this paper we developed a framework and showed how this framework can help us to relate and understand existing projects dealing with CIM. More analysis than the examples shown here is of course required (e.g. for all contributions for this CIMCON conference).

The instantiated, prescriptive framework gives us the possibility to define CIM more precisely and also to develop a methodology for CIM, based on this framework (this work is now going on at TNO).

The image of the CIM Base Model shown here only served to illustrate this paper. It appears that it is easy to represent the complete model in NIAM, but the models themselves become hard to represent in this graphical way because of the many dimensions involved. Therefore we are now investigating the possibility of a different primary representation than NIAM (algebras, predicate logic, object oriented techniques etc.) from which different, more user-oriented, views can be generated in the form of e.g. NIAM. These views might be 2D or 3D subspaces of the 9-dimensional framework space.

In this paper we concentrated on the modeling of CIM Architectures. When modeling CIM Infrastructures too, the CADU has to be adapted.

Finally a remark about the CIM Reference Architecture itself. Although it was not possible to show a complete reference model in this paper, we hope that the principles of modeling CIM are clear.

It is our intention to develop a complete CIM Reference Architecture expressed in a CIM Base Model that covers all dimensions. This architecture will be used to place all major (standard) CIM Components and interfaces. This architceture will, together with the defined CIM methodology, support the application of CIM in industry.

Finally we hope that we have made clear that for the specification of a CIM Reference Architecture, a CIM base Model, a CIM methodology and even the definition of CIM itself, a CIM Framework (preferably defined as an international standard) is crucial.

7 Abbreviations

CIM	Computer Integrated Manufacturing
CSP	Complex Shaped Parts
DPM	Discrete Parts Manufacturing
EDI	Electronic Data Interchange
ESPRIT	European Strategic PRogramme for Information Technology
IMPPACT	Integrated Modelling of Products and Processes using Advanced Computer Technology
IT	Information Technology
MAP	Manufacturing Automation Protocol
MRP	Material Requirements Planning
NIAM	Nijssen Information Analysis Method

PT	Production Technology
RIA	Reference model for Industrial Automation
SMP	Sheet Metal Parts
STEP	STandard for the Exchange of Product definition data
TOP	Technical Office Protocol

8 References

[AMI89] Esprit I AMICE 'CIM-OSA', Public Document, 1989.

[BOH89] H.M. Böhms, F.P. Tolman, Modeling the CIM Architecture, CIM Architecture seminar University of Karlsruhe, June 1989.

[CAM88] CAM-I Advanced Technical Planning Committee (ATPC), An Architecture of CIM, R-88-ATPC-01, 1988.

[CEN89] Framework for modeling CIM (draft preENV), CEN/CENELEC AMT WG-ARC N80.

[SOL82] H.G. Sol, T.W. Olle, A.A. Verrijn-Stuart, Information Systems Design Methodologies, Proceedings of the IFIP WG 8.1 Working Conference on Comparative Review of Information Systems Design Methodologies, Noordwijkerhout, The Netherlands, 10-14 May, 1982.

[SOl88] H.G. Sol, Information Systems Development: a problem solving approach, University of Technology Delft, January 1988.

[TOL89] F.P. Tolman, H.M. Böhms, TNO, Proposal for a Reference model for CIM Architectures, ISO TC 184 SC5 WG1 N98, January 1989.

MANUFACTURING SYSTEM DESIGN METHODOLOGY: EXECUTE THE SPECIFICATION

ROBERT P. JUDD, RAYMOND S. VANDERBOK,
MARK E. BROWN, JOHN A. SAUTER
Industrial Technology Institute
Ann Arbor, Michigan 48106

Abstract

An improved method for designing manufacturing systems is presented. Primary features of the improved method include frequent design iterations by executing the specification early in the life cycle, flexibility to accommodate changing requirements, coordination of engineering disciplines in evolving integrated solutions, and reuseability of commonly used software and hardware portions of a system. Benefits of the improved method include more optimal system designs, solving integration issues during design phase rather than installation, and verification of design specifications. The improved method is called XSpec™, which stands for eXecutable Specification. The XSpec notation and design process are described. Specifications are executed on a tool called XFaST™, which stands for eXecutable Factory Simulation Tool.

Keywords: concurrent engineering, continuous improvement, control design, design life cycle, design methods, manufacturing systems, modular system, object-oriented, rapid prototyping, simulation, XFaST, XSpec.

1. Introduction

Today's manufacturers are facing increasing pressures from rapidly changing markets. In the past, U.S. manufacturing has relied on mass production techniques to produce a narrow range of products. Today's markets demand a greater variety of products coupled with shorter product life cycles. In order to stay competitive in today's market, industry must learn how to shorten the time from product concept to production. We must also be able to quickly build the manufacturing systems which can respond to a changing market. This flexibility requires a greater role for computer integration in manufacturing.

The initial ventures into large scale advance computer integration are met mostly with disappointment. Projects tend to be late and over budget. Many systems, once they are running, never meet their designed production goals. Changes in the system are difficult to make. Much of the difficulty can be traced to the complexity involved in the integration of computers and factory equipment. Although there has been much progress made in product design (through concurrent engineering and new CAD/CAM tools) as well as software design, little work has been done to solve the problems

involved in designing a mechanical system which is tightly coupled with software control.

XSpec™ which stands for eXecutable Specification is an attempt to pull together some of the best approaches in industry along with some novel additions to address the particular issues involved in manufacturing system design. It provides a methodology for developing manufacturing systems that tackles the tough problems of systems integration. The methodology has been supplemented by a toolset called XFaST™ (eXecutable Factory Simulation Tool). Both the methodology and the tools have been used in several manufacturing system designs. This paper introduces the methodology and some of the advantages that it brings to the manufacturing community.

2. Traditional manufacturing system design

A summary of traditional design practice follows in order to identify areas of manufacturing systems design in need of improvement. We are focusing in this paper on the design of workstations, cells or individual lines as opposed to entire factories. At this level there are particular difficulties related to the combination of hardware and software that exist on a factory floor. The business and technical relationships presented here are typical, but do not attempt to describe the many possible variations.

Normally the end user has a product design. Based on the process technologies required to manufacture this product, the end user will contract a prime contractor (generally a hardware builder) to design and implement the manufacturing system. The end user defines process requirements to the prime contractor. The end user works with the prime contractor to select process specifics, such as using robots rather than hard automation.

The prime contractor designs the overall system and may subcontract portions of the system. The prime contractor purchases equipment such as robots, parts washers, and gaging stations. Subcontractors can provide both equipment and technologies that extend the capabilities of the prime. Typical technologies that are subcontracted include hydraulics, controls, and simulation. The prime contractor oversees the detailed design and fabrication. Partial integration of the system is performed at the prime contractor's site. After approval from the end user, the system is shipped to the plant for final integration, debug, and buy-off from the end user.

Deeply embedded in this overly simplified view of building manufacturing systems is the notion that one can march down a road from requirements to design to implementation to a working system. It assumes that each step is well known, without risk, and error free. It also assumes a static environment where nothing changes while the new system is being built. In addition, systems are generally built with fixed price contracts and hard schedules.

In order to meet this Herculean task of building manufacturing systems, end users and vendors may make many conservative decisions or gamble that the system can be coerced to work. This results in systems that are over built and take an inordinate amount of time to become operational.

3. Problems with traditional approach

There are several problems with the traditional approach to the design of manufacturing systems. A brief critical evaluation of the traditional design approach follows.

1. **Incomplete requirements.** Requirements are never complete. Even with the best efforts, requirements from the end user to the prime contractor or from the prime contractor to the subcontractors will be inadequate, incomplete, imprecise, or contradictory. Because of the complexity of interactions in even simple systems it is not technically possible or economical to know all the requirements before the system is built. Some requirements can only be discovered when the system is run and inappropriate system behavior is observed.

2. **Requirements are not static.** Requirements are never stable. Due to the long lead times required to procure manufacturing systems, product designs are seldom complete when manufacturing facilities are being constructed. A new marketing report can quickly change the desired production rate. Traditional methods require a repeat of the system life cycle for each change. Because of the large amount of work required, this usually does not get done.

3. **Disciplines are not coordinated.** The traditional design approach does not provide a facility for cross discipline coordination. For example, a robot programmer is unlikely to be able to understand the effect of variability in part placement on system production rates. Often the results of poor interdisciplinary coordination are not detected until final integration at the plant site.

4. **Long integration and debug time.** Factory systems take a long time to integrate and debug. This cost is the result of incomplete requirements and less than perfect execution. Significant reductions in the time and cost of integration and debug can be made if more errors are caught during the requirements and design phases.

5. **Inadequate design verification techniques.** Even the well intentioned system designer cannot, using traditional methods, verify the design. It is left to the experience of the staff to notice flaws in the design during long, boring, design reviews. Subtle interactions between devices are not likely to be found with this approach. Most often the designer must wait until the system is fully implemented before he/she can watch, in amazement, as the system runs below the design production rate (while the end user watches in disappointment).

In summary, experience has shown that development of manufacturing systems

cannot flow in a lock step fashion, from requirements to run-off. Problems of the traditional design approach reviewed here identify a need for an improved design method.

4. Goals of XSpec

Xspec is a method of design that has grown out of the experiences of software and control engineers involved in developing manufacturing systems. It pulls together the best techniques in industry from both of these disciplines. XSpec was developed to address the problems that are particular to manufacturing system design and are therefore not addressed by other software methodologies.

Specifically, XSpec addresses the problems in manufacturing design as outlined above:

1. **Incomplete requirements.** XSpec attempts to provide a framework where requirements models can be quickly built and easily analyzed. Current modeling techniques fail to communicate requirements and do not lend themselves to analysis. One technique employed by XSpec is the use of executable requirements models. These models can be animated or analyzed to better understand the requirements of the system. This leads to a more complete and precise requirements specification.

2. **Requirements are not static.** Changing requirements are a fact of life that must be accommodated in any manufacturing life cycle. It should always be the goal of any systems designer to reduce the number of changes that are made during the life cycle. XSpec accommodates those necessary changes by reducing the time and cost in making those changes.

3. **Disciplines are not coordinated.** XSpec addresses this area through a unique approach to system decomposition and through techniques which allow the hardware and software designs to be tested against each other. This provides a vehicle for different disciplines to cooperate on a system design and coordinate their efforts in making improvements early in the life cycle.

4. **Long integration and debug time.** XSpec provides the designer with an approach to system design which identifies the integration issues and accurately specifies the interfaces early in the life cycle. The XFaST tools provide a way to discover problems in software, hardware and subsystem integration before system implementation begins.

5. **Inadequate design verification techniques.** Design verification suffers from the same problems as requirements specifications. XSpec attempts to provide the systems designer with an approach which allows his/her evolving design to be quickly tested at different levels of detail.

5. Foundation of XSpec methodology

Xspec is built on the principles of object-oriented design and programming, and rapid prototyping. These are commonly cited as the two most promising areas of research in the design of complex software systems. XSpec extends these approaches into multi-disciplinary design arenas such as manufacturing systems.

XSpec uses a rapid prototyping approach to design that involves building executable specification models of the software and hardware in the system. These models allow complex systems to be analyzed from many different viewpoints. This ability to execute the specification allows the designer to understand how his designs will behave in the actual system. It has proven to be a powerful tool for validating and improving factory designs.

The term object-oriented has been applied to many things but in general it refers to design representations and programming languages which are based on objects. Objects are normally associated with a set of data and the operations which can be performed on that data. This binding of data and procedures is in contrast with traditional languages where the two are kept separate. The action in object-oriented systems comes from sending *messages* between objects. These messages invoke the procedures or *methods* within an object that operate on that object's data. Objects are usually organized into classes. Objects can inherit behavior (methods) and data from other objects to form many different kinds of objects. By using inheritance, objects can build upon existing systems and add new functionality where needed.

Many of the constructs in object-oriented systems map onto manufacturing systems. Objects can be used to represent factory devices. The "methods" are the operations which can be performed on those devices. Messages represent the digital I/O or other mechanisms used to communicate between the controller and the hardware. Messages can also represent the material flow in a factory. Similarly, manufacturing devices can be grouped into a class hierarchy. For instance conveyors might be grouped under a class of devices called "material transport". There would also be several classes of conveyor defined such as "accumulating" and "non-accumulating". Each of these classes has behavior which is similar to its parent class.

XSpec extends the work in object-oriented design by making a distinction between physical objects (representing the hardware) and control objects (representing the software). This distinction is critical to integrating and coordinating the mechanical and software disciplines.

6. XSpec constructs and notations

This section looks at the constructs and notations used by XSpec to decompose a system and build executable models of the requirements and design. A brief definition of terms is presented followed by a more detailed discussion.

The following terms identify the constructs of an XSpec model:

- **Physical Element** -- A physical element is a representation of the actual hardware in the factory system.
- **Control Element** -- The control element is a representation of the control strategy or software for the physical element.

- **Component** -- A component is a logical group of factory equipment (represented by physical elements) and the software which controls it (represented by control elements).
- **Message** -- A message is an interaction between elements. It is used to convey information or material between two elements.
- **Pin** -- A Pin defines services that are needed or provided by the element. A message leaves or enters an element through a Pin.
- **Connector** -- A Connector defines a set of related Pins on an element which are available for interfacing with other elements.
- **Path** -- Paths define how messages travel between elements. A Path connects a set of Pins (usually within a Connector) on one element with matching Pins on another element. It therefore defines all the messages between the two elements it connects.

6.1. Elements and components

The basic building block in XSpec is an *element*. An element is like an object in object-oriented systems. It includes data (state) and procedures (methods) which operate on that data. An element is used to represent the behavior of some part of the system. It can be thought of as a model. It also serves as a specification for that part of the system. In Xspec these models are built in such a way that they can be plugged together with other elements and executed on a computer. The element, therefore, is the basic executable unit within an XSpec model.

There are two kinds of elements in XSpec: *physical elements* and *control elements*. Physical elements represent factory devices or subsystems (i.e. the hardware) which respond to some controlling agent. Control elements represent those controlling agents (i.e. the software) in the system. An example of a physical element might be a section of conveyor track. It would include the motors to drive the conveyor and any sensors used. A control element could represent the software that reads the conveyor's sensors and controls its motor.

These elements are naturally grouped into a logical construct called a *component*. A component groups related control and physical elements together. Together these elements represent the hardware and software which make up a distinct piece of factory equipment. As will be shown later, this grouping allows a natural approach to system decomposition. Components are also used to enforce good control design practice by clearly defining the scope of control for a control element.

6.1.1. Element and component notation

Figure 6-1 shows the notation used by XSpec for diagramming elements and components. Components are shown by rectangles and elements are represented as boxes with rounded corners. Control elements are placed on top of physical elements. Components are named for the piece of equipment they represent (such as the name "Conveyor" used in Figure 6-1). An element's name identifies whether it is physical or control and to which component it belongs.

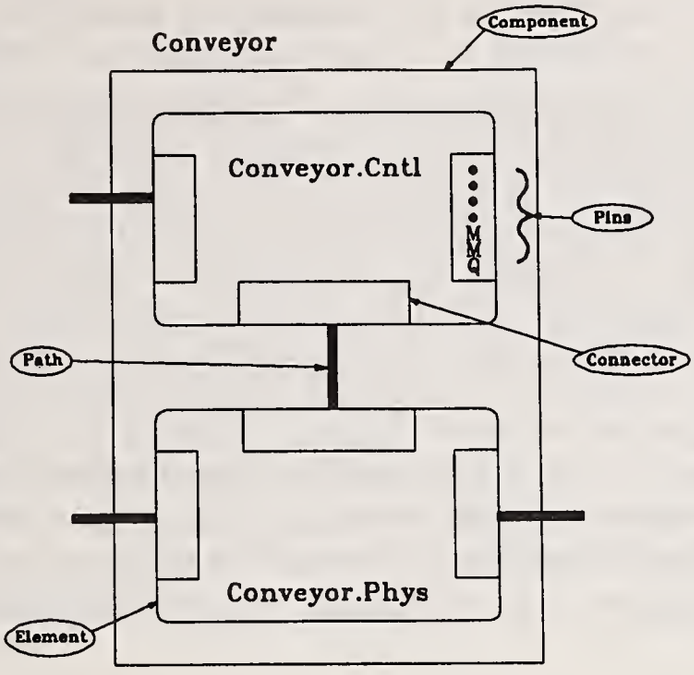


Figure 6-1: XSpec Component

6.1.2. Components Integrate Design Efforts

The physical elements are the focus of the design efforts of the mechanical designers. They hold the evolving designs of the hardware in the form of an executable specification or model. Because this specification is separated from the software it provides a "clean" specification of the hardware design.

The control elements are the focus of the software or control engineers. They hold the evolving design of the software in the form of an executable specification. This specification is also free of any hardware descriptions so it provides the software engineers with a clear specification from which to develop their code.

Separating these specifications is critical for proper development of hardware and software, but proper system design requires a specification that covers both. Rather than developing a third specification, XSpec brings these two models together within a component. This becomes the specification for both the hardware and software of a particular subsystem. This is not possible with traditional methods since they don't define a common protocol for the exchange of information between hardware and software models.

Components are then literally "plugged" together to form the specification for an entire system. Thus components provide the proper segmentation of design information with the required integration of software and hardware behavior to define the whole system. The ability to execute the entire specification allows a multi-disciplinary team cooperating on a system design to verify that their individual designs interoperate to achieve the overall system objectives.

Components are also used in decomposing the system into different levels of detail. Thus, what is represented as a single component at one level may become a whole group of components at another level. This allows the designer to reduce some of the complexity of the model and view the interactions of major subsystems. Although XSpec allows for a logical decomposition of the system into different levels of detail, an XSpec model is present only in the elements at the lowest level of detail.

Components can be grouped into a hierarchy of classes much like the class hierarchy of object-oriented systems. This allows new components to be developed by deriving them from an existing class of components. For example, one might have a class of components called SISO_Conveyor (Single Input, Single Output) which represents any conveyor section where material enters at one point and exits at one point. Two sub-classes could be developed to represent accumulating conveyors (such as power and free) or non-accumulating conveyors (such as chain and belt). Developing components in class hierarchies helps to reuse existing components. It also guides the designer in the development of new components which can be reused.[1]

6.2. Messages

XSpec uses the term *message* to describe all interactions between elements. A message represents the flow of information and material in the system. In an actual system, information flow can be accomplished through many different means: digital I/O, serial lines, local area networks, common memory, etc. XSpec can model each of these modes of communication through messages. This keeps the details of information flow out of the design while still giving the designer the ability to accurately model a particular system. Messages also allow the specification to be executed through standard computer techniques.

A message has a name which identifies the kind of information it conveys. For example a message named "PartPresent" could be used to indicate information from a part presence sensor. The message would be sent when the sensor (as modeled in the physical element) detects a part. A message can optionally have *parameters*. Parameters contain the additional data which may need to accompany a message. Thus the message "Move(Destination)" could be a command to a robot to move to the point described by the Destination parameter.

In XSpec notation a message is indicated by its name with any parameters enclosed in parentheses. Multiple parameters are separated by commas and different parameter values are separated by a vertical bar "|". Examples of valid messages would be:

- **OpenVent** -- a message with no parameters
- **Move(Dest)** -- a message with a single parameter
- **Move(XLocation, YLocation)** -- a message with two parameters

---

[1]Although objects make it easier to develop reuseable software it is always possible to develop something which is not reuseable. Developing class hierarchies to represent the application helps guide the designer in developing components which can be reused.

- **Conveyor(On | Off)** -- a message with one parameter which can take on one of two values.

Messages are represented graphically with arrows that are labeled as described above.

A control element can send a message to any other control element in the system. However, it can only send a message to physical elements which are within the same component as the control element. Likewise physical elements can send messages to any physical element in the system but only to a control element in the same component. This restriction enforces the design rules for maintaining proper scope of control.

It is helpful to look at the kinds of messages which are exchanged between different pairings of physical and control elements.

- **Physical to physical messages** -- These messages represent material flow and physical state changes. The physical elements use these messages to move parts and inform other elements of internal state changes which may affect their neighboring element. Example: a transfer of a part from a robot to a conveyor.

- **Physical to control messages** -- These messages typically represent sensory information. Physical elements model the sensors in the actual system. This sensory information is sent to the control elements in the form of a message. Example: a part sensor indicating that a part is present.

- **Control to physical messages** -- These messages typically represent actuation or control signals. Control elements control the hardware within their component by sending messages to a physical element. The physical element will use this information to change the behavior of its model to match what would happen in the actual system. Example: a signal to turn on a motor.

- **Control to control messages** -- These messages represent information flow and logical state changes at the software level. Example: a machine controller sends a signal to a part loader that it is ready for another part.

Say you have a robot loading parts onto a conveyor. A part sensor on the conveyor is used by the robot to determine when there is space for another part. In this system there would be two components which we will call Robot and Conveyor. We will model the robot component with two elements, Robot.C (control element) and Robot.P (physical element). The conveyor will also be modeled with two elements, Conveyor.C and Conveyor.P.

The Robot.C element begins by instructing the robot to put a part that it currently has in its gripper onto the conveyor. Robot.C would send the message "Move(Conveyor)" to Robot.P. The physical element would then model the time it takes to move the robot to the conveyor and return a "Done" message when the robot has reached the destination. The control element would then send the message

"ReleasePart" to Robot.P which would model the time to open the gripper. When Robot.P determines that the part has left the gripper it would send a "Done" message to Robot.C indicating the part has been transferred. At the same time it would send a message named "Part" to Conveyor.P. This message represents the transfer of the part from the robot to the conveyor.

Meanwhile, the Robot.C element is monitoring the part sensor on the conveyor. When the sensor indicates that space is available it should send a "SpaceAvailable" message to Robot.C. Because of the restriction on messages between control and physical elements, the message must originate within Robot.P. However the model of the sensor must exist within Conveyor.P (since the motion of the part across the sensor is modeled there). The situation is resolved by having Conveyor.P send a "SpaceAvailable" message to Robot.P which then forwards the message to Robot.C. This message between Conveyor.P and Robot.P is an example of a physical state change.

6.2.1. Message Types

In XSpec there are two types of messages: Commands and Requests. These types are distinguished by the form of the interaction between elements. Commands are one way messages from one element to another. They are used to represent information which flows in one direction only. There is no requirement that the receiving element return any information to the sender in response. For example, a controller sends a message to a conveyor motor "Motor(Off)". The motor does not send a message back to the controller so the flow of information is only in one direction. Commands are also useful in modeling digital I/O as will be seen in section 6.2.3.

Requests are a two way transaction between elements. When an element receives a Request message, it *must* return a Reply message. Like any message, Replies can have parameters associated with them. For example, a Press Station control element sends a "LoadPart" message to a Part Loader control element requesting that a new part be loaded. When the Part Loader has finished it replies with a "Done" message. These types of transactions are so common in manufacturing systems that the Request/Reply message pairing was created to easily support it.

Requests and Replies are listed in textual descriptions with a slash "/" mark separating the two messages. Thus, in the example above one would use the notation "LoadPart/Done" to designate the Request and Reply pair. A Request is shown graphically by listing the Request message on top of the Reply message with a bar in between. An arrow shows the direction of the Request half of the transaction.

6.2.2. Receiving Messages

An element receives messages in either a *mailbox* or a *queue*. A mailbox holds only a single, and most recent copy of a message. Queues hold a copy of every message in FIFO order until they are processed. Commands can be received in queues or

mailboxes. Requests can only be received on queues since a Reply must be returned.[2] Elements can wait for the arrival of messages in mailboxes and queues and read the contents of the message.

6.2.3. Modeling with Messages

In order to understand how messages can be used to model information and material flow in the factory we will look at a few examples.

- **Digital input, edge triggered** -- The rising or falling edge of a digital input is often used as a signal or trigger in control systems. Such a signal can be modeled as a Command with a single parameter indicating the level of the signal (high or low). The Command is sent to a mailbox when the signal changes level. The arrival of the message is the signal to the receiving element. If only a single edge is of interest, the parameter can be eliminated and the message sent only on the proper signal transition.

- **Digital input, level sensitive** -- A signal level is used to indicate the state of some device. The receiving unit polls the state of the line when it needs to know the current state of the device. This is modeled as above with a Command with a single parameter indicating the level of the signal. The Command is sent to a mailbox. The receiving element simply checks the current value of the message parameter in the mailbox at any time to determine the current state of the modeled line.

- **Serial input** -- Serial communications is commonly used to send parametric information which is difficult to specify with simple digital I/O. This can be modeled with messages and parameters that are sent to queues (since most serial inputs are also queued). The choice between using Command or Requests depends on whether a response is required to a particular message.

- **Software procedure call** -- Within a control program, information is exchanged between modules through parameters on a procedure call. Procedure calls are easily modeled through the use of a Request/Reply message pair. The Request parameters are the input parameters to the procedure and the Reply parameters represent the output parameters.

- **Common memory** -- Software modules may also communicate through a common memory pool. One module stores a value in a memory location which another module is free to read at any time. This is modeled as a Command sent to a mailbox with the parameters used to reflect the values written to memory.

---

[2] A Request arriving at a mailbox could be overwritten by a second Request before the element had a chance to recognize and respond to the first message.

- **Material Transfer** -- Material transfer can be modeled as a message. The transfer of a part from one physical element to another is modeled with a Command message sent to a mailbox input.

Figure 6-2 shows an example of how manufacturing equipment and its information and material flow can be represented with components, elements and messages.

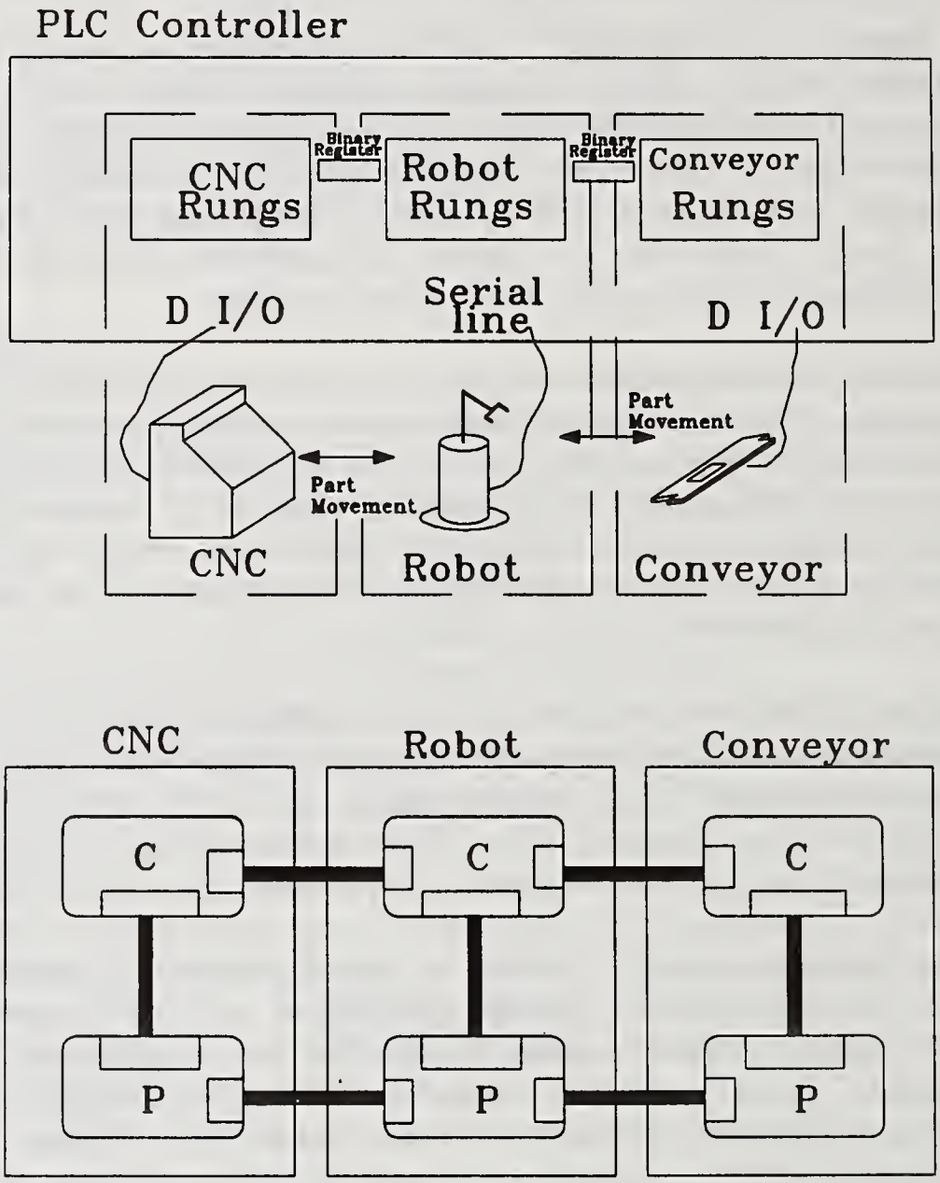


Figure 6-2: Example XSpec Model for a Manufacturing System

6.3. Pins, Connectors and Paths

XSpec diagrams showing elements with messages between them begin to resemble circuit or wiring diagrams. This analogy is extended further with the addition of *Pins*, *Connectors*, and *Paths*. These constructs help promote the building of generic elements that can be reused by "plugging" them into a new application. This is similar to the

use of standard electronic components such as integrated circuits. One of the keys to developing such reuseable elements is a well-defined interface of all the inputs and outputs. Pins and Connectors are used in XSpec to define the element's interface in this manner.

All messages which are sent by an element exit through a Pin. Similarly, all messages received by an element enter through a Pin. Pins are grouped into Connectors which are named. The named Connector identifies a set of Pins. From the perspective of an element, a Connector provides a port or a "window" to another element. Paths are used (like cables in electrical wiring) to connect the Pins of one element's Connector with the matching Pins on another element's Connector.

The element sends messages by naming the Connector and the Pin. It does not specify which element will receive the message since an element is actually only aware of what Pins it has, not which elements are connected to those Pins. The applications designer determines the actual destination of a message by using a Path to connect the source Pin to the proper destination Pin. This allows an element to be designed without concern for the actual elements it will be connected with in a given application.

6.3.1. Pins, Connectors and Paths notation

Connectors are graphically represented as boxes on the edges of an element along with the Connector's name. Pins are represented as dots within the Connector (one for each message sent and received on that Connector). Input Pins are labeled with a *M* if it is a mailbox input or a *Q* if it is a queued input. XSpec component diagrams (diagrams showing the interconnection among components) show Paths. Element diagrams (a blowup of a single component) show individual messages and Pins. Figure 6-3 is an example of an element diagram with all the Connectors, Pins and messages defined. Figure 7-4 shows an example of a component diagram.

7. XSpec design process

In general any design process can be viewed as the development of a series of models which gradually move from the problem domain (an abstract concept) to the application domain (the real-world system). Each of these models acts as a specification; it defines the behavior of the system in response to external events and its internal state. Normally one moves from a requirements model to a design model to the final model which is the implementation. Sometimes there are additional intermediate models developed as well. Every methodology provides a different language (and notation) for defining these models. This has proven to be a barrier to design. Once a requirements model is built it is difficult to translate the syntax and semantics of that model into the language used for the design model. Yourdon once described this transformation as something akin to a miracle.

Object-oriented design has made great progress in developing a single language and notation which can be used to build a system model which reduces the distinction between requirements and design. This allows changing requirements to be brought into the system quickly since the changes can be easily translated into the design language. XSpec capitalizes on this property of object-oriented design.

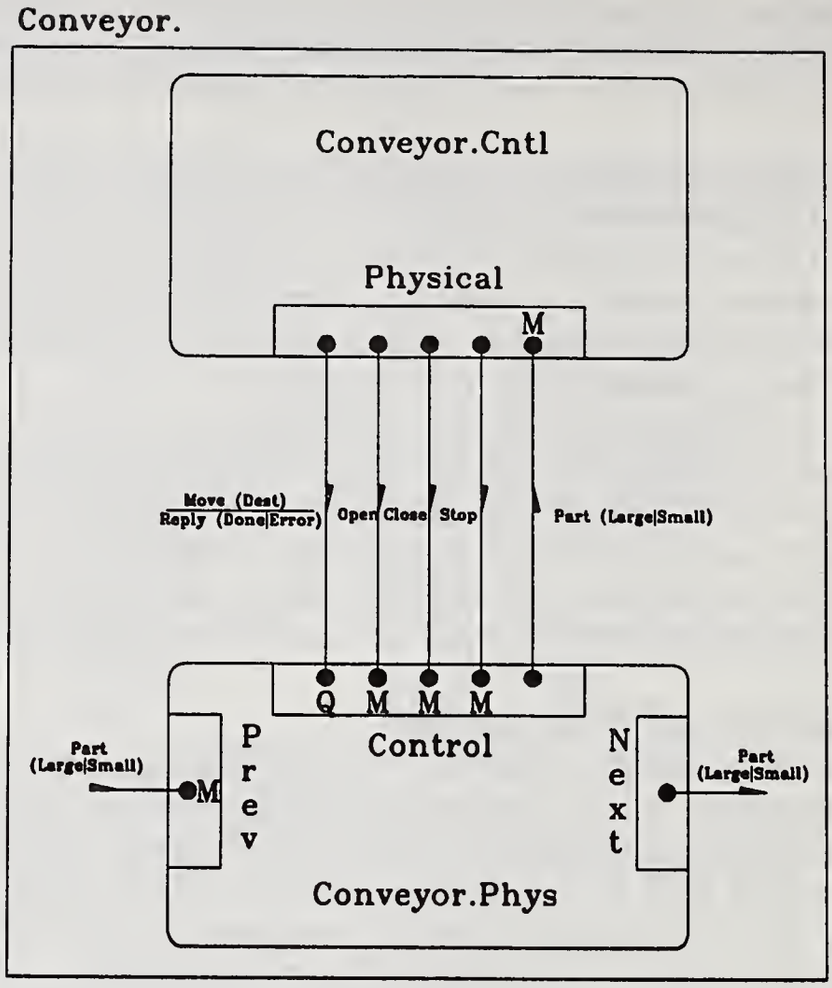


Figure 6-3: Component Diagram with Message Specifications

XSpec allows different parts of the model to exist in different stages. Each element can progress at its own pace along its own life cycle. This allows the inclusion of library elements which might be completely designed with other elements which are very early in their development life cycle. Thus it is not possible to clearly point to a single "requirements" model or a "design" model in XSpec. XSpec allows the designer to progress to different stages of the life cycle and different levels of detail for each element. This gives the designer the ability to explore areas of potential risk in greater detail then back out and continue the development of other elements along the life cycle.

The XSpec manufacturing systems design life cycle includes the following steps:

1. System requirements
2. System partitioning
3. Element specification
4. Execution and testing of the the system specification
5. Detailed specifications

6. Implementation, integration, and maintenance.

Each of these steps is detailed in the following sections. The steps are graphically illustrated in Figure 7-1.

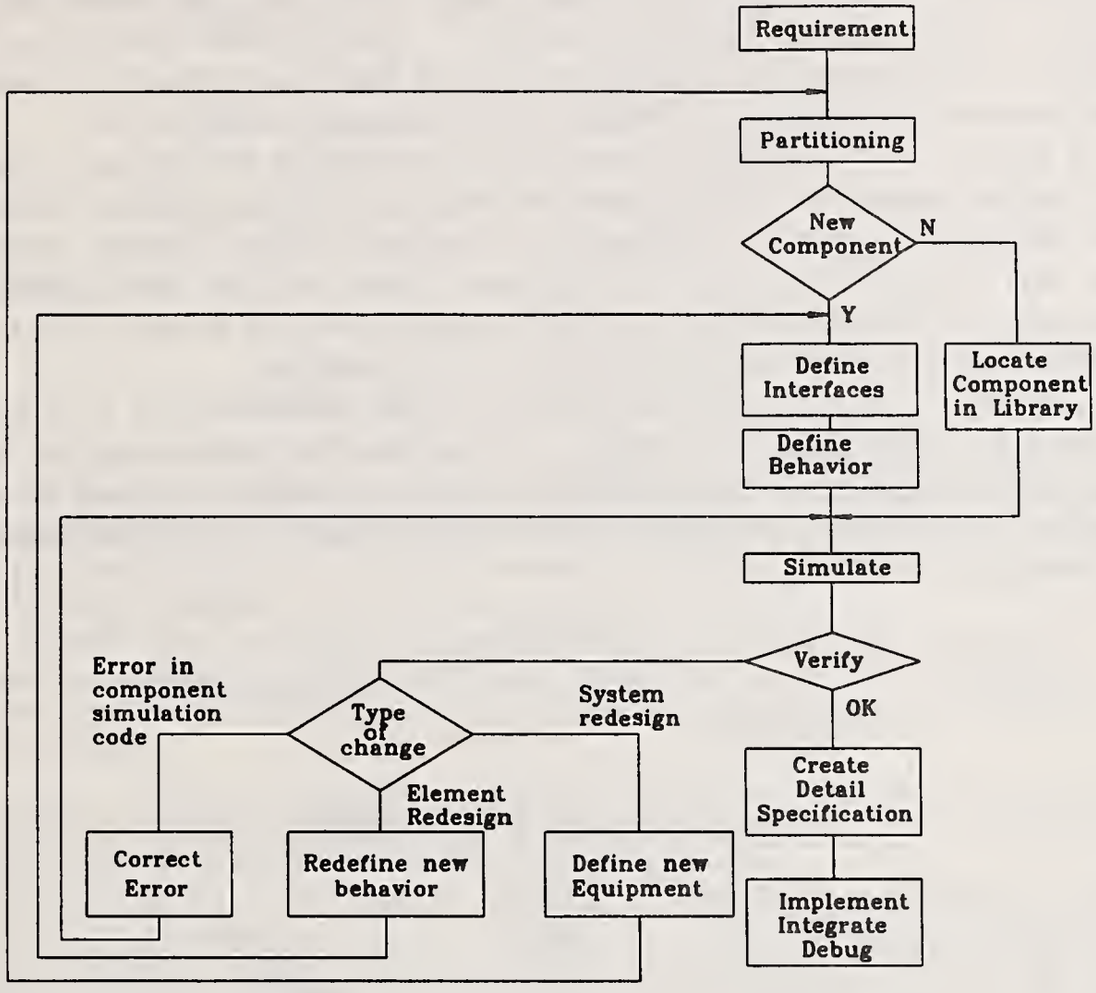


Figure 7-1: XSpec Life cycle

ITI is developing a set of integrated tools to support each step of the XSpec life cycle. Together these tools are called XFaST which stands for eXecutable Factory Simulation Tool. Currently XFaST only supports steps 3 through 5 of the XSpec life cycle. In the future all the steps will be supported through these tools.

7.1. System requirements

Gathering the requirements for a system involves obtaining information from the customer about the behavior of the system as well as any constraints or standards which must be followed. A requirements document is produced that lists constraints, process needs, production capacity, interfaces to other equipment, relevant standards, mechanical layout, and general strategies for implementation.

An XSpec designer should plan on the requirements evolving during the design process. As the design progresses, more is learned about the system and requirements change accordingly.

7.2. System partitioning

All methodologies incorporate some mechanism for partitioning the system into smaller units. Traditionally, the hardware is designed and all I/O is specified. From this description, control designers group related functions and design the control strategy. The grouping is somewhat arbitrary. There is little chance that the next system will use the same partitioning or the same functions. Therefore, there is little chance for reuse of the design. This situation is illustrated in Figure 7-2.

In XSpec, the system is partitioned along the lines of well-defined objects in the system such as process machines or material handling equipment. Figure 7-3 shows how XSpec partitions a system. Although the functions within different manufacturing system may change drastically, the equipment used and the basic operations they perform do not. By partitioning the system along those lines we are much more likely to develop components which can be reused on future designs.

A library of components is developed which can be searched for those components matching the requirements of a system. A class hierarchy helps guide the designer in finding and choosing appropriate components from the library for a new system. If no appropriate components exist then a new component class is created and placed in the class hierarchy.

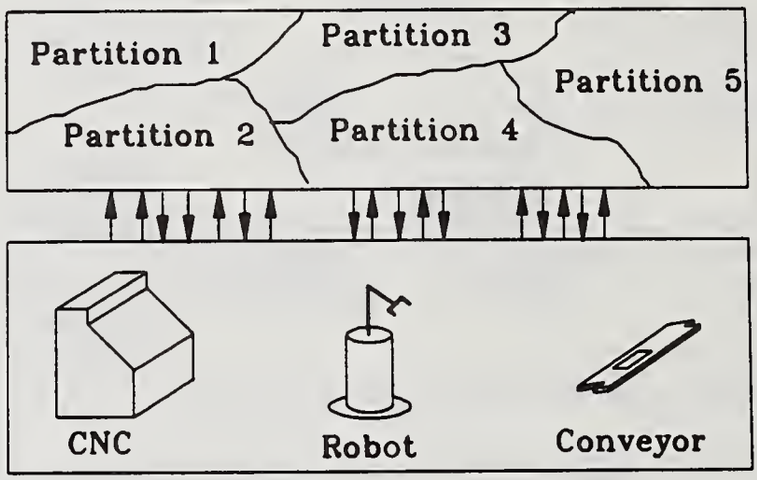


Figure 7-2: Traditional Partitioning

7.3. Element specification

As noted above most elements can be retrieved from a library of standard components. Those elements which do not exist in the library can be derived from existing elements or created from scratch. The procedure for developing a new element is described below.

7.3.1. Define the interfaces of a new element

To create a new element the designer begins by defining the interfaces. The following procedure is used:

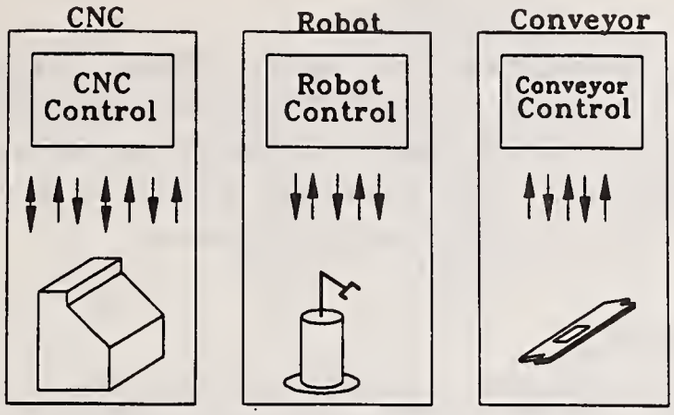


Figure 7-3: XSpec Partitioning

1. If this element will have a Path to elements already defined, then a Connector can be established for this Path. The Connector must match the message names of the corresponding library Connector.
2. Identify other Connectors and their messages which may be required for elements not yet defined.
3. Walk through the operation of the element and see if any additional messages are needed. If the additional messages are added to existing Connectors, add the corresponding Pin to the other Connector on the other side of the Path.

The interface definition does not have to be correct on the first try, there will be plenty of opportunity to experiment and test out the interface.

As the elements are being defined, missing or conflicting information in the requirements specification will be found. This is one of the advantages of the XSpec methodology. It helps to flush out errors in the requirements early in the life cycle before designs have been created. This prevents many of the surprises that normally occur in the implementation or integration phase of a project.

7.3.2. Describe the functionality of the element

The behavior of the element must now be described in terms of an executable model. XSpec requires the use of modeling languages which can be executed or analyzed directly. Here is where the use of a tool such as XFaST is critical to the employment of the methodology. XFaST provides a suite of modeling languages and techniques from which the designer can choose. Currently it includes tools for describing control (using Grafcet or ladders), and physical behavior (using queuing models or kinematic models). Appropriate support has been added for sending and receiving messages between elements.

There is no perfect language for developing executable specification models. Ideally the language would be graphical and allow for hiding of details. It should be self documenting so the models can be used directly as a specification statement. Languages which can be extended are also important since extensions can be used to develop a very high level language dedicated to a particular application domain.

We are currently experimenting with several different languages none of which meets all the criteria. Nonetheless, most of the advantages of XSpec can still be achieved with what is currently available. XSpec is not limited to the use of any particular modeling language. This allows the method to include new and more powerful specification languages as they are developed.

7.4. Execution and testing

Once the element models have been developed they can be tested individually or in groups. XFaST allows element models developed on different tools to communicate with each other during execution. It handles all the necessary routing of message between tools. XFaST also synchronizes the advancement of time across all the tools in a way which is transparent to the applications designer. The distributed simulation environment of XFaST provides the necessary support to execute and test complex system designs.

Because the models can be executed, their behavior can be observed and analyzed. Animation of the physical elements can help visualize how the system operates. Animation of the Grafcet diagrams allows the designer to see a software design being executed and the states that it enters. XFaST maintains a log of all messages which can be used as test data for validation of the designs. Performance data can also be obtained from this data. It can even be used to predict average and peak loads on different communication lines.

As errors are discovered either in behavior or performance they can be corrected quickly. The ease of making changes allows several approaches to be tested to find the optimal strategy. This contrasts sharply with the normal process where control engineers, struggle at the last minute to try to fix major design flaws that where not discovered until the equipment was installed and integrated at the customer's site.

The advantage of XFaST over a typical simulation is many-fold. In XFaST the actual software designs are driving the simulation. A normal simulation model is only a rough approximation of the control algorithms. It is not written by the controls engineer, nor is it used to develop the software for the system. Simulation tools use standard queues and stations which are only idealized representations of the physical system. They tend to make many simplifying assumptions about the actual details of material movement. XFaST designs can model details down to the operation of a single digital line or sensor in the system. This gives the systems designer the tools needed to test a design at any level of detail desired. A risk-driven approach is used to determine which elements should be modeled in greater detail to avoid overly complex models.

One can move up and down in level of detail. Once a subsection of a factory has been designed and debugged, this entire subsection can be aggregated to a single simple aggregate component model. Then another component can be decomposed into many components to explore details of its design. Decomposition is the opposite process of aggregation. Figure 7-4 illustrates the processes of aggregation and decomposition.

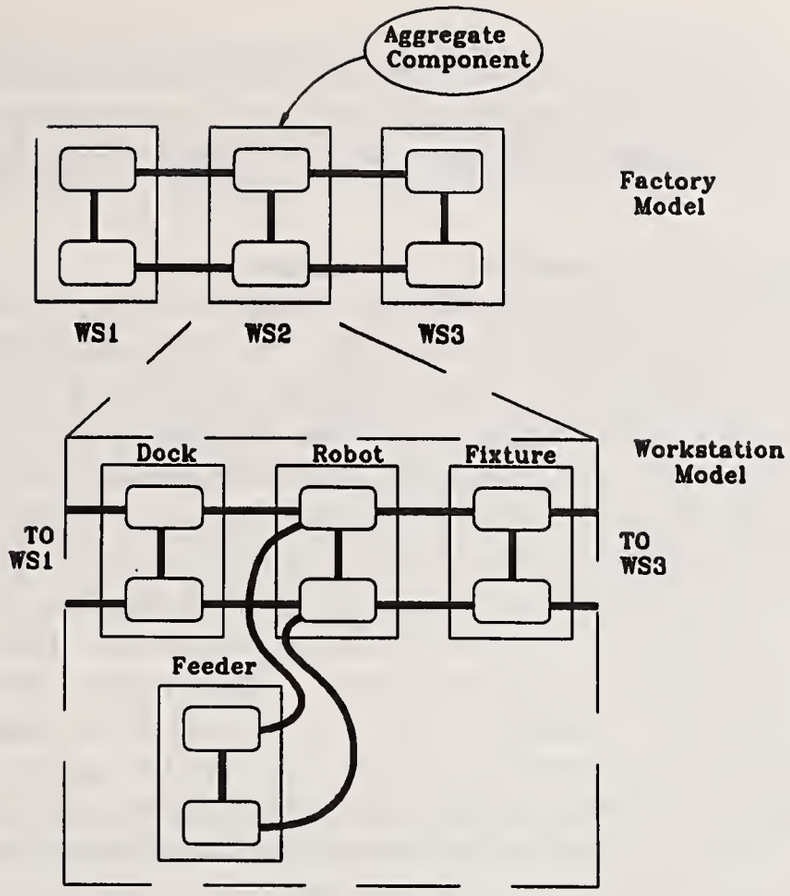


Figure 7-4: Aggregation / Decomposition

7.5. Detailed specifications

The life cycle of each element progresses at its own pace. XSpec allows the designer to explore different elements in greater detail as he chooses. The designer could put together a rough model for one component (more like a requirements model) and develop other components into a detailed design specification. This matches the actual progress of a project where some parts of the system may be designed and implemented before other parts.

In XSpec, elements tend to *evolve* along a path toward a detailed design specification rather proceeding in distinct phases. There is no clear break between what might typically be referred to as a requirements model and a design model. Because the same language is used to specify both models the transition from requirements to design becomes a process of constraining the use of certain modeling constructs inappropriate to the chosen implementation environment. For example, if a robot software design is being developed for a robot controller which does not support multiple tasks then the designer would restrict the use of parallel execution in the design model.

Changes in the requirements can be quickly implemented in the element at whatever stage the element is at. It is not necessary to return the element to an earlier stage unless the change makes drastic changes in the behavior of the element.

When the designs have been thoroughly tested, the executable design model is the

detailed specification. The element interfaces serve as preliminary I/O wiring lists. In other words, once the model is debugged the specification is complete.

7.6. Implementation, integration, and maintenance

Ideally one would want to have the actual code generated directly from the detailed model. Currently this is not possible, but automatic translation can produce most of the code. The rest of the code must still be generated manually, but the coder is able to work from formal design specifications which have been thoroughly tested.

Although more time is spent working on the design than in traditional life cycles, the implementation, integration and debug times are significantly shortened. Most of the design or requirements errors will have been detected before implementation begins resulting in only minor corrections having to be made during later phases.

An XSpec model also provides significant advantages during maintenance and upgrades. No manufacturing system is static. Requirements will continue to change even after the hardware is installed. New product designs or product mixes may need to be produced. Performance improvements and system tuning can be done on the model. Changes to the system can be made off-line and tested without having to bring the line down. When the changes have been tested on the model, integration of the new code with the manufacturing system will proceed much more quickly.

8. Conclusion

This paper provides an introduction to XSpec, a methodology for the design of manufacturing systems. In comparison with the traditional design approach, XSpec offers numerous benefits to a factory system designer. By using an executable specification of the entire system early in the design life cycle, many design and integration errors are detected before implementation. XSpec allows teams of hardware and software engineers to cooperate and communicate during the design process. XSpec also facilitates the reuse of existing designs by its approach to partitioning and through its use of a class hierarchy.

Xspec design notation is presented to help designers become aware of the conventions used to describe system partitions, messages, and physical and control aspects of a system. The notation is capable of handling simple and complex design applications. All activities within the system are described in terms of elements. Elements represent the physical hardware and control strategies of the devices in the manufacturing system. The interactions between elements are described in terms of messages. The XSpec design process is described. This process promotes reuse, innovation, and correct designs. In general, XSpec defines a single consistent representation of the manufacturing system that allows execution and verification of a specification.

XSpec has already demonstrated its value in several actual manufacturing design applications. Additional opportunities for methodology development exist through application in a broad range of system designs.

An Integrated CIM Architecture - A proposal

D. CHEN, B.VALLESPIR, G. DOUMEINGTS
GRAI Laboratory, University of Bordeaux I, FRANCE

This paper intends to present a skeleton of an integrated CIM architecture which is supported by a well defined method. It was elaborated by GRAI laboratory of University of Bordeaux I in the frame of two Esprit projects: project 418 (Open CAM systems) and project 2338 (IMPACS/Integrated Manufacturing Planning And Control System). The objective is to develop an overall approach for CIM solution. We think that an overall CIM approach should comprise a reference architecture which acts as a reference model for the analysis and design, and a method which shows how to build the architecture of studied enterprise from the reference architecture. First, the paper will propose a modelling framework defining the content oF the overall approach. Then the architecture, the method and the formalisms used will be presented in detail.

1. Problem statement

Trying to define what is an overall approach for CIM, we will link it to some concepts dealing with architectures, design methods and formalisms.

Within European ESPRIT program, efforts are being made to seek for overall CIM solution. For example, ESPRIT Project 688 has developed a CIM Open System Architecture (CIM-OSA), ESPRIT Project 418 has developed a GRAI Integrated Method (G.I.M) to design CIM systems etc... However, CIM-OSA is more architecture oriented in which no method has been proposed so far to show how to build an architecture. G.I.M is more method oriented in which no reference architecture has been defined to support the method.

It is now particularly considered that no amount of sophisticated tools and techniques to design CIM systems are really useful unless they are developed within an architecture context, no CIM architecture is really usable unless it is supported by a well defined method. The experiences and lessons learned in these projects lead us to look for an overall approach in which a reference architecture, a method as well as formalisms to build the models are developed jointly and with consistency.

2. Modelling framework

When analyzing, designing and implementing a CIM system, various concepts and models need to be defined and built. In order to ensure the completeness, consistency and integration between the concepts and models, we propose to define a modelling framework in which all models needed for the analysis, design and the implementation of CIM systems find their place (Fig.1) [VAL 90]. In this scope, the GRAI laboratory is developing a modelling framework within the GIM method. But in Esprit IMPACS project the proposed modelling framework is slightly different in order to take into account the various constraintes of the project.

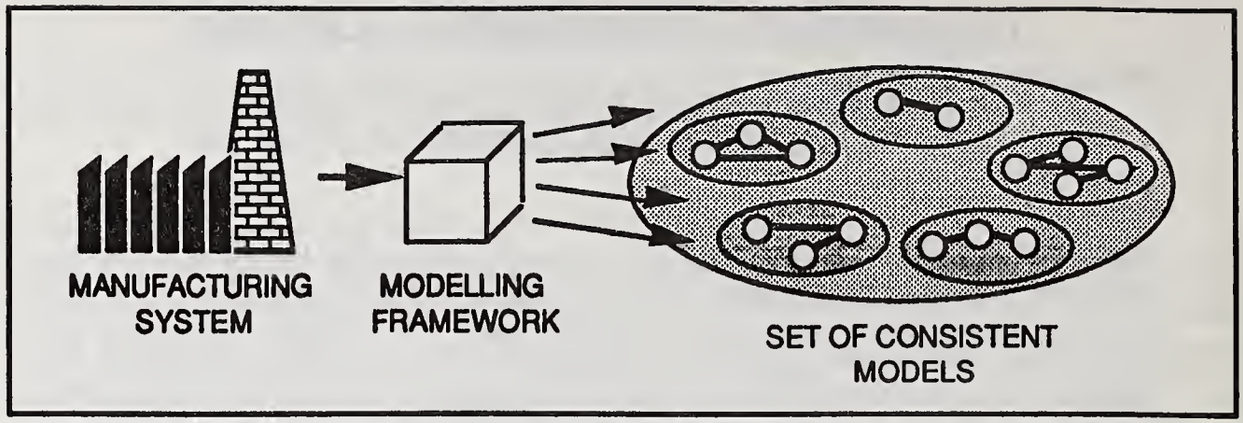


Figure 1. The use of modelling framework [VAL 90]

2.1 The GIM modelling framework

This modelling framework comprises the three following axis (Figure 2):

- Views/Decomposition: Decision, physique, information
- Abstraction: Conceptual, structural, realisational
- Reference: Generic, ..., partial, ..., particular

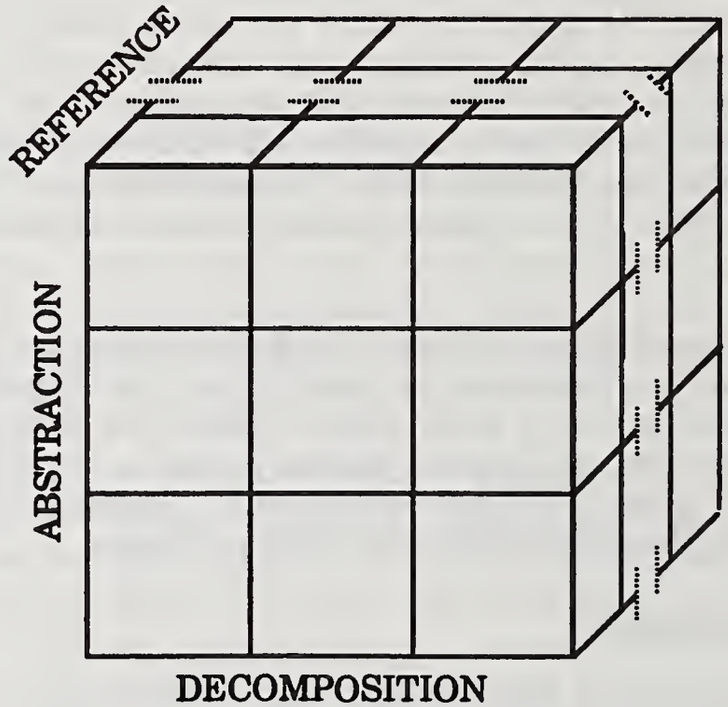


Fig.2 The modelling framework developed in GIM

A view is a selective perception of manufacturing systems. It concentrates on one particular aspect and disregards others. Each view can be represented by a view model which allows coordinated modelling, structuration and optimization of a specific aspect of CIM systems.

An abstraction level describes the degree of detail of model building. Three abstraction

levels have been defined; namely conceptual, structural and realisational. The concept of abstraction level is also related to the concept of integration. Our objective is to achieve the integration in all abstraction levels. The physical integration at the realisational level should be coherent with the integration defined at the conceptual and structural level.

The reference describes the degree of genericity of the models. The particular architecture of a studied enterprise can be derived from the reference architecture and generic architecture building blocks.

2.2 The IMPACS modelling framework

The modelling framework proposed in IMPACS project presents the following differences with respect to the previous one:

- IMPACS framework is composed of "View axis", "Abstraction axis" and "Project life cycle axis". The reference axis appears no longer. Because IMPACS develops only one type of reference architecture for discrete part batch manufacturing.
- A functional view is added in IMPACS modelling framework. Because it is considered that functional view is easy to be understood by the user, and the functional view model can be a basis to achieve the integration of other views.
- The project life cycle axis is added in order to show the relation between the structured approach and the different models.

So, this IMPACS framework specifies the different viewpoints, different degree of detail of models to be built as well as the various steps of CIM project life cycle (see figure 3) [DOU 90].

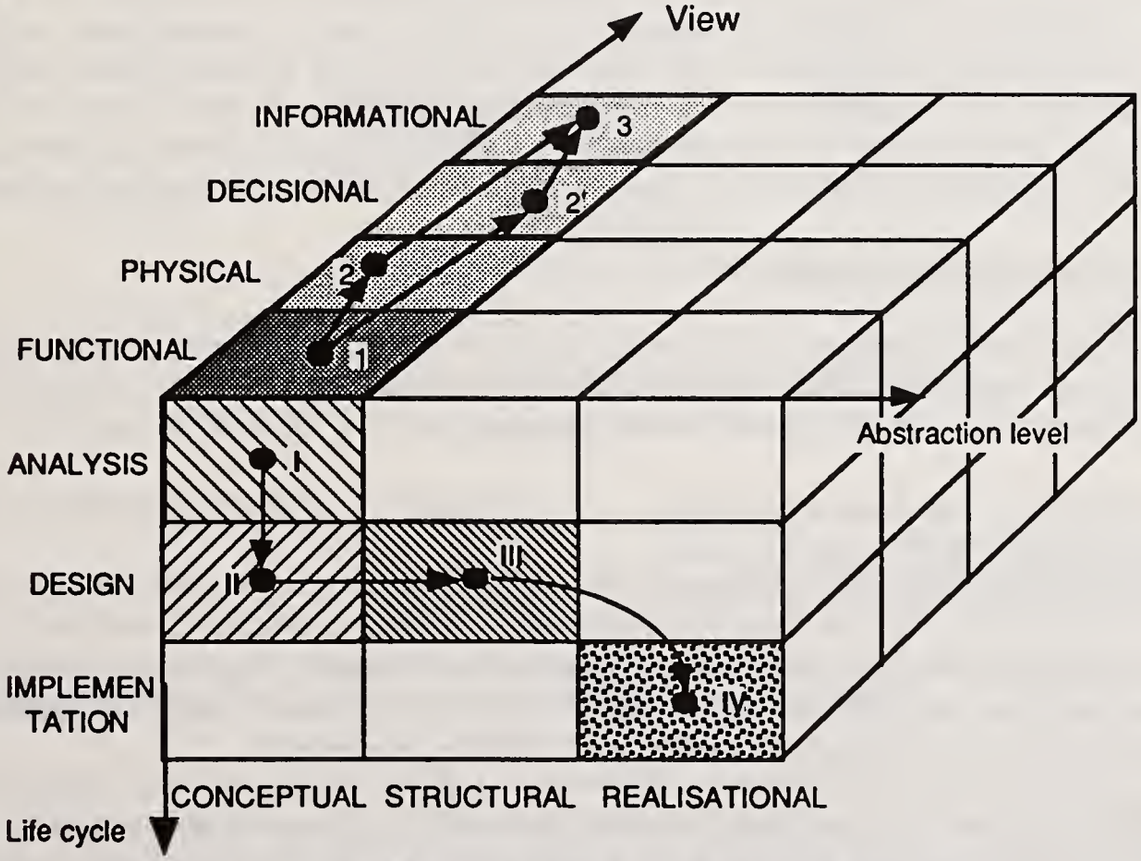


Figure 3. The IMPACS Modelling framework [DOU 90]

A life cycle defines the main phases of a CIM project. It is possible to distinguish three main phases; the analysis, the design and the implementation. A model should be created at the end of each phase. The analysis phase results in a conceptual model of existing or pseudo-existing system. The design phase generates a conceptual target model defining the future system and a structural model, the implementation phase generates an implementation model which is the picture of real CIM system.

The modelling framework can be considered as the skeleton of the architecture and the method. To use this modelling framework in the analysis and design of CIM systems, we have to define explicitely the content of the reference architecture and the method.

3. Architecture

3.1. Architecture definition

There is no generally accepted definition of CIM architecture. For us, a CIM architecture can be considered as a set of models which describe the elements and the relationships between these elements of the whole CIM system. An architecture represents various aspects of the manufacturing and is considered as the basis for design and implementation of CIM systems. Different kinds of models are needed to describe respectively the 'WHAT' (what a CIM system is conceptually composed of), and the 'HOW' (how a CIM system is technically working), and to show the way of transforming the models into realities, e.g. working system [IMP 89].

A reference architecture is a basis of comparison to derive a particular architecture of studied enterprise. In order to represent a wide range of enterprises, a reference architecture should be general and contains the invariant elements and properties of these enterprises. According to the architecture definition given above, a reference architecture is composed of a set of reference models. Each of these reference models stands for representing one point of view of CIM system at a given abstraction level.

3.2. The reference architecture

According to the architecture definition given above, we think it is possible to define three types of reference architectures representing respectively the "WHAT" and the "HOW" of CIM systems as well as the transformation process from the "WHAT" to the "HOW" (Fig.4):

- the conceptual reference architecture
- the structural reference architecture
- the realisational reference architecture

The conceptual reference architecture defines the "WHAT", e.g. what functions, what physical facilities, what decisions and what information we need in a CIM system. These correspond to the four views defined in the modelling framework.

The structural reference architecture shows an "ideal" structure of elements defined in the conceptual reference architecture by taking into account the various criterion and constraints of organization.

The realisational reference architecture is a picture of an "ideal" real CIM system. The realisation reference architecture represents a CIM system in terms of information technology components, manufacturing technology components and organization technology components.

The use of the reference architectures is defined in the method. To be applied to a very large field, the reference architecture can not be very accurate, but generic.

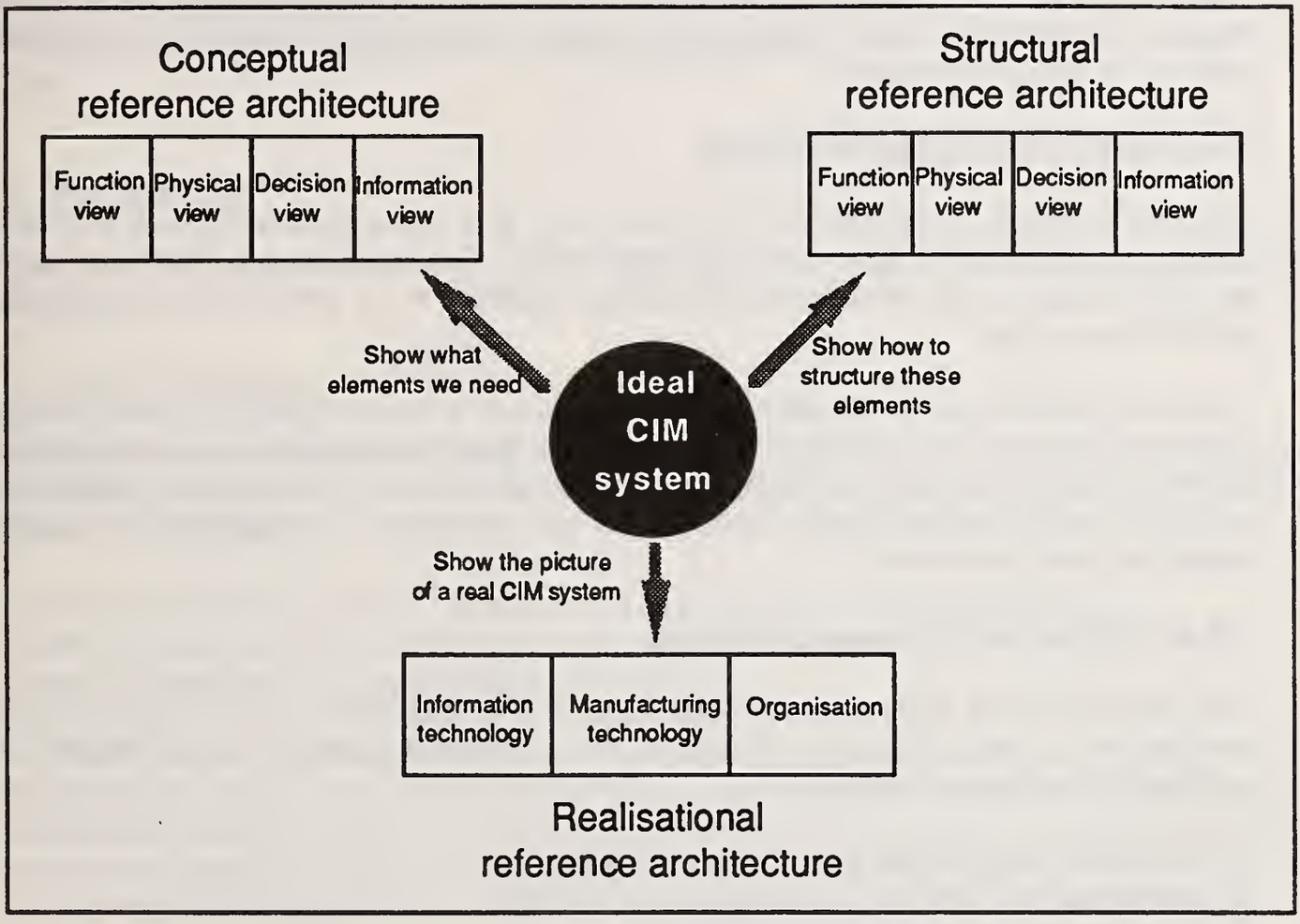


Figure 4. Integrated CIM reference architectures

3.2.1 Conceptual reference architecture

The functional view model describes the total manufacturing system in terms of functions. The 'Business Planning', 'Master Production Scheduling' and 'Factory Coordination' etc... These functional blocks will be detailed successively to a desired degree of detail. The formalism used to build this functional view model is IDEF0.

The physical view model describes the enterprise physical resource, processes, products etc.. The physical view model shows also the location where functions defined in the function model are located. It is important to note that dynamics simulation technique is

very useful to study physical system. We should study how to use this technique in the physical view model. The formalism used to build the physical view model is IDEF0.

The decisional view model represents the structure of decisional system. A decision is made for an horizon of time and should be revised periodically. Decision levels, decision centers and decisional links as well as informational links should be represented in the decisional view model. The formalism used to build the decisional view model is GRAI grid and GRAI nets.

The information view model represents the information needed within a manufacturing system. A conceptual data model will be built. The formalism used for the information view is the Entity/relationship.

3.2.2 Structural reference architecture

The role of structural reference architecture is to give some guidline of how to structure the elements defined in the conceptual architecture. To define such a structure, we must consider various criterion (economic, safety, flexibility etc..) and various constraints of studied enterprise.

The structural reference architecture can not provide a direct solution for each particular enterprise because each enterprise has its own priority of criteria, its own constraints and its own business objectives. But the structural reference architecture can define some possible "ideal" configurations which act as the references to aid people to build their particular structural model.

3.2.3. Realisational reference architecture

The 'Realisational Reference Architecture' is the description of the final physical realization of the computer integrated manufacturing system as a "whole". This architecture contains three parts :

1° Information Technology components [AMI 89]
2° Manufacturing Technology components [AMI 89]
3° Organizational technology components

The Information technology Components are composed of mainly:

- Application softwares
- Computer hardwares
- Communication networks
- Information exchange services
- Local operating systems
- Databases and database management systems
- etc..

The Manufacturing Technology Components are composed of mainly:

- People

• Machines
• Robots
• etc..

The Organizational Technology Components are composed of mainly:

• Rules to structure the Information technology Components
• Rules to structure the Manufacturing technology components
• Rules to define the working of real world CIM system
• Rules to control CIM system
• Rules to modify CIM system structure and components
• etc...

4. Formalisms

Only the formalisms used to build the conceptual architecture are presented in the following.

4.1 IDEF0 Formalism

IDEF0 is used to build the functional model and physical model of the conceptual architecture. Using IDEF0 to build a functional model consists in answering the following specific questions :

• what is the basic functional breakdown or decomposition of manufacturing ?
• what is being transformed and what is the result ? what influnces these functions ?
• what is necessary to carry out these functions ?

An IDEF0 formalism is made up of labeled boxes and arrows. Boxes represent the decomposition of the topic into parts, arrows connect boxes and represent interfaces or constraints between boxes. A control describes the conditions or circumstances that govern the transformation. A mechanism could be the person or device which carries out the activity Fig. 5).

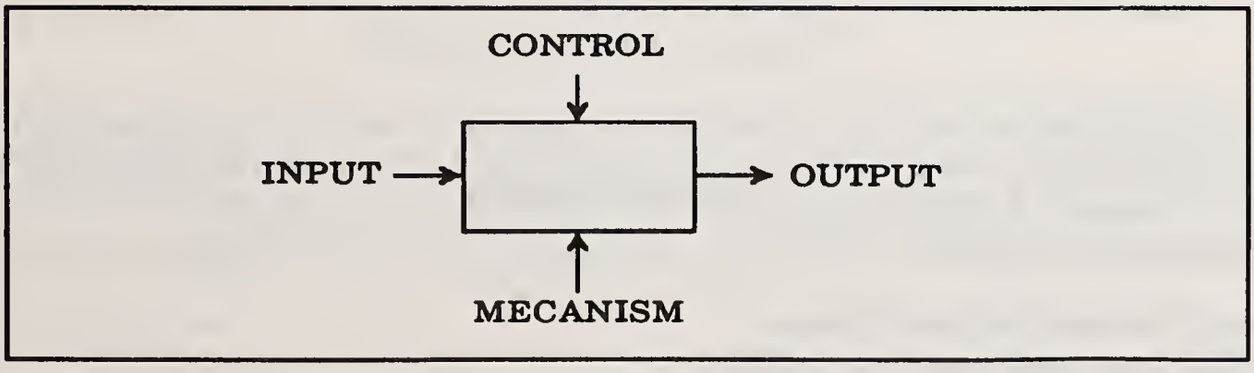


Figure 5. IDEF0 formalism

4.2 GRAI formalisms

GRAI formalisms consist of GRAI grid and GRAI net. GRAI grid was developed within the works of GRAI method [DOU 84]. GRAI net was developed to meet the needs of discrete activity modelling, and particularly decisional activity modelling.

GRAI grid gives a hierarchical representation of a decisional view of CIM systems. GRAI grid is a frame-like table using a functional criterion to identify production management functions, and decision horizon (H) and revision period (P) criterion to identify decision levels. Decision horizon is a time interval through which decisions are valid. Revision period is a time interval at the end of which decisions are revised. The building bricks of the grid are decision centers mutually connected by decision links and information links. A decision link is made of decision variables, decision objectives and decision constraints. A decision link provides a decision frame to another decision center. Information link transmits the decisions made in one decision center to another decision center as information (Fig. 6).

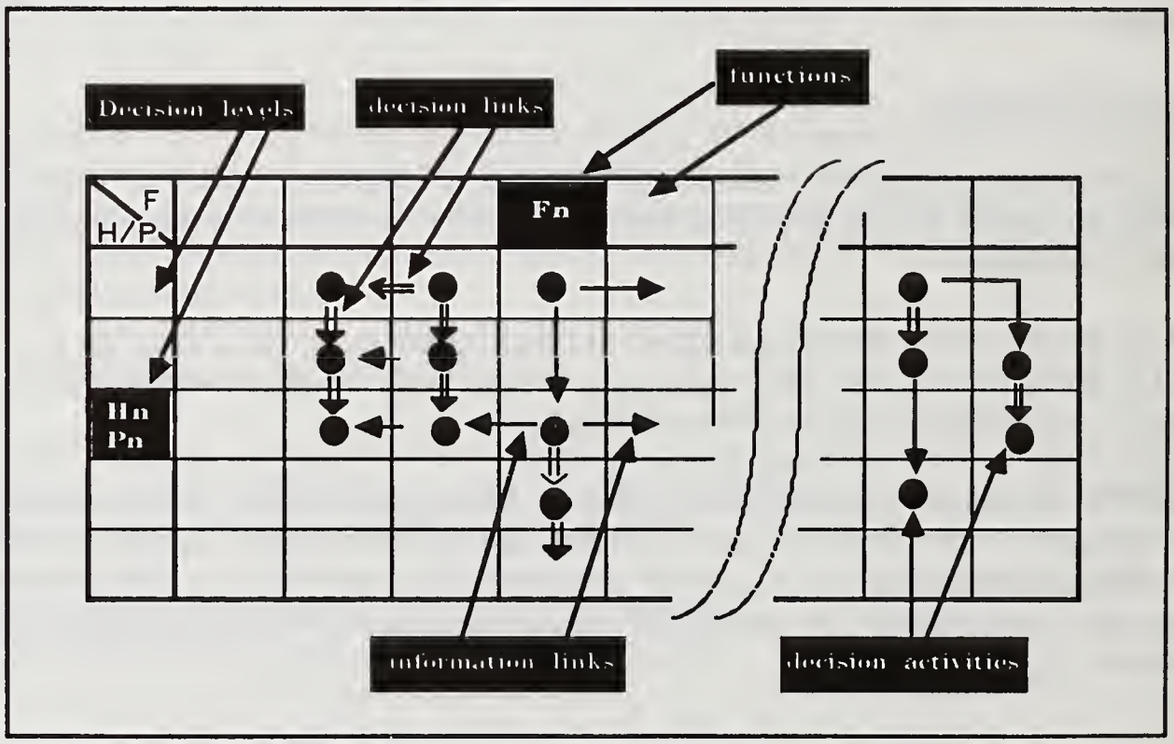


Figure 6 . GRAI grid formalism

GRAI nets give the structure of the various activities in each decision center identified in the GRAI grid. By using GRAI nets, the result of one discret activity can be connnected with the support of another discret activity. With GRAI nets, four fundamental elements are to be identified : (see Fig. 7).

- to do or to decide (activity name),
- initial state (main iuput of an activity),
- supports (informations, decision frame, methods and materials),
- results (results of an activity).

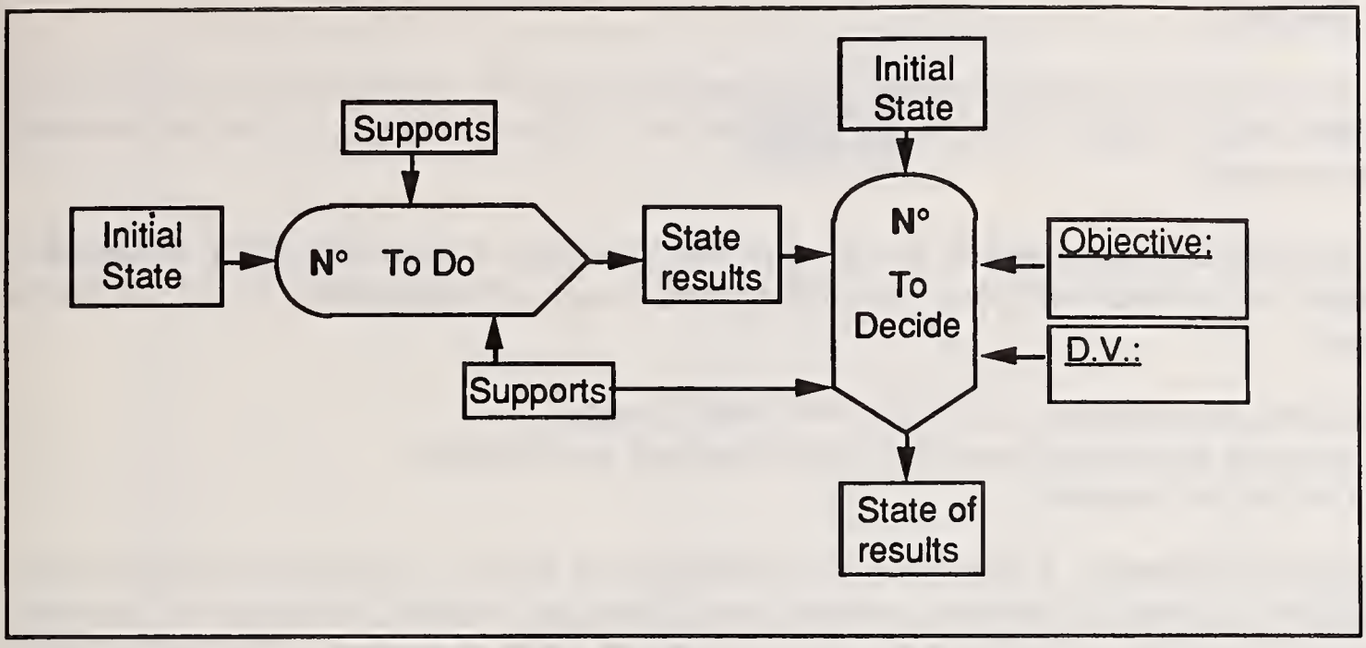


Figure 7. GRAI net formalism

4.3 Entity/relationship formalism

The purpose of information modelling is to structure the memory of the enterprise [TAR 83]. There exist different ways to model an information system, but the most elementary one is to identify information entity by its name, to describe this information by its attributes and then to establish the relationships between them.

An entity is "anything relevant to the enterprise about which information could be or is kept". An entity represents data, it is not itself data. For instance, a drilling machine is an entity but its capability, number of tool, availability and so on are just data.

A relation is "an association between two or more entities". Anything that shows or sharpens a connection between two or more entities may be thought of as a relation.
We can introduce the concept of degree (cardinality) in relationship (Figure 8).

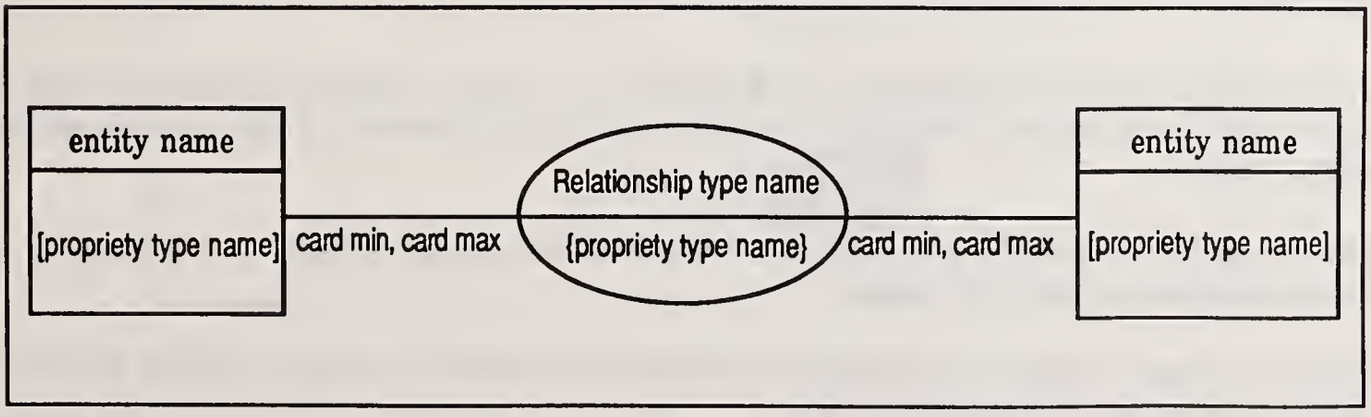


Figure 8. Entity/relationship formalism

5. Method

The method is developed within the architecture context. The role of the method is to define how to derive a particular architecture of studied enterprise from the reference architecture.

A method defines the way of doing something according to some principles. A method to support a CIM architecture for the analysis and design of CIM systems should involve the use of :

- reference architecture with various reference models,
- graphical and computerized formalisms derived from models,
- a structured approach.

Generally speaking, a structured approach defines a set of steps to be followed when applying a method to attack a problem. In a CIM design method, the structured approach should cover the whole life cycle of a CIM project which is splitted up into several steps (analysis, design, development, implementation, operating). For each of these steps, models to be built, decisions to be made, validation and coherence checking to be done between models, must be explicitely defined.

Three main phases are needed: the analysis, the design and the implementation (Fig.9). The existing or pseudo-existing system will be described in the analysis phase by a conceptual model representing the existing function, physical organisation, decision and information of studied enterprise. This conceptual model of existing will be re-arranged and optimized in the conceptual design phase to meet the needs of future system. We call this new model the conceptual target model. Then in the structural design phase, the elements defined in the conceptual target model are structured by a structural model. The structural model contains the detailed specification for the software development. At last we should specify how these softwares, hardwares, machines and people are connected to each other. This is made a realisational model which is a picture of real working CIM system that we will implement.

The structured approach has defined three main phases in a CIM project life cycle: the analysis, design and implementation. In each of these phases, the reference architectures are used to aid to build the particular architecture of studied enterprise (see Fig. 10).

The Conceptual reference architecture is used at the analysis phase and design phase to aid to build an conceptual model of existing system and a conceptual target model of the future system.

The structural reference architecture is used at the design phase to aid to build a structural model of the futur system.

The realisational reference architecture is used at the implementation phase to aid to build a realisational description model.

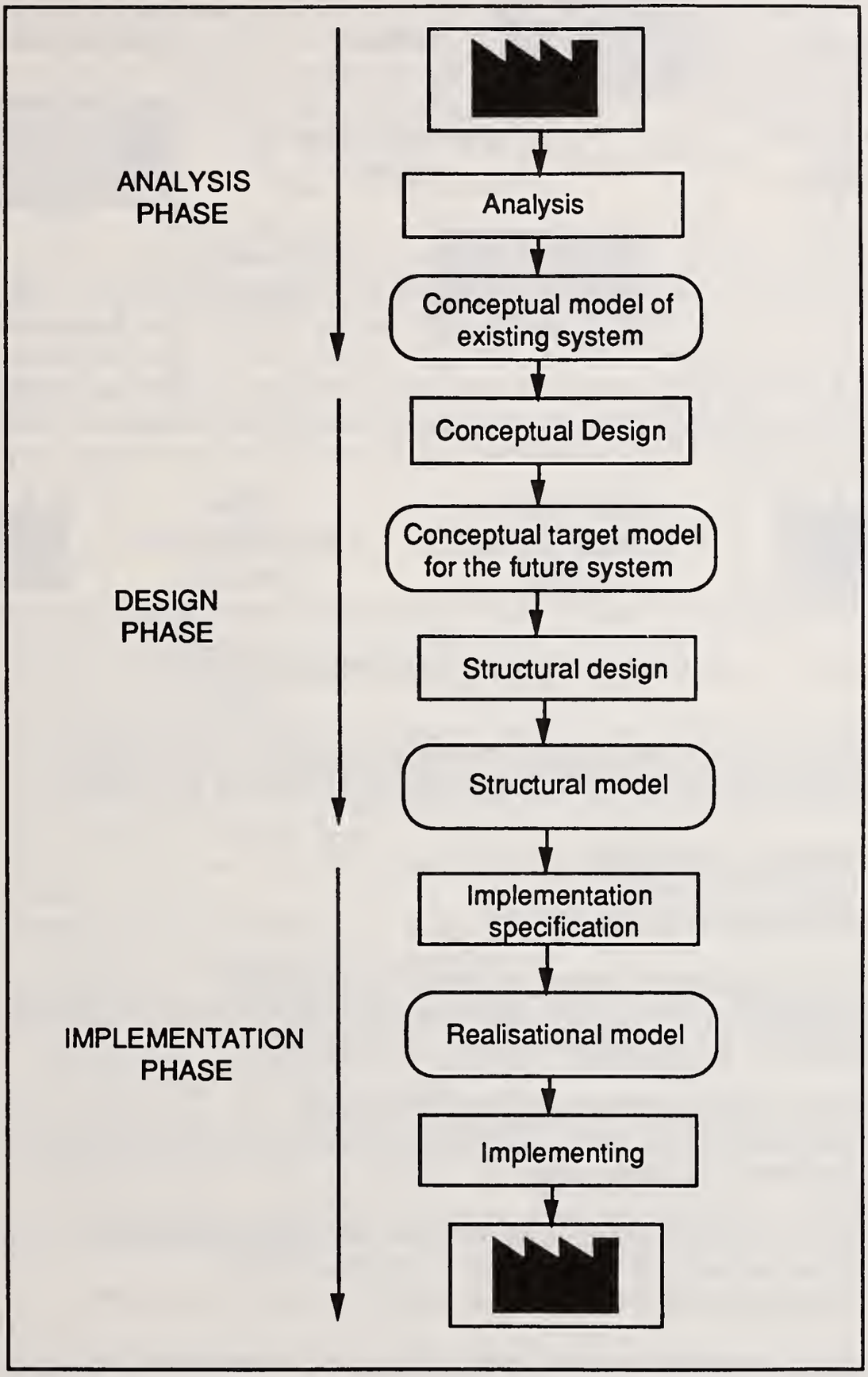


Figure 9. The structured approach

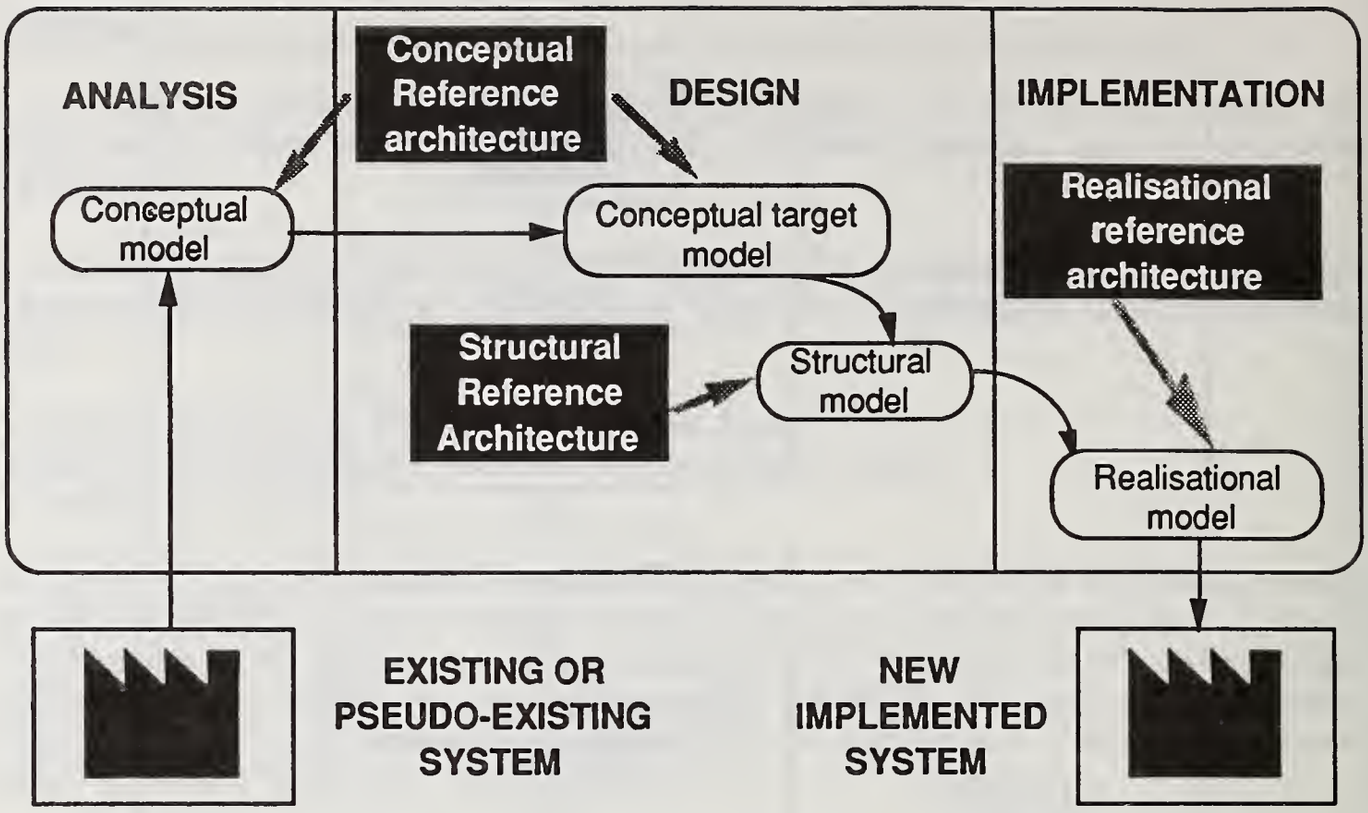


Figure 10. Uses of reference architectures

The three main phases; the analysis, the design and the implementation can be detailed into the following subjects which will be treated and defined in the method:

- How to initialize a CIM project,
- How to organize the project,
- How to determine the studied domain.

- How to do the analysis of existing or pseudo existing system,
- How to build the four view models using the conceptual reference architecture,
- How to check the inconsistencies between these models.

- How to do the conceptual design of the future system,
- How to build the new four models using the conceptual reference architecture,
- How to simulate the design model.

- How to build structural models using structural reference architecture,
- How to make the technical choices and the regroupments,
- How to separate the automatic part and manual part of the futur system.

- How to build realisational models using realisational reference architecture,
- How to implement the realisational models.

The reference architecture contains a set of reference models used by the method. When studying a particular CIM system, the use of reference architecture enable to create the

architecture of studied enterprise more quickly without much inconsistencies. Generally, the studied system corresponds rarely exactly to a reference architecture. So, we have to show how to get the solution by adapting the reference architecture to the specificity of studied system. This adaptation is made according to the difference identified between the reference architecture and the studied system.

6 Conclusion

The reference architecture and the method form an overall solution for CIM. Without reference architecture, method can not be used efficiently. Without method, there is no guideline to show how to build an architecture.
The approach presented above is an ongoing work. It is currently developed in the frame of ESPRIT project IMPACS (Integrated Manufacturing Planning And Control System) in which GRAI is involved. This approach is also coherent with ESPRIT project CIM-OSA (CIM - Open System Architecture) which is considered a good reference in the field of CIM.

7. References

[AMI 89] AMICE - "Open system Architecture for CIM" - Esprit Project 688 research report. vol.1. Springer-Verlag 1989.
[CAM 80] CAM.I - "Architect's manual: ICAM definition method IDEF0". - CAM.I document N° DR-80-ATPC-01. April 1980.
[DOU 84] DOUMEINGTS G. - "Méthode GRAI: Méthode de conception des systèmes de productique". - Thèse d'état en Automatique. Université de Bordeaux 1. 1984.
[DOU 90] DOUMEINGTS G. - " Modelling Techniques for CIM". - ESPRIT CIM workshop proceedings. Brussels, 7-8 March 1990.
[IMP 89] IMPACS - " Review of Existing CIM architectures, methods and tools". - IMPACS WP1 deliverable. 1989.
[TAR 83] TARDIEU H., ROCHFELD A. & COLLETTI R. - "La méthode MERISE, Principes et outils". - Les éditions d'organization. Paris 1983.
[VAL 89] VALLESPIR B., DOUMEINGTS G. & ZANETTIN M.- "Proposals for an integrated approach to model and design manufacturing systems: the GRAI Integrated Method". In the third International Conference on Computer Applications in Production and Engineering, Tokyo, Japan, 2-5 Octobre 1989. 11 p.
[VAL 90] VALLESPIR B. -"Modélisation et conception des systèmes de production". CNRS/G.R. Automatique/SED./GT4 - Intégration. 1990.
[ZAN 89] ZANETTIN M., VALLESPIR B. & DOUMEINGTS G. - "Elaboration of specification for the CMS according to the results of the integration". - deliverable D40. Esprit project 418. Feburary 1989.

Towards a Distributed Control Architecture for CIM

Matt Johnson and James R. Kirkley III

Global market pressures have reshaped the enterprise. Both internal competition and external collaboration are common, and the concept of the independent company has become obsolete. Meanwhile, advances in distributed computing technology have broken the boundaries between information systems. In this environment, CIM architectures that view manufacturers as hierarchically controlled entities fail to provide an adequate framework for development. We propose the development of a technical architecture based upon the principle of ***distributed control*** *to remedy the discrepancy between contemporary business practices and existing CIM models, and demonstrate its advantages in enterprise application development.*

1 Introduction

The structure of effective information systems for manufacturing follows expectations set by the information and procedures of the businesses themselves. As business practice and computer technology evolve, it is important to reevaluate the assumptions of existing architectures to determine whether they continue to serve the manufacturing mission. In particular, the control model of a CIM architecture has a major impact on the structure of applications, so it is a good starting point for analysis.

1.1 The Legacy of Hierarchical Control

Conventional technical (or implementation) architectures for CIM have selected hierarchical control as the best way to respond to the requirements of top-down business models. Most resemble the NBS reference model of manufacturing [NBS], which divides application functions among five levels of responsibility in a centrally managed hierarchy:

- *Facility (Plant):* CAD, group technology, process planning
- *Shop (Area):* Order splitting, release, and tracking, resource allocation/preventive maintenance
- *Cell*: Job sequencing and material handling
- *Workstation:* Machine coordination and control
- *Equipment:* Programmable devices

These models have been instrumental in establishing a common vocabulary and expectations for systems integrators. In fact, they have spawned an entire industry, in which vendors can build, position and sell generic workcell controllers, area controllers, and various production management applications. But they have also lead to monolithic, difficult-to-distribute plan-

ning systems, and underutilized processing capacity at the shop floor and device levels. Even though systems installed in the factory today are often as sophisticated as any others in the enterprise, they do not have adequate information or authority to respond to changing conditions within their scope of operations.

1.2 The Opportunity of Distributed Control

The principle of *distributed control* provides a more realistic model of the existing complexity and future diversity of enterprise information systems. In this scheme, an open set of peer entities contract with one another to accomplish business tasks, and compete for distributed resources.

A distributed control model is more general than a hierarchical one, in that cooperating entities may be aggregated and encapsulated recursively into hierarchical levels of control (as illustrated in Figure 1). However, they do not *impose* a hierarchy, as top-down models do. Using distributed control, it is possible to construct flatter organizational structures, in which (for example) a shop floor cell might generate an order for materials from an external supplier itself, instead of relying on a centralized MRP system to determine its needs.

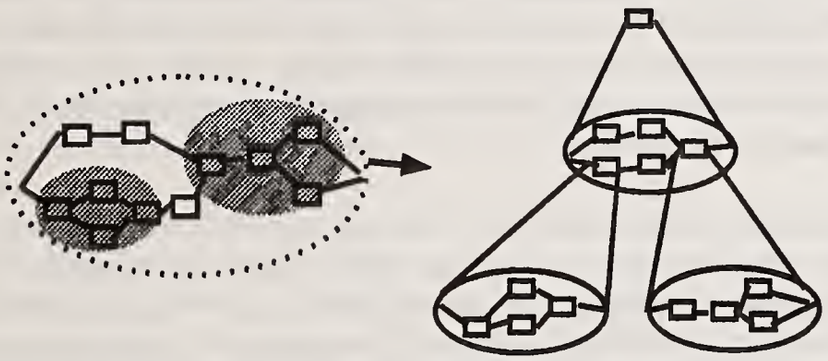


Figure 1. Constructing a Hierarchy from Distributed Control Elements

The distributed control concept takes advantage of the similarity of the manufacturing problem at all organizational levels. While planning horizons and control mechanisms vary greatly throughout an enterprise, every intelligent manufacturing entity must accept external or self-generated orders for a product, and then schedule and dispatch activities that transform material (as defined by a manufacturing process) into that product. The basic abstractions and the relations between them are as true for a corporate-wide MRP system as they are for a single robot loading parts onto an assembly line.

2 Characteristics of a Distributed Control Architecture

A distributed control model requires a new set of goals, benefits, and design criteria for CIM architecture. It also gives developers the tools to identify the scope of manufacturing applications in the enterprise domain, and to sketch out some basic abstractions.

2.1 Goals

As the CIM model evolves from hierarchical to distributed control, the technical architectural focus changes. Instead of determining the best way to decompose applications into a rational set of functions, a distributed control architecture promotes the interoperability of independent CIM applications defined in terms of objects.

Functional decompositions constrain interoperability to one dimension -- an application can provide only the range of functions allowed it in the hierarchy. The goal of a CIM distributed control model is to expand the flexibility of interoperation, while maintaining discipline. It capitalizes on the basic similarity of the characteristics and behaviors of production orders, operations, resources, and activities throughout the enterprise, so that independently developed applications that manipulate these production entities can interact in new ways.

2.2 Benefits

In a distributed control environment, application providers "plug and play" by offering sophisticated features behind industry-standard component interfaces. An algorithm and its associated data, which once may have been internal to a large application, now may be repackaged and used in other contexts. For example, a statistical quality control vendor might sell a package that used advanced algorithms (developed for factory-wide systems) to analyze local production in a shop floor vendor's cell. By specializing, the quality control vendor could support additional, realtime features, while still providing a standard interface for applications to retrieve quality information throughout the enterprise.

The interoperability achieved using objects improves upon the current practice of using shared data to integrate applications, because it presents information in behavioral as well as structural terms. Formal methods define the appropriate use for each piece of information, so that access to it is controlled. They hide the internal format of manufacturing data (which may vary over time and according to implementation) behind standard interfaces, and draw strict boundaries of data ownership with access synchronization.

Implementation framework developers should be able to select the off-the-shelf application components that best suit their corporate needs, with the assurance that they will work together. Application vendors should be able to concentrate on developing *application* features instead of base technology, and compete on the basis of the unique added value they can offer to manufacturers. The intent of a CIM application interoperability architecture is to make this vision a reality.

2.3 Design Criteria

The selection of a control model is only a starting place. To realize true application interoperability in a heterogeneous environment, a CIM technical architecture must be based on a thorough analysis of business requirements, and developed according to rigorous design criteria. The following sections identify some design goals to be considered during development.

2.3.1 Incorporate Existing Standards

In many of the areas that a CIM technical architecture covers, CIM standards activities are already well established. The Product Structure Configuration Management (PSCM) portion of the STEP/PDES [STEP] draft proposal specifies how product definition data may be structured to support manufacturing applications, such as bills of materials. EDIFACT [EDI87] and related electronic data interchange standards provide a basis for the exchange of production orders. MMS [MMS90] standardizes the protocol between factory applications and intelligent manufacturing devices. An architecture should incorporate these standards to the greatest extent possible; it should also catalyze standardization activity where none currently exists. Ideally, a CIM technical architecture would simply be a *standards profile*, which would guide implementation framework developers in the application of a set of encompassing international standards.

Objects offer a way to encapsulate the features of existing, standards and applications. For example, the notion of a "product" class may be used to represent the features of a PDES Product Model in object-oriented terms, so they may be imported into the architecture.

2.3.2 Use Object-Oriented Modeling Methodology

The emphasis on modularity, genericity, and formalism in the object-oriented paradigm makes reusable, interoperable applications feasible. Through the mechanism of inheritance, developers can trace the most abstract architectural concepts -- through products -- to unique implementations. They can also select the appropriate components of an architecture for their business environment, and apply the discipline for further development in areas that are unique to their organization.

The choice of object-oriented modeling methodology does not imply that implementations must be object-oriented, however. A CIM architecture can use objects as a conceptual aid, as many ISO OSI standards do. For example, MMS (at the OSI Application layer) defines a Virtual Manufacturing Device object as an abstract class, with standard component classes such as variables, domains, and events. Representatives of NC, PLC, robot, and other device types specify how they inherit the features of the VMD by developing companion standards. But even though the standards identify concepts in terms of objects, implementors may use any means they prefer to conform to the MMS service protocol (and its companion standard extensions).

2.3.3 Isolate Policy Decisions

Another key requirement for a CIM architecture that serves existing, heterogeneous enterprises is that it must isolate policy information. It must not depend upon a single manufacturing discipline, such as Just-in-Time or OPT. As effective as any policy may be, an architecture cannot hope to eliminate other, established business practices. Change in an enterprise is evolutionary, and multiple, hybrid policies often prevail.

One example could be termed "Just In Case" production, in which a company strives to implement JIT techniques, but keeps a small amount of inventory on hand *just in case* the materials for the next task fail to arrive on time. Here, both pull and push styles of production are in use,

and a system that recognizes only one may lose orders, miss deadlines, or build unneeded products.

Policy decisions occur everywhere throughout manufacturing. Under different conditions, such conflicting factors as the lowest production cost, highest machine loading, or fastest delivery may take precedence. The moral is that a technical architecture must allow manufacturers to put complex policies into place, and to change them rapidly as business conditions change.

2.3.4 Minimize Base Technology Constraints

Just as it must isolate policy decisions to remain general, a technical architecture should strive to minimize base technology constraints. It must yield major benefits without requiring that existing information systems be thrown away.

By setting application interoperability as its goal, a distributed control architecture enforces the semantic consistency of independently developed applications. Semantic mismatches are at the heart of most integration problems; once they are solved, protocol, database, or language conversions are straightforward. Of course, future technologies (such as distributed object management systems) will make the advantages of a distributed control architecture even more accessible, but object-oriented architecture principles can be applied in conventional implementations by making the network protocol the point at which formal methods are defined.

2.4 Scope

Figure 2 positions manufacturing in the overall framework of the enterprise. A CIM technical architecture must encompass the domains currently represented by MRP II, shop floor control, quality management, and portions of concurrent engineering applications, while it establishes clear interfaces with finance, warehousing and distribution, product definition data, and manufacturing devices.

There is a tremendous temptation for CIM architects to tackle topics that belong in the generic, enterprise application domain. Unfortunately, CIM experts are in no better position to specify the application components that are common to all application areas than office automation, finance, CAD, laboratory, and other vertical application experts would be. If CIM architects define aspects of the base technology as well as their own, integration across vertical application domains will remain an ad-hoc process.

An enterprise application architecture will eventually be fundamental to integrating information systems that cross traditional application boundaries, but it must be influenced by the detailed requirements of a spectrum of existing vertical application architectures (such as CIM). Standards efforts and consortia (such as ODP, IEEE 1003 [POSIX], X.Open, and OSF) have already begun work in the generic application architecture arena; CIM architects should set requirements for these groups, but rely on them to supply more general application features.

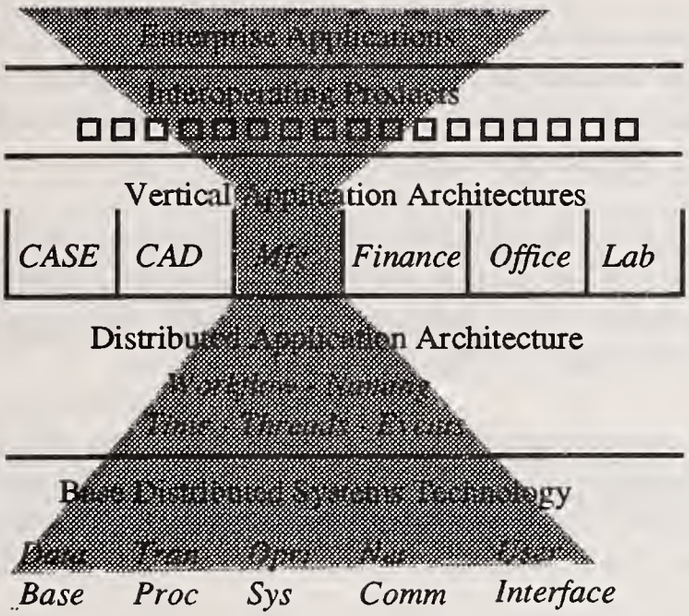


Figure 2. CIM in the Enterprise Domain

2.5 Basic Abstractions

Even after drawing strict boundaries, CIM architects still face a broad application domain. Developing a viable technical architecture will demand the effort and agreement of the industry as a whole. Nonetheless, it is important to illustrate the kinds of objects a distributed control architecture would contain, and step through some of their interactions. The following is an outline, by object category, of a draft library of CIM object classes.

2.5.1 Production Resources

A production resource is any physical entity involved in production operations. There are several general subclasses of resources, including *material, operators,* and *equipment.* Material describes any object of production operations that yields a product. Equipment provides a logical view of production facilities (as opposed to the communications view, standardized by the Manufacturing Message Service [ISO 9506]). Operators are production personnel; they inherit additional characteristics from other superclasses (for example, *employee*) unrelated to CIM.

Some resources are schedulable. In certain manufacturing environments, schedules are associated with equipment; in others, they are linked to the product, or a combination of resource subclasses.

Equipment and operators export a set of capabilities, which define the types of jobs that they can perform. A distributed control architecture allows binding of a resource to a production operation at any point in time, from the initial definition of the production operation, up to the moment the resource is required. During production, these resources match their capabilities to the requirements of production operations through an intermediary called a *role.* If they are

also able to establish contracts through an order protocol, resources are termed *producers.*

Material differs from other resource subclasses in that it follows a product data description, instead of exporting production capabilities. The STEP/PDES development effort [STEP] provides standardized representations of product data, which may be brought into the object model in the form of a *product class* for each instance.

Some resource subclasses have hybrid characteristics. In metalcutting applications, for example, the tools (blades and bits) used to machine parts are consumed like material, but function as a part of the machinery. Operators order replacements regularly (using product descriptions), but tool consumption is not tied directly to the orders being produced.

Resources also provide a model that statistical quality management and capacity planning applications can exploit in the collection of production (historical) data. Production planners may substitute physical resources with virtual counterparts, and run detailed simulations of processes to estimate material cost, resource loading, or production time for orders (following the suggestion of [PAN89]).

2.5.2 Production Orders

Orders are the means for one producer to request delivery of a quantity of goods or services from another at a specified time. In a distributed control architecture, the exchange of orders within an enterprise follows the same formal protocol as external orders do. This genericity of orders supports flexible environments, such as one in which a shop floor cell requests parts directly from an external supplier. EDI standards such as EDIFACT provide a protocol for orders, and an object-oriented architecture can map order class methods to EDI messages.

Orders fall into several subclasses. *Workorders* are simple production requests, without a monetary basis. *Purchase orders* associate delivery of products with an exchange of funds. *Forecasts*, or "soft" orders, assist capacity planners by projecting the arrival of "firm" orders (workorders or purchase orders).

The dialog between producers may begin with a request-for-quote (RFQ) message. A producer that wishes to bid on the production request may generate a response (an RRFQ), stating its terms and conditions. Depending on its policies, a producer may then update its production forecast to reflect the probability of getting the job, and allocate resources in anticipation of the order.

A producer that responds to an RFQ establishes a *contract* with the requester. If it receives an order for the work within the scope of the terms it set, it should be able to fill it. On the requester side, the placement of an order carries a similar commitment: to pay for the product (in the case of a purchase order), or make use of it (in the case of workorders).

Contracts may be seen as long-running, distributed transactions, supported by two-phase commit protocol standards for heterogeneous environments (such as [CCR]). Researchers continue to develop more sophisticated serial transaction models to manage long-running activities.

Other methods related to orders include order acknowledgment, status requests, and cancellation.

2.5.3 Production Schedules

A production schedule (a component of some classes of resources) consists of an ordered set of planned activities with specific starting times, ending times, and durations. When a resource is scheduled, the planned activities represent the allocation of the resource to future work. Depending upon the scheduling policy in effect, a schedule dispatches activities that correspond to its plans, causing production operations to be executed.

2.5.4 Production Operations

Operations are abstract representations of programs and procedures that transform material during production. Production activities (threads of control) execute operations to produce products.

The recursive technique of building hierarchies out of aggregations of objects (as illustrated earlier) is particularly valuable for composing complex operations from smaller, atomic steps. For this purpose, we define a *production process*: a directed graph with production operations as edges, and production states as nodes. Use of the abstract type saves others from having to be concerned whether an operation is implemented as a process or as a single step.

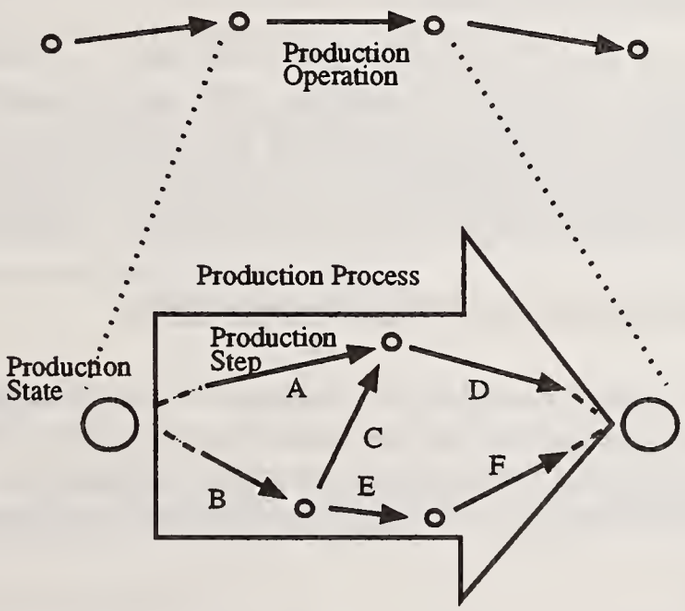


Figure 3. Operations, Processes, and Steps

Figure 4 illustrates the relationship of production operations, processes, and steps. In their network topology, production processes can represent assembly, disassembly, transformation, creational, and destructive steps. They also support highly flexible environments (a key architectural requirement) by defining alternative sets of steps between beginning and concluding states.

Operations establish a set of requirements for producers that may execute them. However, these are stated in process terms, which are very different from the terms producers use to advertise their capabilities. For this reason, an intermediary, called a role, acts as a broker in the resource-selection process, allowing resources with widely different sets of capabilities (such as a production worker and an FMS) to subscribe to the completion of the same task (such as drilling a 2cm hole in a car body).

2.5.5 Production Activities

A production activity represents the thread of control that a resource dedicates to a production operation. Some resources can support multiple, concurrent operations; others handle only one task at a time.

Most production activities occur under the terms of a contract established by an order, so they must be highly reliable. A nested transactional model (which provides commit or rollback as alternatives for activity completion) is appealing, but inadequate because:

- Typically many long-running activities execute at the same time; each of these may be nested to several levels of depth. The requirement that all transactions be atomic and independent leads to an unsupportable amount of locking and contention for resources.
- Many manufacturing steps cannot be rolled back in the same sense that database updates can. Once a part is machined, it remains that way; it must be assigned to another order or scrapped if the activity that created it is aborted.

New transactional models such as *sagas* [SAGA87] apply much better to production activity control. They relax the strict ACID[1] criteria of transactions, allowing subtransactions to commit before subsequent ones begin to execute. Compensating steps for each subtransaction are executed if the master transaction (called a saga) needs to "roll back" to a consistent state. The compensations must undo the effects of their associated steps, as defined in application terms.

3 Applying the Architecture in CIM Implementations

Even after an architecture is complete, individual implementations must interpret and apply it effectively. The implementation process includes adaption (the specialization of the generic architectural abstractions to the particular needs of an organzation), followed by a selection of available products and applications that can be assembled together into a working CIM system.

3.1 The Adaption Process

A danger of CIM architecture is that it can become overly schematic in its attempts to be both specific and comprehensive. For example, CIM-OSA [DIN89] suggests that a set of specialized "partial" models be developed for manufacturers along three dimensions: industry segment, enterprise size, and type of product. This classification approach fails because the set of determining business factors is open, and not architecturally definable. Politics, corporate culture, the age of the enterprise, and the level of distribution of operations are just some of the

[1] ACID: Atomic, Commutative, Isolated, and Durable.

host of points that may affect the application of an architecture as much as business size or product type.

For adaption purposes, an object-oriented model serves well. Its class interfaces are modular, making selection and substitution simple. Further, inheritance offers a way to specialize both the methods and data of the architecture for individual applications. An object-oriented architecture serves innovation, by providing avenues for improvement. It does not predetermine how a business should be run.

3.2 The Software Selection Process

The selection or development of software platforms for CIM is an implementation framework decision. Here, considerations such as portability become relevant. Corporations may choose to standardize on portable or non-portable platforms; in all cases, if the implementations follow the semantics of the technical architecture, they will interoperate. Portability does not imply freedom from base technology constraints; in fact, it imposes a set of implementation constraints. Using portability (an implementation concern) to achieve interoperability (a technical architecture goal) is a mistake.

4 Conclusion

As CIM architecture enters the realm of international standards, all of its traditional assumptions bear reexamination and further development. Simply by generalizing one fundamental principle (hierarchical control), we have shown how a distributed control architecture with different goals, benefits, and structure could be conceived.

Nonetheless, the architectural concepts presented here are only suggestions. They are meant to help stimulate a serious analysis of technical requirements for CIM architecture, which must be public and broad-based. Past architectures have gone unused when they have not respond adequately to manufacturers' requirements; future investors in CIM cannot afford to have that happen in our ever-more-volatile business climate.

5 References

[DIN89] *DIN-NAM 96.5 Proposal: Framework for CIM System Integration*, ISO TC184/SC5WG1-N105, 04-15-89.

[EDI87] ISO IS 9735, EDIFACT.

[STEP] ISO TC184/SC4, Standard for the Exchange of Product Model Data, First Working Draft Proposal for STEP 1.0.

[MMS90] ISO IS 9506, *Manufacturing Message Service*.

[NBS] CIM Reference Model, National Bureau of Standards.

[PAN89] Jeff Y. -C. Pan, Jay M. Tenebaum, Jay Glicksman, "A Framework for Knowledge-Based Computer Integrated Manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, Vol 2, No 2, May 1989.

[SAGA87] Hector Garcia-Molina and Kenneth Salem, "Sagas", *SIGMOD'87 Proceed ings*, May 27-29, 1987, San Francisco, California, ACM Press.

CIM-OSA - A VENDOR INDEPENDENT CIM ARCHITECTURE.

Presented by RICHARD PANSE on behalf of the ESPRIT AMICE CONSORTIUM.

Abstract.

CIM-OSA is a strategic architecture supporting all phases of a CIM system life cycle from requirements definition, through design specification, implementation description and execution of the daily enterprise operation.
Standardised modelling constructs enable generation of particular enterprise models for analysis, improvement and simulation of the daily enterprise operation.
Standardised Information Technology services and protocols within the CIM system environment enable execution of the daily enterprise operations under control of the above derived models.
A brief discussion of the necessary international standards activities provides an outlook of the future work.

1 Introduction.

Computer Integrated Manufacturing (CIM), some years ago only known to real manufacturing experts, today is a well known and often used term. However, even today, the meaning of CIM is controversially interpreted. The ideas vary from "catchword without meaning" through "humanless plant". Despite of this controversial interpretation every enterprise intending to keep its position at the market must use all offered possibilities to keep its products and operation competitive. Amongst other factors like labour costs, technology and education of employees, CIM is one of the most promising possibilities to stay competitive.

To introduce CIM effectively skilled people, knowhow and structured implementation plans are required - in addition to the necessary investment capital and the top management's support for CIM.

An enterprise-wide agreed upon strategy for the introduction of CIM is a mandatory prerequisite if all steps towards CIM are to be successful. Regardless whether the introduction of CIM shall be accomplished slow or fast, regardless which part of the enterprise is the forerunner in the introduction of CIM, and despite the resistance of some of the people involved, a CIM model for the whole enterprise must be generated with engineering methods which support all steps of CIM introduction - from the very first concept down to the real implementation.

Within ESPRIT (European Strategic Program for Research and Development in Information Technology) one project for CIM architecture was initiated and performed by the AMICE consortium (AMICE is the reversed acronym for European CIM Architecture). The consortium consists of the following 21 European members:

AEG Germany; Aerospaciale France; Alcatel France; APT Netherlands BV Holland; British Aerospace United Kingdom; Bull France; Cap Sesa Innovation France; Digital Equipment Germany; Dornier Germany; FIAT Italy; GEC United Kingdom; Hewlett-Packard France; IBM Germany; ICL United Kingdom; Italsiel Italy; Philips Netherlands; Procos Denmark; SEIAF Italy; Siemens Germany; Volkswagen Germany; WZL-Aachen Germany.

The goal of the AMICE consortium is to develop a CIM Architecture which supports the following requirements:

- Timely availability of the right information at the right place.
- Adaptability to the continuous change of the environment and the production processes.
- Flexibility of all enterprise processes and organisational structures.
- Processable descriptions of function and behaviour of all activities within the enterprise.
- Real time control of all enterprise processes.
- Most economic use of information technology.
- Possibility to make use of programs and machines from different vendors.

2 "CIM - Open Systems Architecture" ("CIM-OSA").

The "CIM - Open Systems Architecture" ("CIM-OSA") developed by the ESPRIT consortium AMICE addresses the goals described above.

"CIM-OSA" contains two major parts. Each part by itself is a closed subject and standardizable, however the two parts belong together, and therefore both parts must always be seen in context to each other (figure 1).
The first part of "CIM-OSA" contains concepts for generating information technology representations of enterprise models. Models which, at their ultimate stage, can be used to control and monitor the execution of the enterprise's daily operations. The models can be generated for existing enterprises as well as for new enterprises. This means in turn that "CIM-OSA" can also be used to derive the model of the ultimate enterprise from a model of the existing enterprise (the derivation from "as is" to "to be").
The second part of "CIM-OSA" contains concepts for an Integrating Infrastructure within an Information Technology Environment. The Integrating Infrastructure enables "CIM-OSA" enterprise model occurences to be used for the execution,

monitoring and controlling of the daily operations of the modelled enterprise.

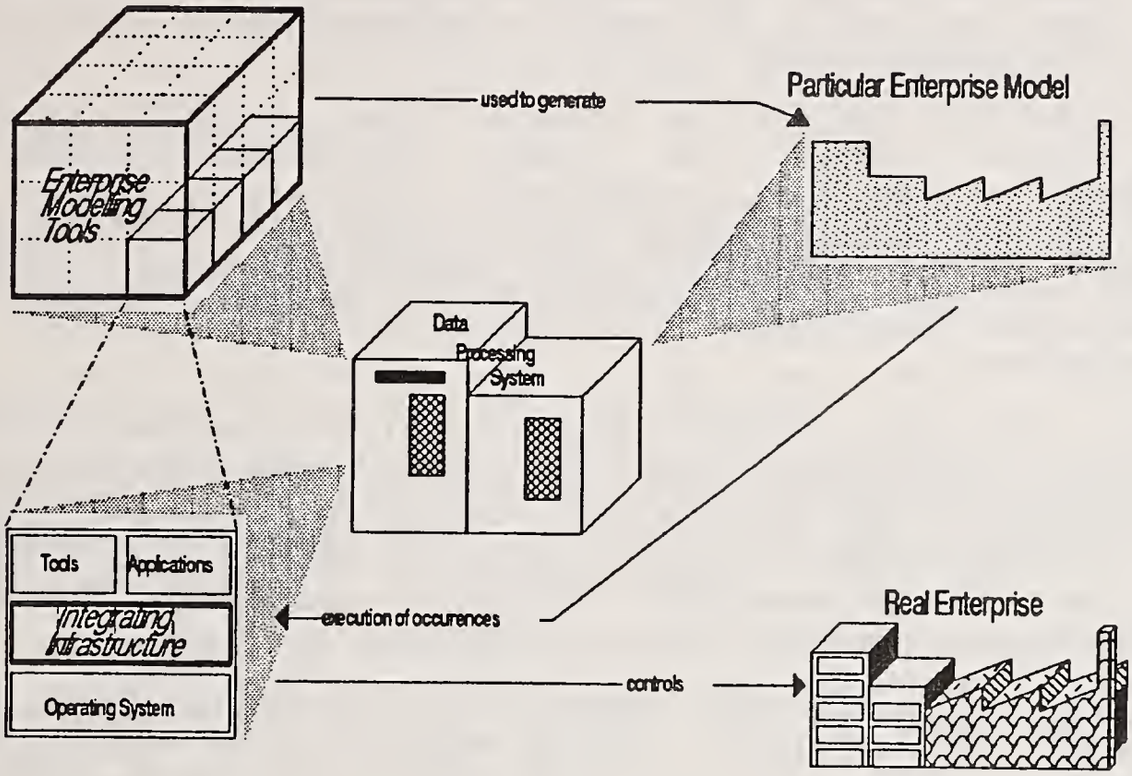


Figure 1. Parts of "CIM-OSA".

2.1 The modelling framework.

Modelling methods like IDEF/SADT, PSL/PSA are available and commercially used since years. All of the existing modelling methods can well be used to describe the material and data flow within enterprises, with different levels of completeness and consistency verification concerning functions and data. None of the methods available today can be used to describe the enterprise's operation down to a level which allows flexible use of resources, or the use of the model to control the daily operation of the enterprise.

The modelling concepts of "CIM-OSA" enable:

Verification of completeness and consistency for all described functions and objects (business processes, data, materials and resources including tools and fixtures) at any detailing level.

Simulation of the enterprise model at any detailing level.

Easy and fast change of the model in case of changing business processes, methods or tools.

The use of the model to initiate, monitor and control the execution of the enterprise's daily operation.

Resource allocation during the execution of business processes to enable better and more flexible load distribution on the enterprise's resources.
Model generation for existing enterprises as well as for enterprises to be built.

The modelling approach of "CIM-OSA" also enables portability of partial models into other enterprises by use of standardised modelling constructs. This will reduce the modelling effort for particular enterprises substantially. Portable partial models will also provide input for a common, standardizable data dictionary, enabling exchange of unambiguous data between enterprises and functional areas of enterprises.

Within "CIM-OSA", function and behaviour of business processes as well as data definitions are separated from the application programs, causing "CIM-OSA" compatible application programs to be limited to the algorithms of todays application programs. Thus "CIM-OSA" application programs are smaller, less complex and less change-sensitive, reducing considerably the ever increasing maintenance cost of application programs.

Due to the structured approach of enterprise modelling, redundant data elements will be avoided. This enables real integrated data bases and prevents unnecessary multiple appearance of the same data as well as working with obsolete data.

2.1.1 Dimension of architectural genericity.

In today's programming environment no responsible application programmer intends to program all components by himself. Instead he will make use of programming languages and software packages available at the market to generate the program support for his particular enterprise. Similarly "CIM-OSA" provides for a "language" and ready for use "partial models" for the enterprise modeller to generate the model of his particular enterprise. The "language", the "partial models" and the "particular enterprise models" are levels of architectural genericity (figure 2).

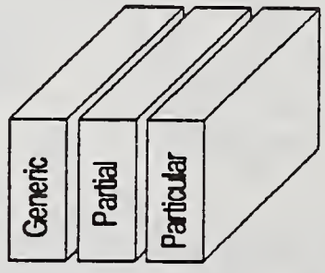


Figure 2. "CIM-OSA" Levels of Architectural Genericity.

Generic Architecture Level. The first architectural level of genericity contains generic constructs (building blocks).

Similar to a programming language which is used to write program modules and/or total programs, these constructs can be used to generate models of major business functions and/or complete enterprises.

Like a standardized programming language which ensures portability of programs within heterogeneous systems, the generic constructs should be standardized to ensure portability of models and/or sub-models.

The Generic Architecture Level is the domain of the standards committees and associated research projects.

Partial Architecture Level. The second architectural level of genericity contains partial models. Similar to application programs, which can be used in many enterprises, partial models can be used in many enterprises.

Many enterprises of the same industry make use of the same or only slightly different major business functions. Therefore industry-oriented partial models can be used in other enterprises of the same industry, reducing the modelling effort for particular enterprises.

Also within different industries similar task oriented business functions are used, like purchasing activities, storage control, change control of parts. Portability of task-oriented partial models to other enterprises enable multiple use of models with little or no additional modelling effort.

Similar to application program which can be used in several enterprises, partial models can have options, which can be used or ignored. This opens the possibility for the wide use of partial models in different enterprises and industries.

A storage model may, for example, contain options for simple manual controlled storages as well as large automatic controlled storages, with manual as well as machine supported loading and unloading facilities, thus offering a storage (partial) model for wide use.

The standardized constructs made available by the generic architectural level of genericity will be used to compose the partial models, thus ensuring that all partial models are compatible in view of the constructs used.

In order to avoid that an immeasurable amount of partial models will flood the market, reference models for the different industries should be developed.

The Partial Architecture Level is the domain of organisations (institutes, software companies, enterprise consultants) which will generate and market partial models. These organisations may also provide significant input to the standardisation bodies being active at the Generic Architecture Level.

Particular Architecture Level. The third level of architectural genericity is devoted to individual enterprise.

Partial models which meet the business requirements of a particular enterprise best will be selected and, if necessary, slightly modified by use of the standardized constructs from the Generic Architecture Level. Thus by use of offered partial models and generic constructs a particular enterprise model can be generated.

The above stepwise instantiation/aggregation from the generic constructs through the partial models towards the particular enterprise model can also partly be reversed, by first generating a particular model using generic constructs and, after addition of possible features and options, offering the result as a partial model for use within other enterprises.

The Particular Architecture Level is the domain of the individual enterprises.

2.1.2 Dimension of modelling levels.

Existing modelling techniques like IDEF leave it to the users discretion how to proceed in the modelling process, although in general a "top-down" approach is recommended. "CIM-OSA" recommends a "top-down/bottom-up" approach, which is supported by a pre-defined structure for the modelling process.

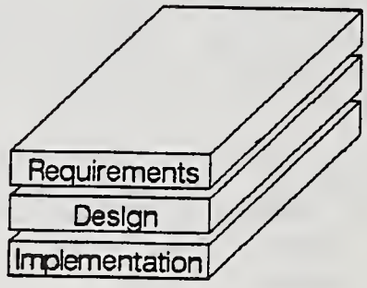


Figure 3. "CIM-OSA" Modelling Levels.

Similar to the approach in engineering of having an objective definition phase, followed by a detailed design phase and an implementation phase, three modelling levels are defined in "CIM-OSA". Each modelling level has defined tasks to be achieved and is supported by constructs. Within the three modelling levels the "top-down" approach is used (figure 3).

Requirements Definition Modelling Level. Within the uppermost modelling level all tasks of an enterprise which are required to perform the business of the enterprise are defined. Within this modelling level it is defined "what" has to be done in the enterprise in order to meet the business objectives.

While defining the tasks for all functional areas of the enterprise, the objectives and constraints, the execution sequence of the tasks, all data necessary and the capabilities of resources required to achieve the objectives of the tasks are defined - at a detailing level which is necessary to understand all business requirements.

Although the language used is non-technical it must be precise. Statements like "highest quality" must be quantified into say "98.5% error free parts", "ability to deliver at any time and place" must be converted into "delivery of XX products per day, within YY miles distance, ZZ hours service per day".

When defining the enterprise tasks it is important that existing business procedures and/or enterprise hierarchies are ignored. This is necessary, because the following modelling levels are based on the results of the first modelling level.

If, during the definition of the enterprise tasks, attention is given to existing procedures and/or hierarchies, most likely the outcome of the modelling exercise is a mirror image of the existing enterprise, preventing the chance to really generate a model driven by the enterprise requirements, challenging the old, empirical grown organisations.

By defining tasks under consideration of the business requirements only, a model of the enterprise is generated which contains a minimum of tasks, omitting some of the old existing procedures and organisations.

However, the tasks of the different functional areas of the enterprise usually are defined by experts who are well knowledgeable with their universe of discourse within the enterprise but not with other areas. This usually results in redundancies and non-optimum description of the enterprise, far away from integrated enterprise wide-functions and data. It is one goal of the following modelling levels to remove these redundancies and suboptima.

Design Specification Modelling Level. Within the second modelling level the requirements definition model will be converted into detailed specifications for each task, its data and resources needed for the execution, and the responsibilities within the enterprise.

The requirements model which defines "what" has to be done is now specified in terms of "how" the goals will be achieved. In case of multiple possibilities, all possibilities should be designed for later decision and implementation at the next modelling level.

Within each task to be performed the necessary data will be specified, broken down into data elements and associated to each other by relations.

Each specified task will be broken down into activities with specified input and outputs.

The required capabilities of the resources defined at the requirements definition modelling level will be converted into detailed specifications of people, machines, tools, fixtures and programs.

The result of the second modelling level is an optimised specification of all functions, data and resources. However, no decision is made at this modelling level about the implementation, i.e. there still exist multiple possibilities. This allows for pragmatic, constraint driven implementations without changing the specification of this modelling level. All design possibilities will be kept for later use and reference in case of changing implementations.

Implementation Description Modelling Level. Within the third modelling level the final decisions will be made about the enterprise operations. The prerequisites for these decisions are the design specifications of the previous modelling level, descriptions of available resources and pragmatic decisions due to enterprise constraints.

Available resources are resources already installed at the enterprise as well as possible resources from vendors, suppliers and service organisations. All these resources can be kept in a "component catalogue", which is updated whenever new resources or resource offerings are known.

Available resources which match resource specifications will be assigned to their business processes. This assignment includes decisions whether new resources will be installed at the enterprise or associated activities will be sub-contracted. In the latter it may happen that the implementation description model differs from the optimum design specification model, however the possibility of sub-

contracting is already defined at the previous modelling level.

In case the resource specifications of the second modelling level do not match resources from the "component catalogue", decision must be made whether new resources will be developed, or other implementation possibilities are considered. This may include a further iteration at the design specification modelling level.
A simulation process follows, during which is verified whether the implemented business processes together with their assigned data and resources achieve the defined business goals.

Although after simulation the modelling process is complete, it will be common praxis to execute a first "try run" of all business processes together with their associated data and resources for verification. After a successful "try run" the model will be released for the execution and control of the daily enterprise operation.

2.1.3 Dimension of views.

To enable optimisation of certain aspects like business process, data, resources and/or responsibilities within the enterprise, it is necessary to present all information belonging to these aspects to experts in a compressed form for their consideration and change. To accomplish this "CIM-OSA" provides different views. The structure of views also facilitates for graceful migration from a non-modelled enterprise by describing one view after the other whenever it is deemed necessary. Although there is room for many views in the architecture, four views are considered to be sufficient (figure 4).

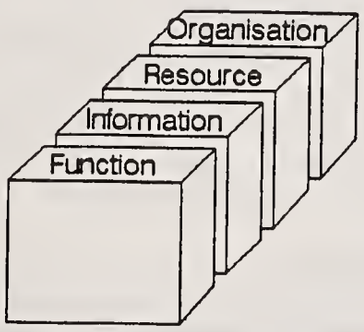


Figure 4. "CIM-OSA" Views.

The Function View. Within the Function View all tasks and sub-tasks of an enterprise will be described by way of a predetermined structure. The structure provided consists of enterprise functions (figure 5), which can be decomposed into further enterprise function levels. The number of levels is open and determined only by the individual modeller.

Each enterprise function contains a functional part, consisting of:

the objectives and constraints of this function,
the action to be performed,
the description of the inputs by way of material, data, controls and resources needed for the action,
the expected outputs from this action in form of material, data, control and resource information.

When an enterprise function is decomposed into a further lower level of enterprise function a behaviour part is added to the enterprise function. The behaviour part describes:

the associated lower level enterprise functions,
the direct or conditioned control sequence of the lower level enterprise functions (set of procedural rules) to be performed, as well as the details of the conditions,
the objectives and constraints associated to the lower level enterprise functions,
the events which can initiate execution of the enterprise lower level enterprise functions.

The third part of the enterprise function, the structural part, contains the relation between enterprise functions and their associated parents and/or children.

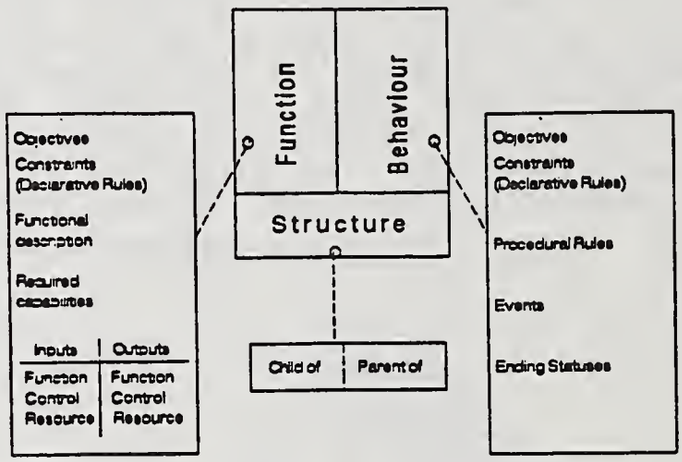


Figure 5. Enterprise Function.

To differentiate between the levels of function decomposition the function view is divided into domains of discourse (a part of the enterprise under modelling consideration), business processes and enterprise activities. Business processes and enterprise activities can be of multiple use if required (figure 6). At this point of the description it should be noted again that the domains and/or business processes are function-oriented and not organisation-oriented.

At the Design Specification modelling level Enterprise Activities can be further decomposed into Functional Operations, which are atomic units of work within an enterprise. A Functional Operation cannot be decomposed further into smaller units of work and will be executed by one and only one Functional Entity (see Resource View). "CIM-OSA" currently specifies five types of Functional Operations:

- Storage Functional Operations,
- Application Functional Operations,
- Machine Functional Operations,
- Communication Functional Operations,
- Human Functional Operations.

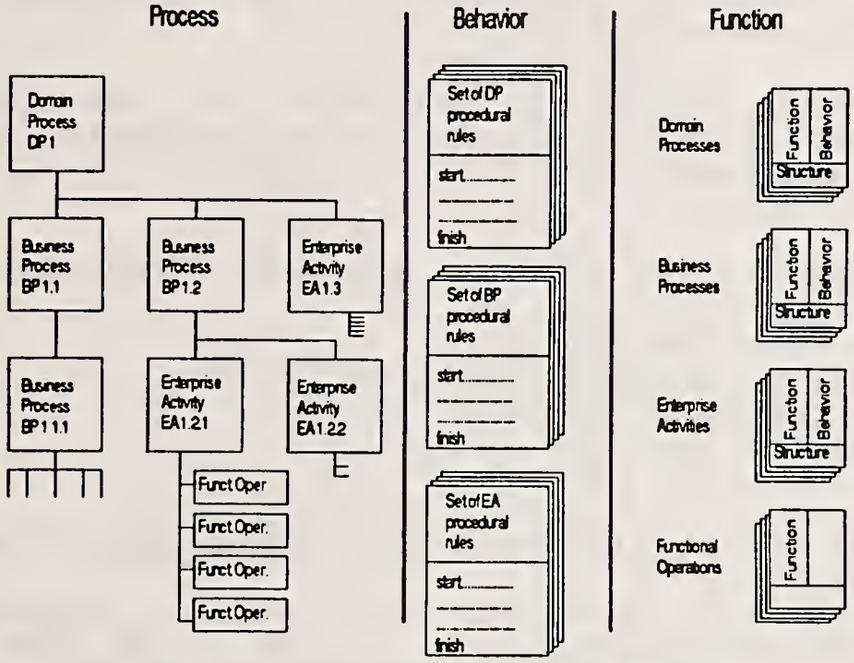


Figure 6. Decomposition Levels within Function View.

Information View. Within the Information View all business processes, enterprise activities, data, controls, resources and physical objects will be described as information objects. Information objects can be decomposed into smaller information objects. All information objects regardless of their complexity can be observed from different object views, which can be decomposed into information elements and linked together by way of the entity/attribute/relationship. Different external, conceptional and internal schemas can be defined.

Resource View. The purpose of the resource view is to present all resource relevant information (employees, programs and all machine technology, office technology and information technology machines) to the enterprise modeller for optimisation purposes. Optimizing resources is one of the most important integration tasks, since this ensures optimum usage and load distribution for all investments. Therefore, CIM-OSA introduced and supports this particular view.

Resources are described by way of required, specified and/or implemented capabilities of **Functional Entities**. A Functional Entity is an independent closed unit which is able to receive, send and optionally store data and/or material. This means that any shop floor device which is addressable by a protocol is a Functional Entity. Functional Entities execute Functional Operations (see Function View).

Functional Entities are similar to the Virtual Machine Device (VMD) defined in the Manufacturing Messaging Service (MMS), however the "CIM-OSA" defined Functional Entity is extended to also include computers, computer input/output devices and application programs (specifically the services of the Integrating Infrastructure).

Creative human activities as well as executing human activities must also be described in an enterprise model. For that reason the humans within the enterprise are also treated as Functional Units.

Within "CIM-OSA" currently five types of Functional Entities are defined:

- Storage Functional Entities,
- Application Functional Entities,
- Machine Functional Entities,
- Communication Functional Entities,
- Human Functional Entities.

The communication between Functional Entities is transaction-oriented, i.e.:

- The requesting Functional Unit sends a request to the responding Functional Unit via the System Wide Exchange Service.
- The responding Functional Unit executes the requested service.
- The responding Functional Unit sends a response to the requesting Functional Unit via the System Wide Exchange Service.

Organisation View. During definition of functions, data, controls and resources the access rights and change responsibilities should be defined. Assignment of responsibilities may be deferred to a later point in time, however, prior to putting the model in control of the enterprise, the responsibilities and access/change rights must be defined. Since these responsibilities are very vital to the total enterprise operation, "CIM-OSA" supports one additional view.

2.1.4 Overview and use of the modelling framework.

Overlaying the three levels of genericity, the three levels of modelling and the four views, the resultant framework is constructed as shown in figure 7.

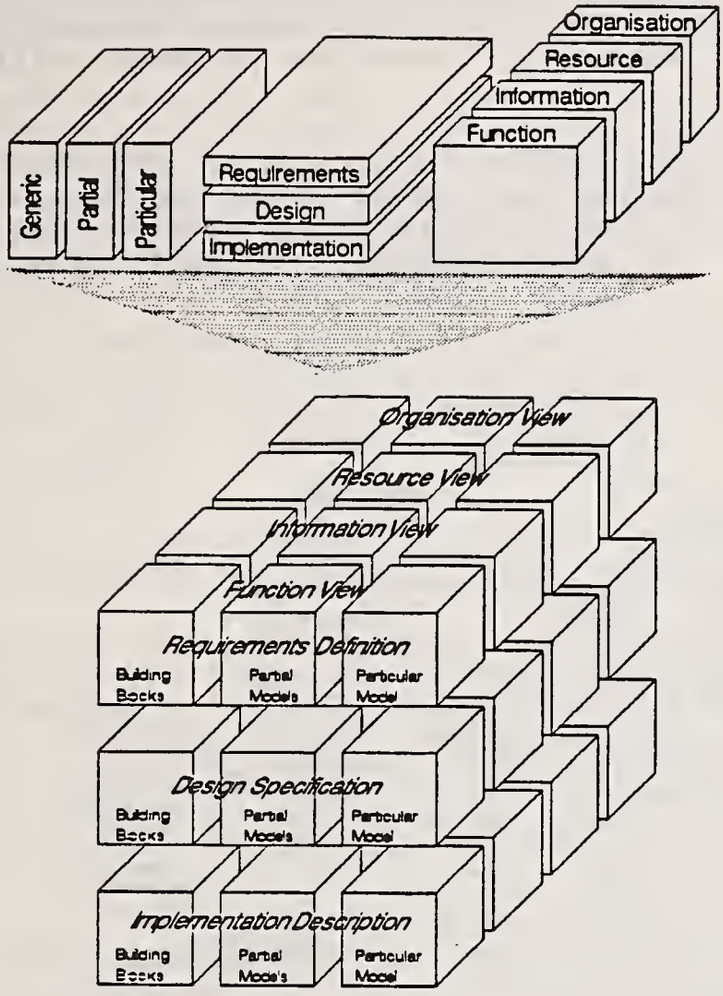


Figure 7."CIM-OSA" Framework for Modelling.

The integrating effects of the Framework for Modelling are:

Detailing all data into elements, storing them in an enterprise wide data file and making them available to all views as well as all application programs, removing existing data redundancies, thus enabling integrated data bases.

A common file containing all defined activities enables reuse of them and subsequently avoids multiple definition of the same activities. This facilitates for integrated enterprise activities.

Resource specifications stored in a common accessible file enables resource assignment at execution time which in turn allows for better utilisation of resources. This resource integration avoids unwanted duplication of equipment and personnel.

Although the modelling process has been described in the above unidirectional way, e.g. from the requirements definition modelling level via the design specification modelling level to the implementation description modelling level and from the

function view via the information view and resource view to the organisation view this unidirectional proceeding cannot always be achieved. Rather an iterative approach will be more likely.

The modelling of a particular enterprise is performed only at the particular architectural level (figure 8), while the generic and the partial architectural levels support the modelling by way of standardised constructs and reusable partial models.

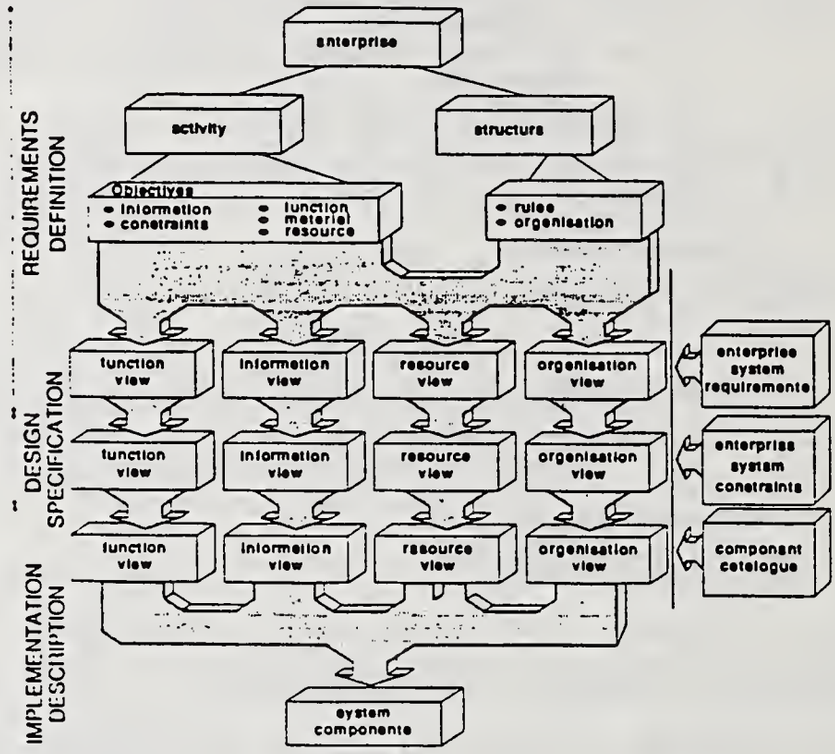


Figure 8. "CIM-OSA" Usage for a Particular Enterprise.

2.2 The Integrating Infrastructure.

Models generated under the rules of "CIM-OSA" can be used for analysis, improvement and simulation of enterprise business processes, however it will - in addition to all modelling methods known until now - also assist in the execution of the daily work of the enterprise operation by using occurences of the model for the guidance and control of the business processes.

In order to achieve this goal, supporting services are required in the enterprises' Information Technology support, which will:

- Generate occurences of business process models, initiated by events of the daily enterprise operation.
- Provide the generated occurrence with the data relevant for this particular business process occurrence.
- Initiate the corresponding enterprise activities.
- Generate other business process and/or enterprise activity occurences under control of the model and the results achieved from previous enterprise activities.
- Plan, start and release the associated resources.
- Prepare the achieved results for the use and/or control of further business processes and/or enterprise activities.

The services must be able to achieve the above goals within a complex, distributed, heterogeneous system.

Such services are addressed by the second part of the ESPRIT AMICE project. As mentioned before this part is concerned with an Integrating Infrastructure, which is - in terms of "CIM-OSA" - a Partial Model at the Implementation Description Modelling Level. Generic constructs used for this Partial Model are Functional Entities, Services, Agents and Protocols.

2.2.1 IIS Services.

Within the Integrating Infrastructure of "CIM-OSA" the following services are discussed and will be supported by pilot installations:

- System Wide Data Service,
- Data Management Service,

- Human Front End Service,
- Machine Front End Service,
- Application Front End Service,

- Business Process Control Service,
- Activity Control Service,
- Resource Management Service,

- System Wide Exchange Service,
- Communications Management Service.

The System Wide Data Service is responsible for the storage and retrieval of data within the total (distributed) CIM-system. Since the user is concerned only with logical data addresses and does not care about real data storage location the System Wide Data Service is responsible for translation into different schemas, context management and data security. The System Wide Data Service works independently of the implemented data storage systems, in order to avoid that

change of data storage system requires change of the total System Wide Data Service.

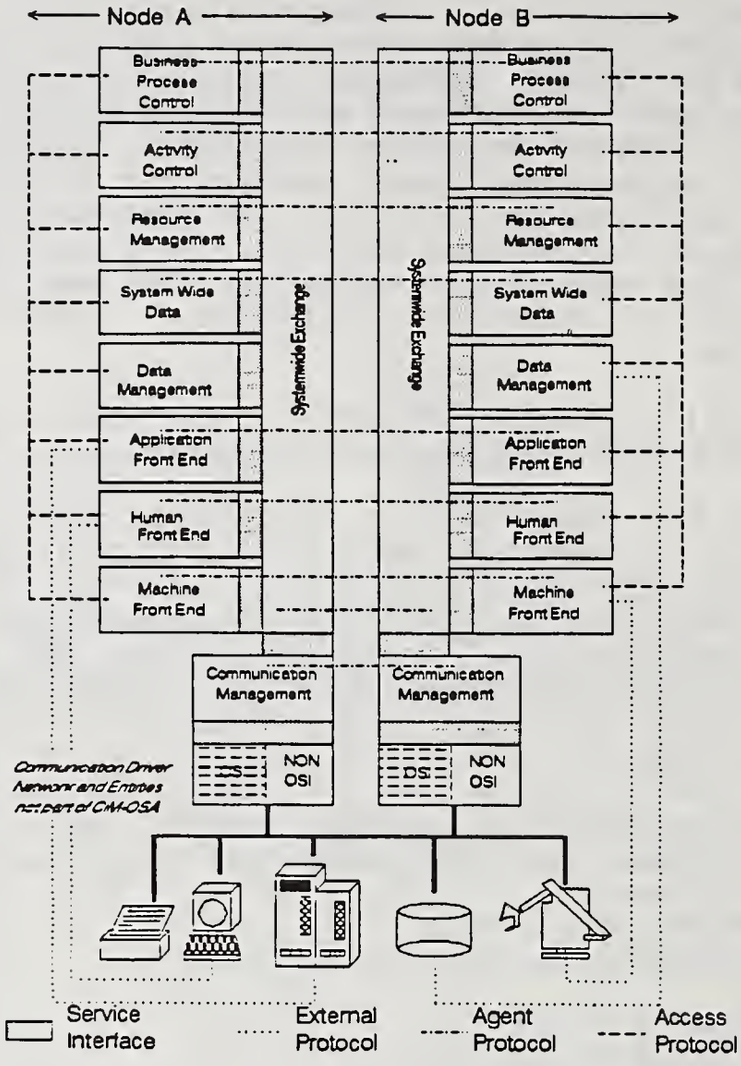


Figure 9. The Services of the Integrating Infrastructure.

The Data Management Service supports the use of heterogeneous data storage systems and data base management systems (DBMS). The storage/retrieval operations requested from the System Wide Data Service are still neutral in respect to physical data storage devices. The necessary adjustments for the DBMS will be done by the Data Management Service and after that directed to the addressed DBMS. This service is necessary to isolate the implemented DBMS and its associated data storage devices from the DBMS independent System Wide Data Service.

The Human Front End Service serves as the connection between the user and the CIM system. It establishes an implementation and application independent communication between the human at

one side and the machines and application programs at the other side.

The Machine Front End Service supports communication with the machines at the shop floor like NC or RC controlled machines, programmed logical controllers and automated guided vehicles. The Machine Front End Service presents all machines of the shop floor to the CIM system, takes care of the communication between the CIM system and the shop floor machines and controls the execution of shop floor operations and reports the statuses to the other services.

The Application Front End Service takes care of the communication between the CIM system and the application programs. "CIM-OSA" compatible as well as non-compatible application programs will be supported. Under control of the modelled business process the application programs will be initiated, interrupted and terminated and appropriate data will be obtained/stored via the System Wide Information Management Service. All communication with the application programs will be routed via the Application Front End Service.

The Business Process Control Service controls the execution of the daily enterprise operation. Initiated by events an occurrence of the appropriate business process will be set up in the Business Process Control Service. The enterprise activities belonging to this business process will then be initiated under control of this business process occurrence. Since within an enterprise hundreds of activities of many different business processes may run in parallel and even several occurrences of the same business process may be executed simultaneously the Business Process Control Service is supported by the Activity Control Service and the Resource Control Service.

The Activity Control Service controls the execution of enterprise activities within a (distributed) CIM system. Under control of the Business Process Control Service the Activity Control Service initiates occurences of enterprise activities, supplies the required data and controls for the execution of the activity and monitors the execution of the activity. After termination of an enterprise activity the result and status will be reported to the Business Process Control Service for initiation of the next defined activity.

The Resource Management Service controls all resources of the (distributed) CIM system. Long and short range planning as well as final association and release for use of resources for the execution of enterprise activities lies within the responsibilities of the Resource Management Service. Resources will be associated and released according to the required capabilities, the requested schedules and the availabilities of the resources. That means the Resource Management Service may, if possible, wait for the most efficient resource,

however it may also associate less efficient resources in order to achieve optimum use of all resources.

Within the Resource Management Service "logical cells" can be assigned. By use of the "logical cell" construct it is possible to combine resources to a group of resources. This means a "logical cell" can be either a physical manufacturing cell or a group of single resources achieving together the same result.

The System Wide Exchange Service takes care of the communication between the (distributed) services. All services operate under use of logical addresses only and the services can be distributed in several physical or logical systems. System Wide Exchange is then responsible for directing the messages between the different services to the correct service (translating logical addresses into real addresses) and forwarding responses to the requesting service. System Wide Exchange Service will make use of the Communication Service in case of intersystem communication.

The Communications Management Service will be used for intersystem communication. The service supports OSI as well as private communication networks, and is responsible for the translation from logical network addresses to real network addresses as well as for depicting the proper communication protocol.

All services can be located within one system, however it may be possible that services not required in a node will not exist in this node. Also all services may be distributed over several (heterogeneous) logical or physical systems.

2.2.2 IIS Communication.

Within "CIM-OSA" communication between services as well as communication between services and external machines will be accomplished by use of three different protocol types.

- One protocol type, the access protocols, to be used for communication between different services.
- One protocol type, the agent protocols, to be used for communication within distributed services of the same service type.
- One protocol type, the external protocols, to be used for communication between the services and the external Functional Entities.

All types of protocols are based on concepts used already in OSI. For example the concepts of service - client and the concept of agents within distributed services will be used.

All protocols are generated by a requesting service, and routed via the System Wide Exchange Service to the responding service. The System Wide Exchange Service makes automatic use

of the Communications Management Service in case of the responding service being in another node. Rules for missing responses, system break down and partial responses will be defined.

It is realised that other standardisation activities like Open Distributed Processing (ODP) and Transaction Processing (TP) will influence the "CIM-OSA" protocols under development.

3.0 "CIM-OSA" within standardisation.

Over time the ESPRIT AMICE consortium intends to move the concepts of "CIM-OSA" into national and international standardisation.

Five standards are currently foreseen, for which contributions for appropriate standardisation bodies will be generated:

"CIM System Integration - Framework for Modelling". This paper has already been discussed in Europe by CEN/CENELEC/AMT/WGArc and will be given to the European member bodies for ballot in March 1990. If the paper is accepted it will be an "Experimental European Standard" for about three years, after which it will be either replaced by an international paper or carried on as a European Standard.
The paper has also been moved into ISO/TC184 and is discussed in SC5WG1.

"CIM System Integration - Constructs for Views". This will be the next paper to be discussed within the CEN/CENELEC/AMT/WGArc, pending ITAEGM (Information Technology adhoc Expert Group for Manufacturing) approval. The paper will discuss the generic constructs for all views, a definition of a formal description language and a definition of a graphical representation. The paper may have several parts, one for each view.

"CIM-System Integration - Framework for the Integrating Infrastructure". This paper shall define the general concepts of the Integrating Infrastructure, like service/client concept, services required, types of protocols to be used.
"CIM System Integration - Service Definitions". This paper will define the functions of all services defined in the Integrating Infrastructure. It may consist of several parts, one for each defined service.

"CIM System Integration - Protocol Definition". This paper will define all protocols used for the services of the Integrating Infrastructure. It may contain several parts, one for each defined service.

Beyond the above listed papers to which the ESPRIT AMICE

consortium intends to provide contributions based on their conceptional findings and their pilot installations, further papers will be required such as:

Data Element Library. Without unambiguous definition of data elements Partial Models will never fit, they will have to be adjusted to each other by a large number of "aliases", wasting a lot of effort in all enterprises. It is envisioned that data elements used by partial models will be properly defined and included in a catalogue, similar to the elements defined in EDIFACT. In fact the data elements defined in EDIFACT may be considered a starting list.

Capabilities of Resources. Component lists as described in section Implementation Description Modelling Level will never match Required Capabilities if both are not using the same terms. Thus a method must be found to describe resources by parameters in an unambiguous way. The recently started project "CAD-LIB" in ISO/TC184SC4 may be a solution to this problem.

The above list is not considered final, it is just an appreciation of the immediate needs. As experience has shown in the area of Open Systems Interconnection (ISO-OSI) standardisation bodies will find more areas to be explored as CIM knowledge matures.

CIM-OSA - AN ILLUSTRATIVE EXAMPLE OF HOW TO APPLY THE MODELLING FRAMEWORK

DIRK BEECKMAN

ABSTRACT

The ESPRIT project AMICE is developing the CIM-OSA architecture. This architecture mainly consists of two parts: the modelling framework and the integrating infrastructure.

The CIM-OSA modelling framework guides and supports the CIM system life cycle : first the business requirements of the enterprise are identified and, starting from these requirements, an executable, physical CIM system is built.

Because of the abstract nature of the concepts of the modelling framework, an illustrative example of how to apply those concepts is given in this presentation.

1. Introduction

The CIM-OSA modelling framework has identified three levels of architectural genericity, three modelling levels and four views needed to guide and support the CIM system life cycle. The modelling framework itself and the basic concepts defined at the generic architectural level are introduced in [PAN90]. In the reference section of this paper, additional documents on CIM-OSA are identified.

From the start of the project, AMICE has always spent important efforts on validation activities, to ensure that the concepts it is developing really contribute to the development of valuable CIM systems. Therefore, several case studies, a CIM-OSA demonstration and several partial prototypes have been realised so far. In the near future, it is the intention of AMICE to demonstrate the feasibility and the usefulness of the CIM-OSA concepts by an overall CIM-OSA prototype.

In all case studies, the CIM-OSA concepts have been used to re-model an existing, operational CIM system. The illustrative example presented here is extracted from a case study modelling a part of a Flexible Manufacturing System (FMS) developed and installed by COMAU in the FIAT AUTO Mirafiori plant in Turin, Italy.

2. The FMS in the FIAT AUTO Mirafiori plant.

2.1 Overall functionality of the FMS

The FMS in the FIAT AUTO Mirafiori plant produces different types of cylinder heads and different types of manifolds for the FIAT Croma turbo diesel engine and for the ALFA ROMEO Alfa 33 boxer engine. The FMS is a dynamic system : although already operational today, an evolution plan with a number of well defined upgrade steps exists to increase the intelligence and the flexibility of the system.

The FMS has 14 machining centers, a robotized washing station, a robotized measuring station, 6 part load/unload stations, 2 refixturing stations, an automated transportation system with 6 AGV's, 18 buffer positions, and 40 pallets used as support for the fixtures holding the parts.

The overall functionality of the FMS can be subdivided into following main functions :

- scheduling (decide which job orders will be launched and when),
- dispatching (launch a job order),
- part program management (see below),
- fixture management,
- tool management,
- monitoring (reporting breakdowns, statistics).

The case study has focused on the part program management, and therefore, this functionality will be discussed in detail.

2.2 The Part Program Management

The part programs (PP) contain all information needed by the FMS to run the machining centers. A part program has a part program body and a part program header.

The part program body is a set of executable NC commands for driving the machining centers' components (moving slides, spindles, tool magazines,...).

The part program header contains the following information:

- the part program name,
- the length of the part program body,
- the part program execution time,
- the list of the tools used by the part program body, and
- for each tool, the cutting time.

The part program body is generated off-line by the CAD/CAM department and introduced by tape into the NC controller. The NC code is tested and when those tests are satisfactory, the part program is completed by adding the part program header. Afterwards, the part program is integrated into the global information system of the FMS and released for operational use.

The part program management function thus allows the creation of part programs. Besides, it contain all necessary maintenance functions (delete, purge, modify) of the part programs and makes the part programs available for setting up the machining centers on request of the dispatching function.

3. Examples illustrating the concepts

In order to illustrate the CIM-OSA concepts, a set of figures has been prepared corresponding to the modelling levels and views on which the AMICE work has reached a stable status.

Abbreviations used in the examples

BP	Business Process
DM	Design Specification Modelling Level
EA	Enterprise Activity
FV	Function View
IV	Information View
RM	Requirements Definition Modelling Level
PR	Procedural Rule

Function View of the Requirements Definition Modelling Level

The concepts Domain and Business Process and the hierarchical decomposition is illustrated. Some Business Processes are shown in detail by developing their set of Procedural Rules. An Enterprise Activity is fully detailed with its Function, Control and Resource Input and Output.

This Input/Output is making the link with the Information View.

Function View of the Design Specification Modelling Level

One of the Enterprise Activities introduced at the Requirements Definition Modelling level is decomposed into Functional Operations. The execution of these operations will realise the functionality of the Enterprise Activity. The Functional Operations make the link with the Resource View, which is currently under development.

Information View of the Requirements Definition Modelling Level

The input and output identified in the Function View, are detailed as Object Views and consolidated through the creation of the set of interrelated Enterprise Objects.

Information View of the Design Specification Modelling Level

The conceptual schema of the Enterprise Objects is developed. This schema is translated afterwards into a relational model to produce the Logical Data Model of the Internal Schema.

DOMAIN

FLEXIBLE MANUFACTURING SYSTEM (FMS)

DOMAIN OBJECTIVES

MANUFACTURE PARTS
MEET DUE DATES
MINIMISE LEAD TIMES
KEEP LOW WIP INVENTORIES

DOMAIN CONSTRAINTS

PAY BACK AFTER 4 YEARS
AVERAGE PRODUCTION OF 400 PARTS/DAY

DOMAIN PROCESSES

SCHEDULING
TOOL MANAGEMENT
MONITORING
PART PROGRAM MANAGEMENT
DISPATCHING
FIXTURE MANAGEMENT

Figure 1. RM - FV - Domain (1)

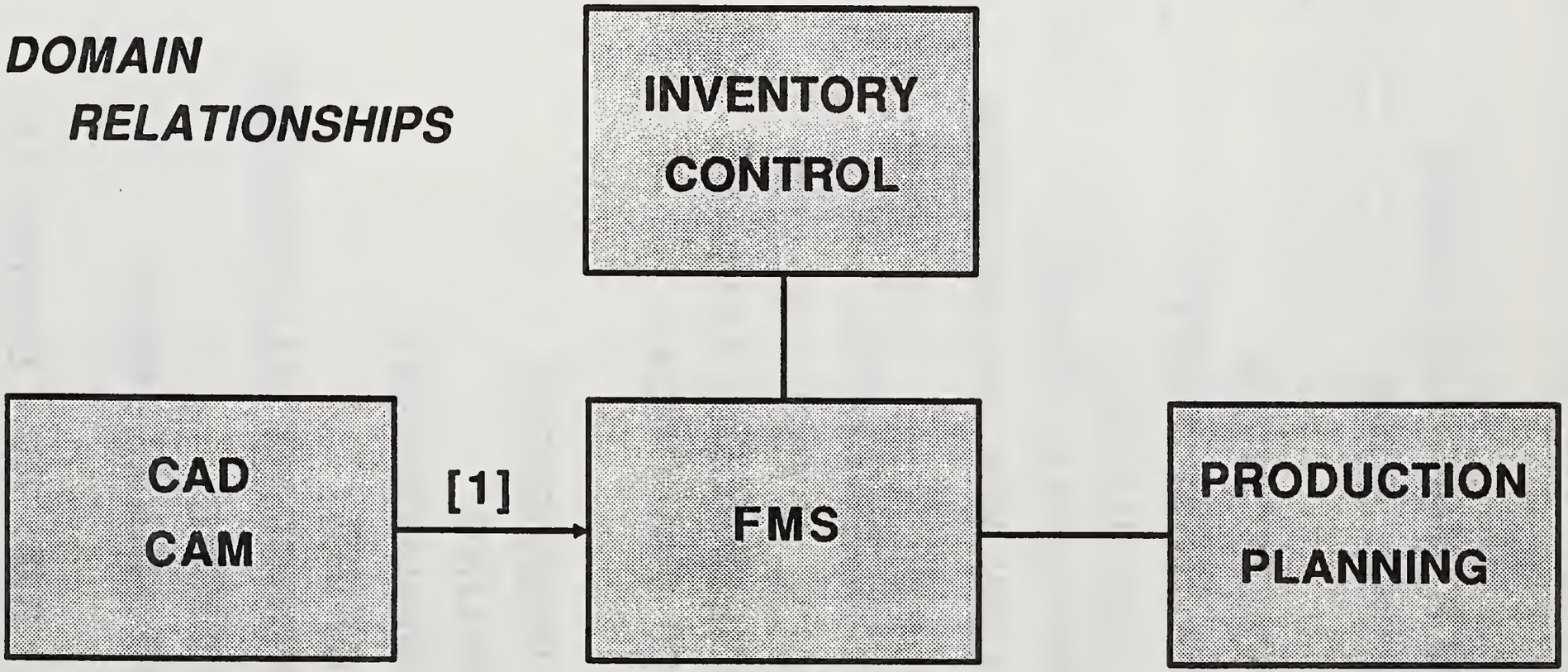


Figure 2. RM - FV - Domain (2)

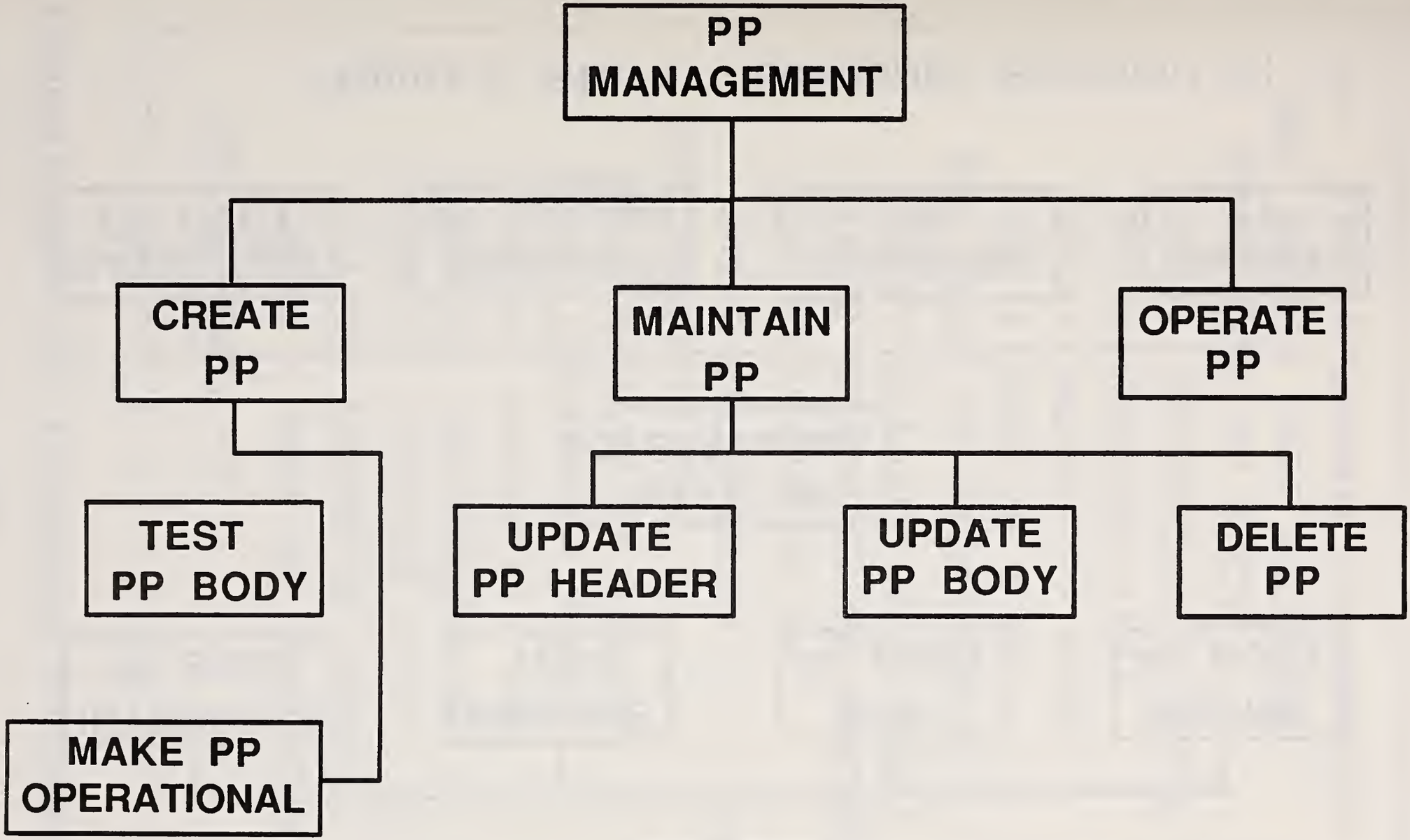


Figure 3. RM - FV - Enterprise Functions (1)

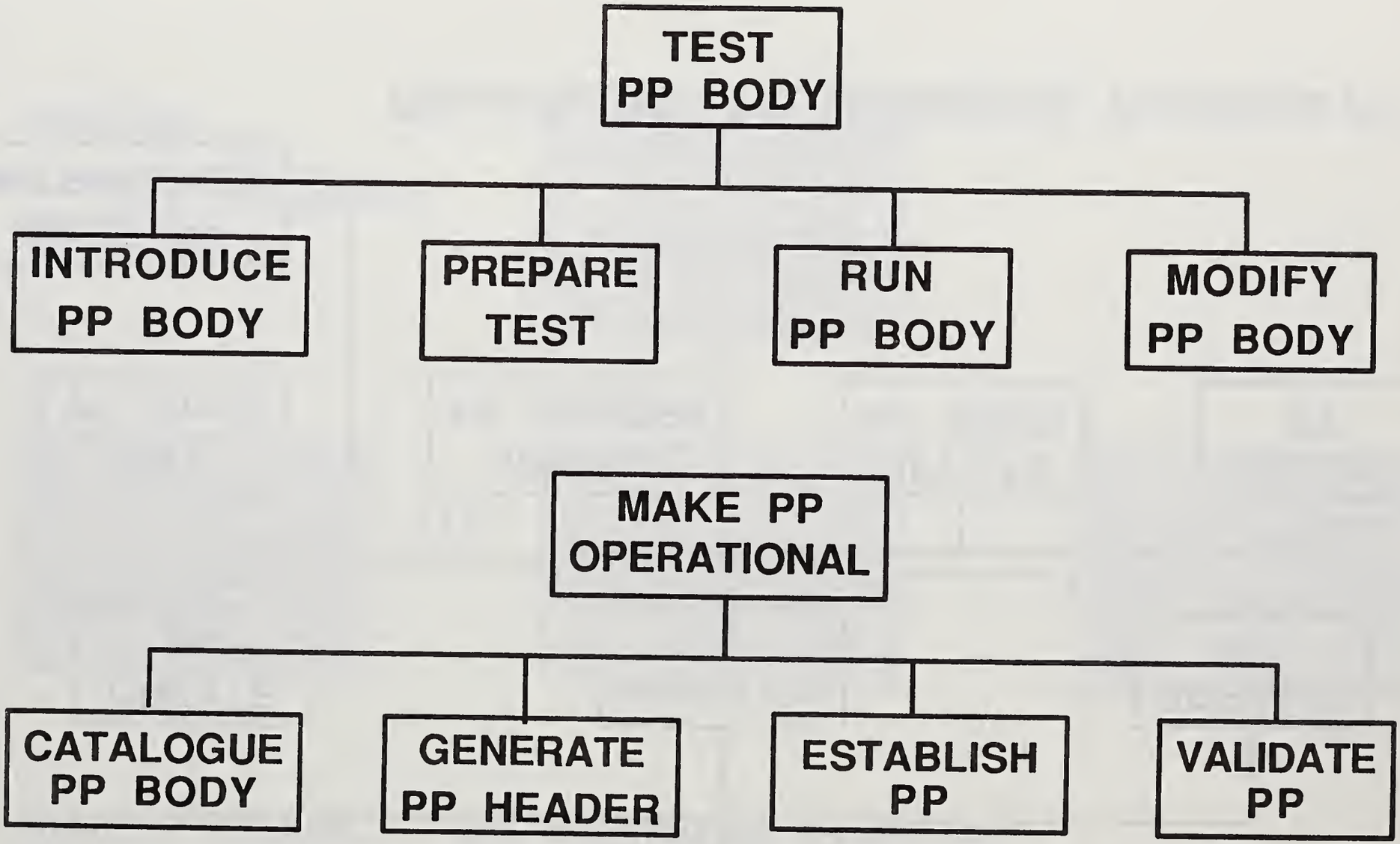


Figure 4. RM - FV - Enterprise Functions (2)

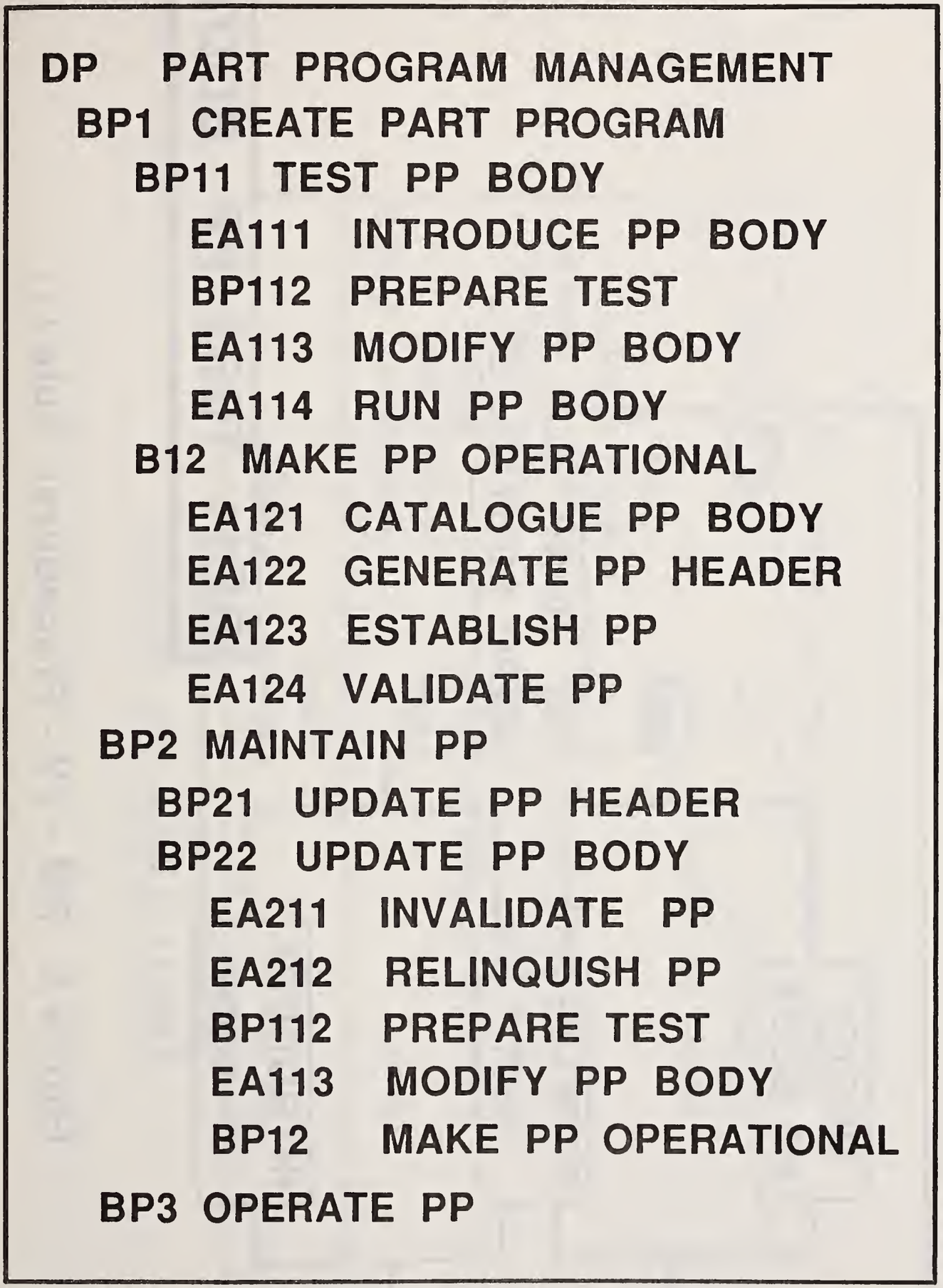


Figure 5. RM - FV - Enterprise Functions (3)

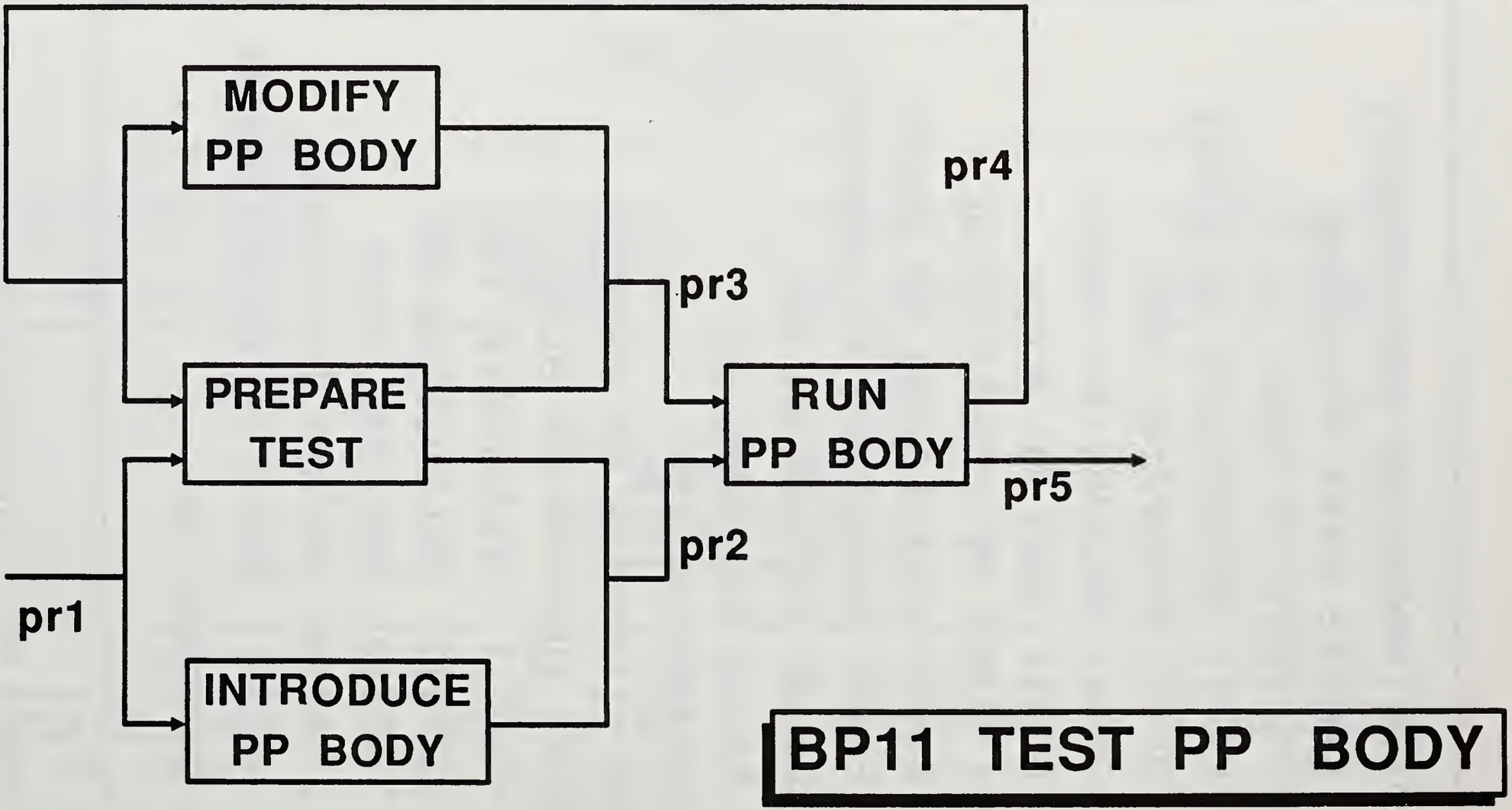


Figure 6. RM - FV - Procedural Rule (1)

BP11 TEST PP BODY

No.	Wait for	Ending Status	Trigger
1	Start		Introduce PP Body & Prepare Test
2	Introduce PP Body & Prepare Test	done done	 Run PP Body
3	Run PP Body	error done	Modify PP Body & Prepare Test Finish
4	Modify PP Body & Prepare Test	done done	 Run PP Body

Figure 7. RM - FV - Procedural Rule (2)

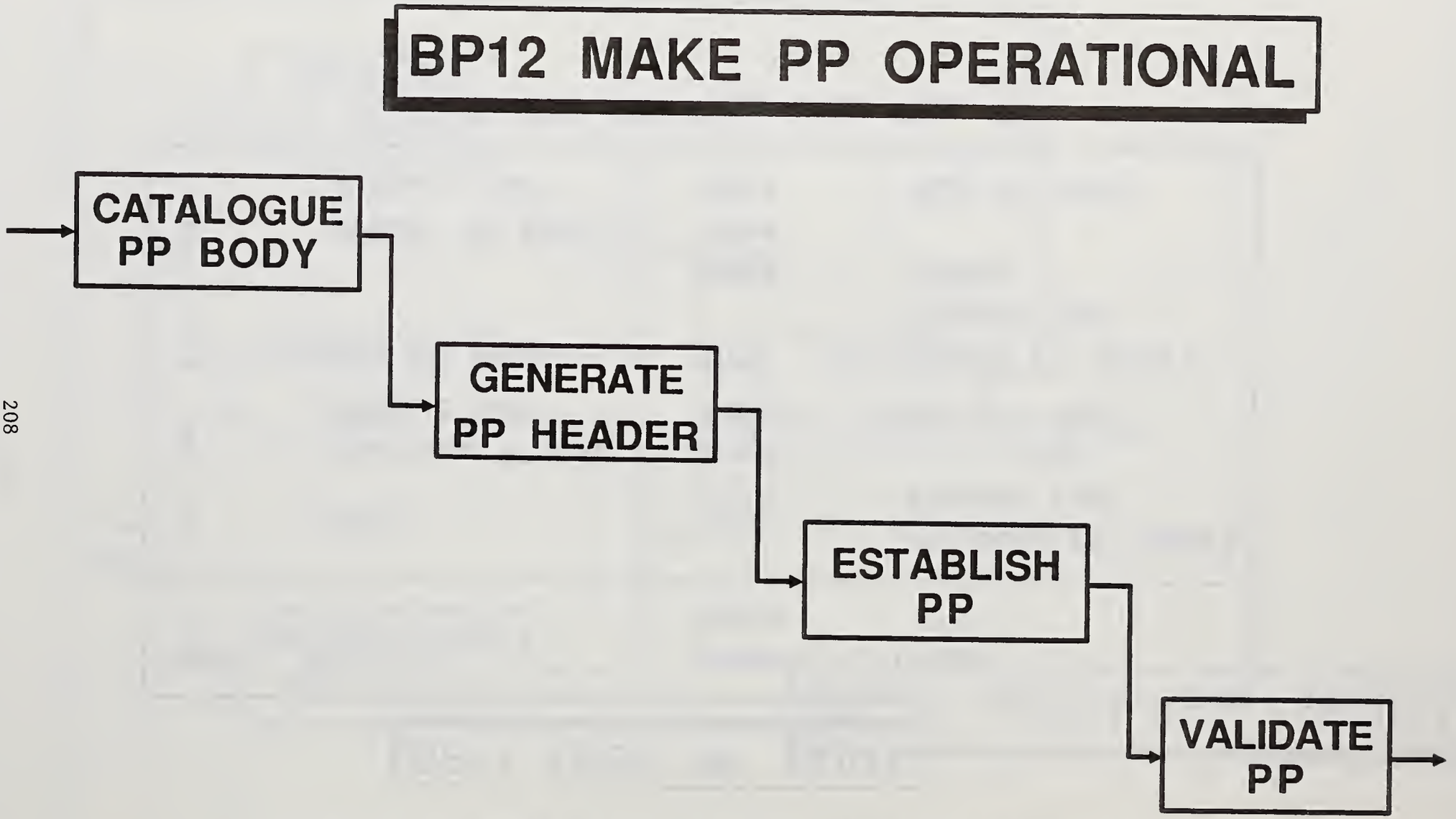


Figure 8. RM - FV - Prodedural Rule (3)

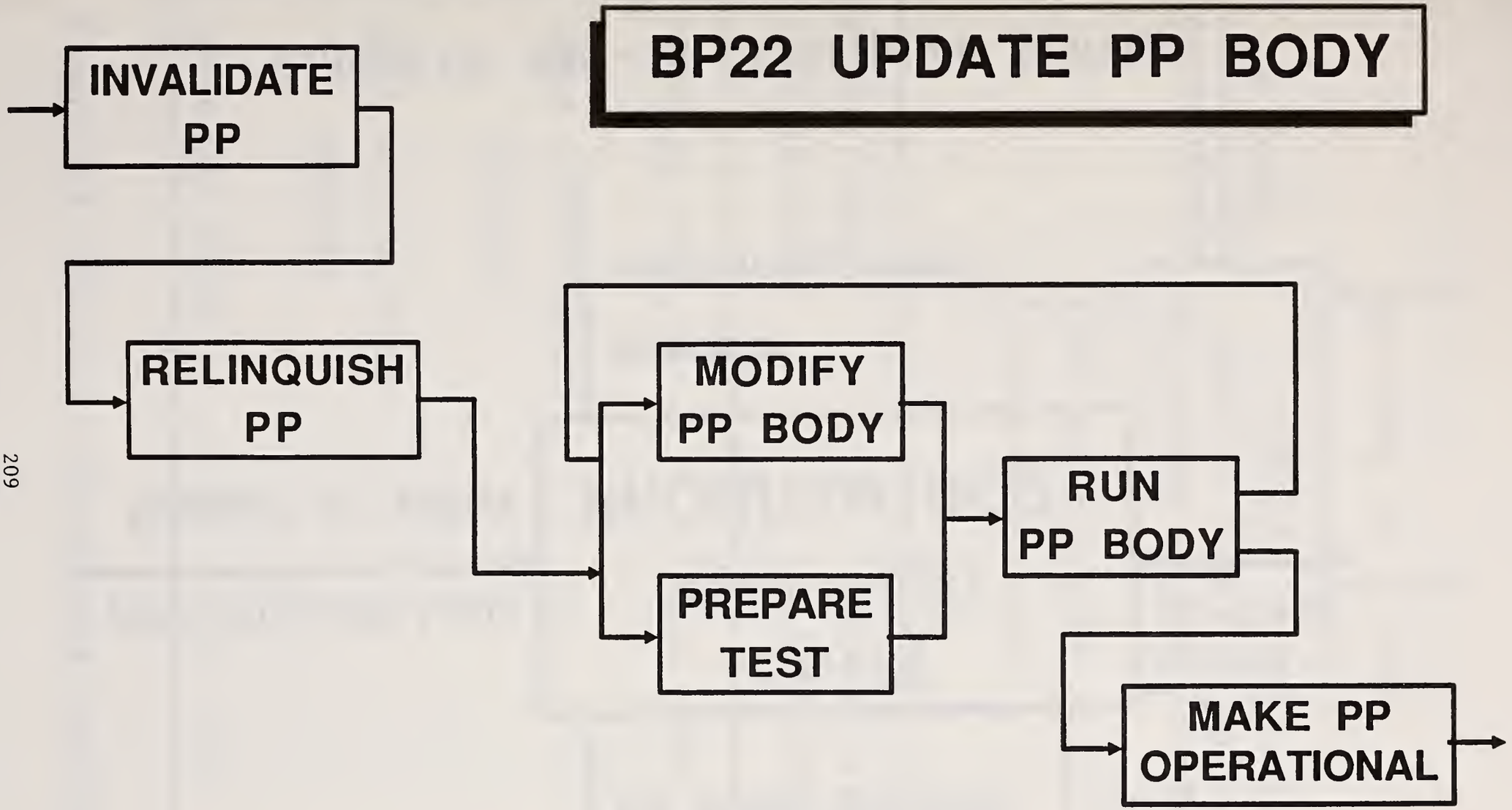


Figure 9. RM - FV - Procedural Rule (4)

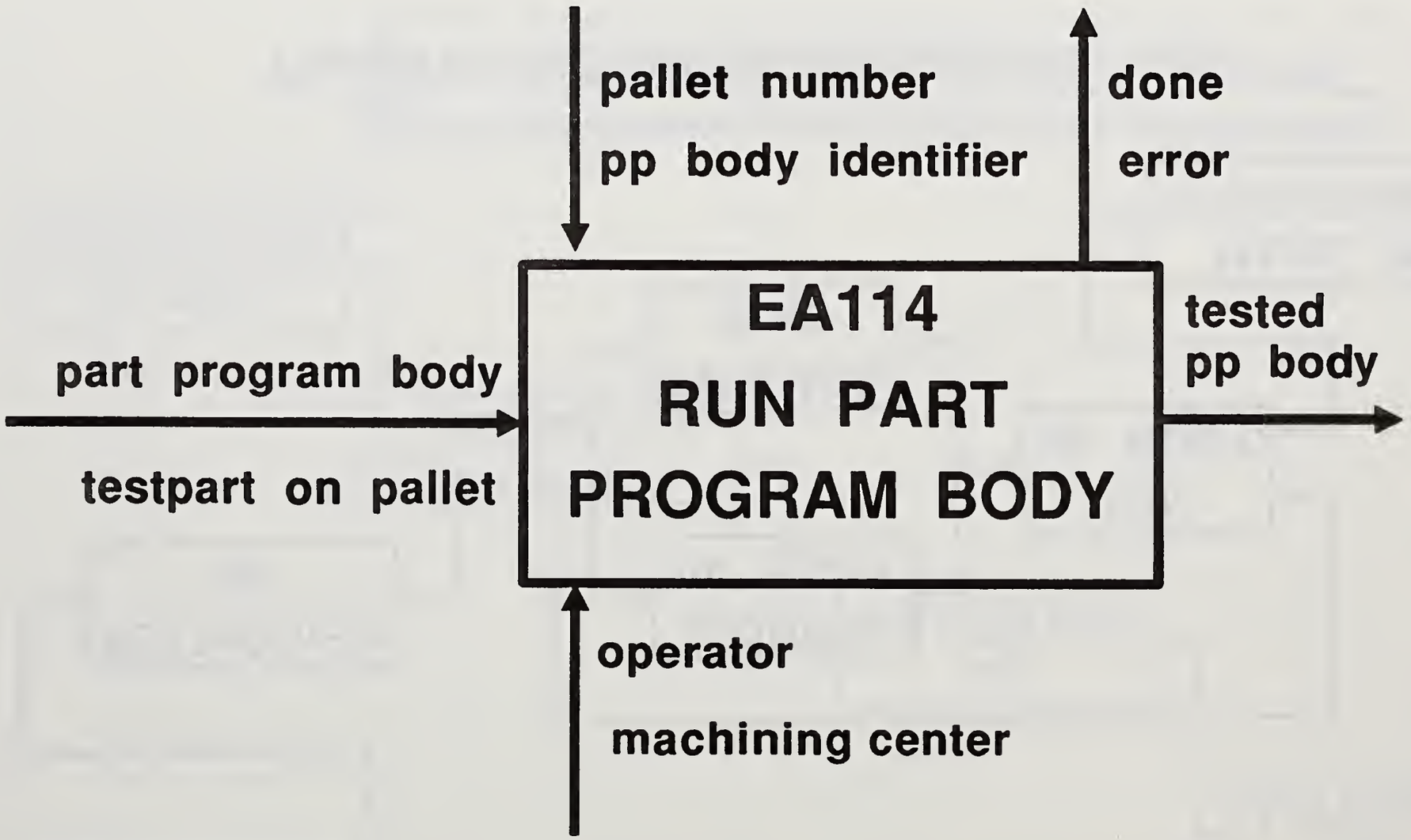
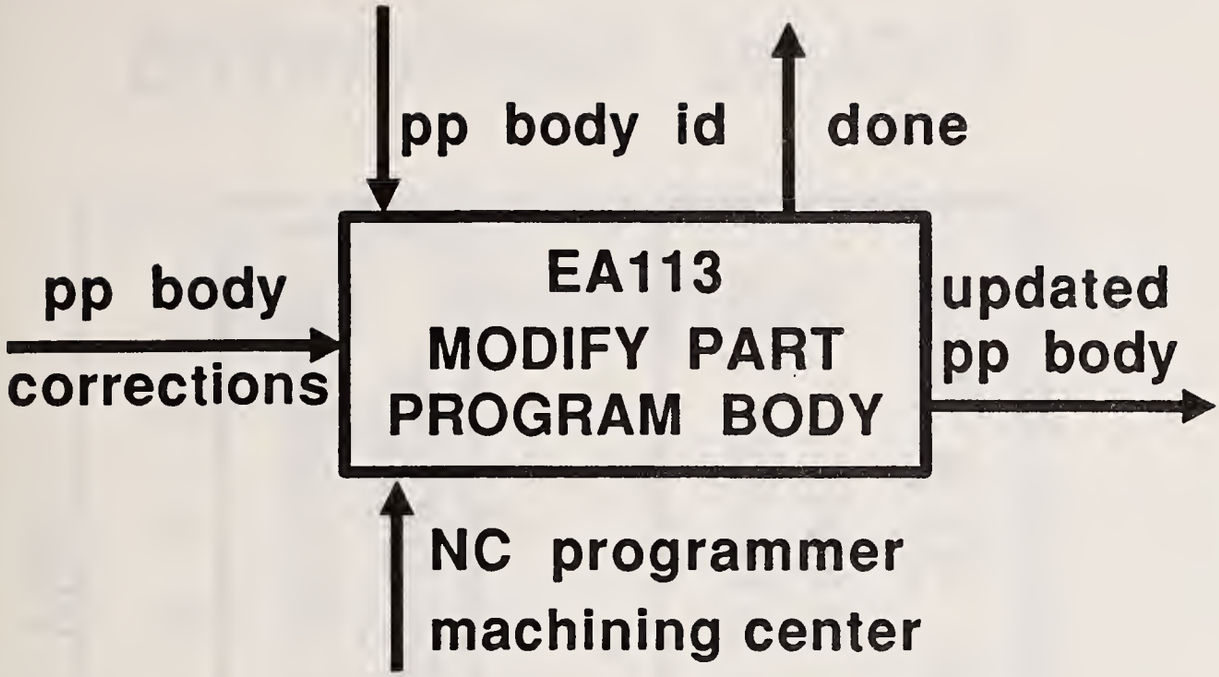


Figure 10. RM - FV - Enterprise Activity



FUNCTIONAL OPERATIONS	SUPPORTING IIS SERVICE
load pp body	sd - dm - mf
start modification session	hf
end modification session	hf
save pp body	sd - dm - mf

Figure 11. DM - FV - Functional Operation

OBJECT VIEWS

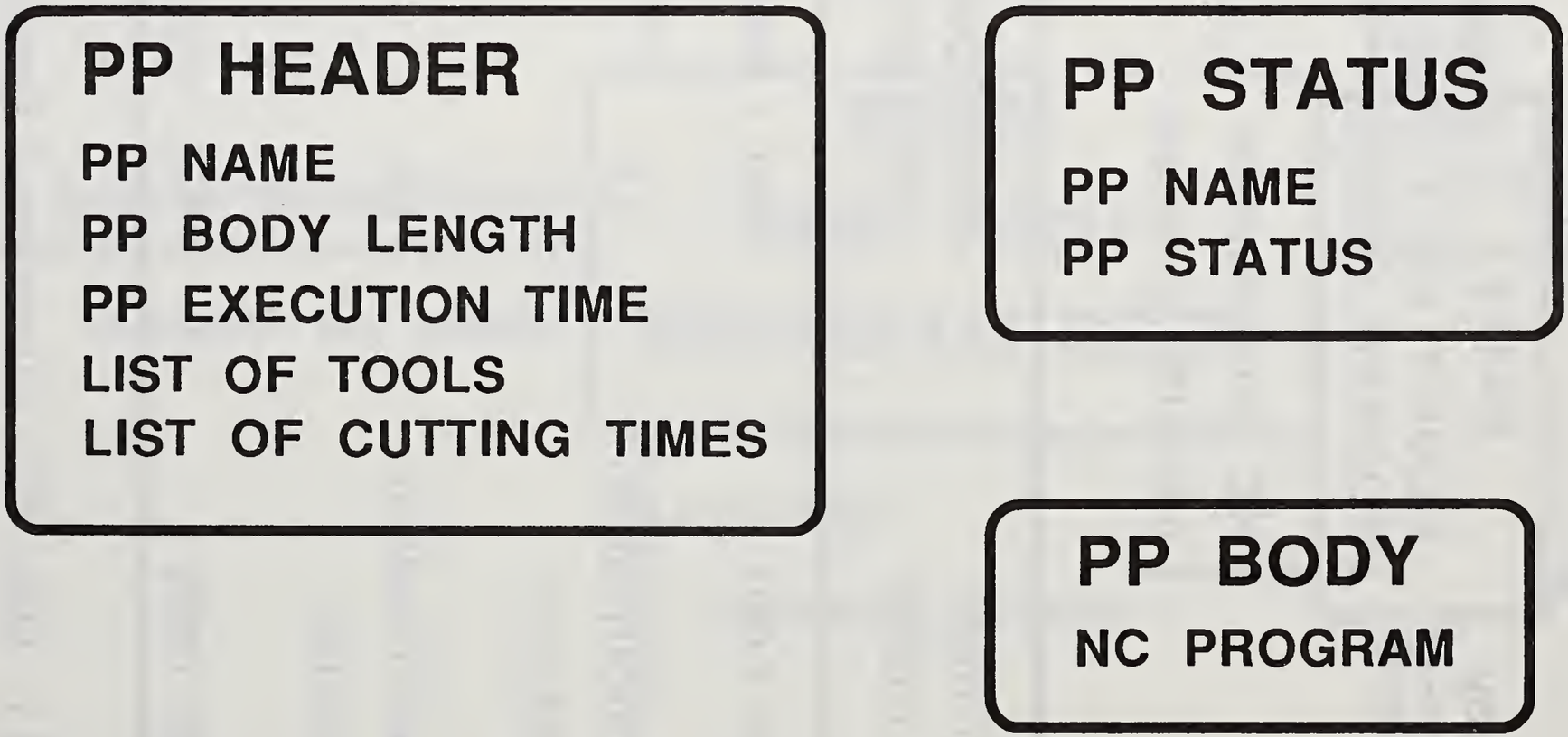


Figure 12. RM - IV - Object View

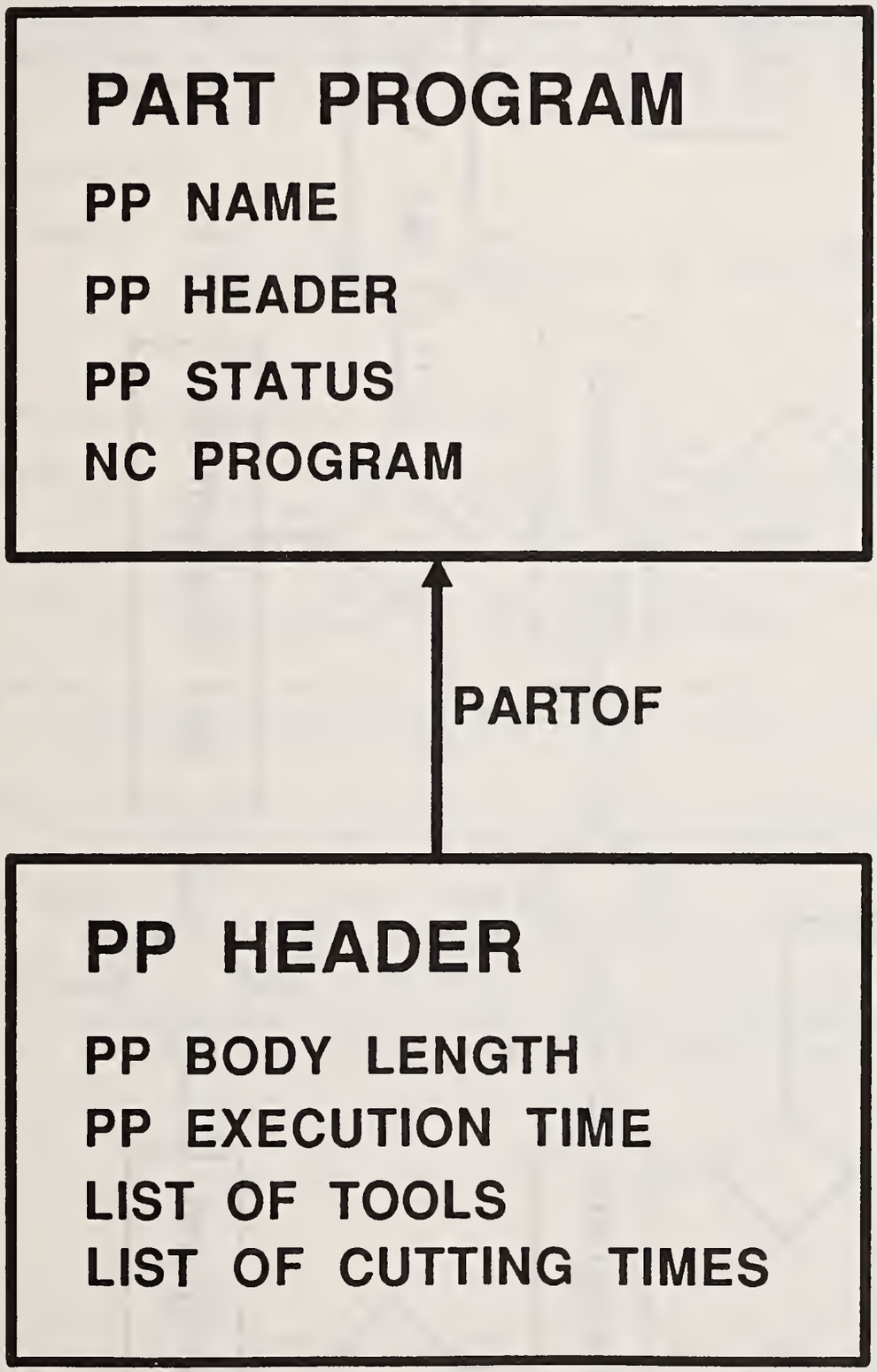


Figure 13. RM - IV - Object View

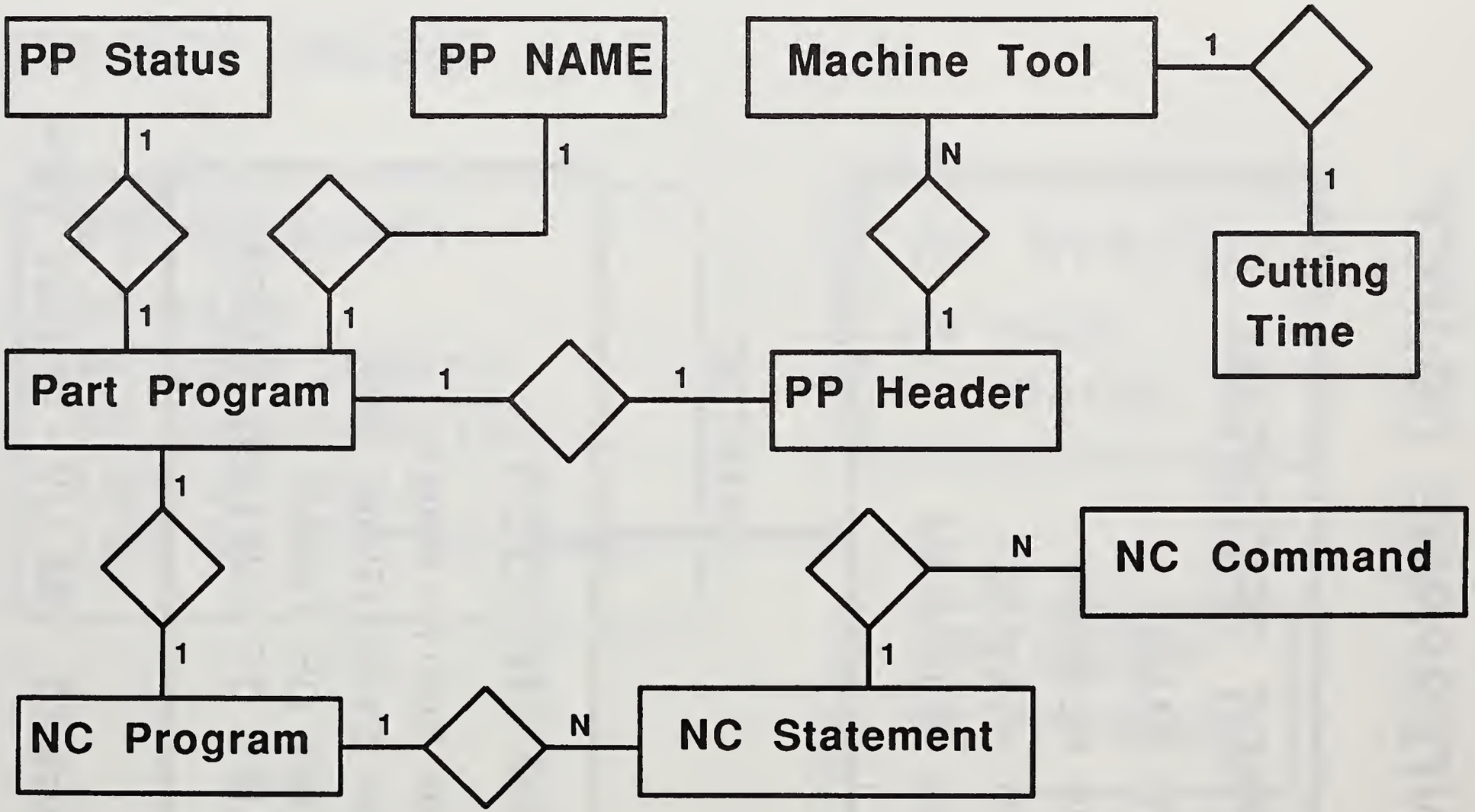


Figure 14. RM - IV - Conceptual Schema

4. Acknowledgements

The author is grateful to the AMICE team and especially to F.Naccari and M.Mollo from the FIAT Strategic Group for Development, Coordination and Control, who contributed to this presentation.

5. References

[PAN90] Panse Richard, 1990, A vendor independent CIM architecture, Proceedings of the CIMCON conference at Gaithersburg, USA.

Additional papers on CIM-OSA:

Beeckman D., 1989, CIM-OSA: Computer Integrated Manufacturing - Open System Architecture, International Journal of CIM, Vol.2, No.2, pp.94-105.

Esprit Consortium AMICE, 1989, Open System Architecture for CIM, Springer-Verlag ISBN 3-540-52058-9.

Jorysz H. and Vernadat F., 1990, CIM-OSA Part I: Total Enterprise Modelling and Function View, International Journal of CIM, Vol.3.

Jorysz H. and Vernadat F., 1990, CIM-OSA Part II: Information View, International Journal of CIM, Vol.3.

Klittich M., 1990, CIM-OSA Part III: Integrating Infrastructure, International Journal of CIM, Vol.3.

Vernadat F., 1990, Modelling and analysis of enterprise information systems with CIM-OSA, Proceedings of the sixth CIM Europe Conference at Lisbon, Portugal.

PROGRESS TOWARDS STANDARDS FOR CIM ARCHITECTURAL FRAMEWORKS

D. N. SHORTER

Abstract: Activities within the ISO and European standards-making groups on Reference Models for Shop Floor Production Standards and a Framework for Modelling CIM System Architecture respectively are reviewed. The main concepts of the Model and Framework and the ways in which they are intended to be used are presented. Lastly the way in which the Framework for Modelling can act as a basis for collaborative development within ISO and other standards-making bodies is discussed.

1. Introduction

Over the past five years and more, work on the development of architectures for CIM has been directed to various goals and these have not always been distinguished - at least in the minds of those not directly concerned with the projects themselves. These goals can include:

(i) Standards Identification
(ii) Standards Development
(iii) Systems Development
(iv) Systems Integration
(v) Systems Operations

In the standards-making world, there currently seem to be three main activities related to CIM architectures:

- in ISO TC184 SC5 WG1, on a Reference Model for Discrete Parts Manufacturing directed to (i) and (ii) above, which has resulted in an ISO technical report to be published shortly **[ISO90]**;
- again in ISO TC184 SC5 WG1, on a CIM Framework for Integration, under development;
- in the European CEN/CENELEC Working Group on AMT Architecture (WG-ARC) in producing a draft preliminary European Standard **[ENV90]** on an abstract Framework for Modelling. This is intended as an abstract framework or meta-structure for models which themselves support (i)..(v) above. The standard builds on the work of the AMICE consortium within the CIM-OSA project, itself part of the European ESPRIT collaborative research programme.

The first and third of these are nearing completion and so it is possible to report progress in some detail; the second is a more recent activity still at a preliminary stage.

2. Development of the ISO Reference Model

The work of ISO TC184 SC5 WG1 is aimed at addressing the need for a model to identify where manufacturing standards are required, the so-called Reference Model for Shop Floor Manufacturing. It is intended to guide TC184 in setting new work items to address areas where standards are missing, but it should also be useful for implementors as a systematic way of reviewing which standards might be appropriate for a specific CIM implementation.

Work started in 1984 with presentations to ISO TC184 SC5 on identified relevant standards, in particular those of IEC (especially on programmable controllers), CAM-I (Discrete Parts Manufacturing Model), AFNOR (multi-user requirements), IBM (standards, tools and interfaces for CIM) and (what was then) NBS (Architectures for CIM). This SC5 committee then established a Working Group (WG1), charged with the specific task of "developing a basic reference model, specifically to create a multi-dimensional open-ended reference model as a basis for long term planning". Initial work was to be directed at Discrete Parts Manufacture and the development of a functional model. During the first meeting of WG1 in February 1985, various specific models were discussed and gradually a limited scope of the production process was defined for further work - see Figure 1.

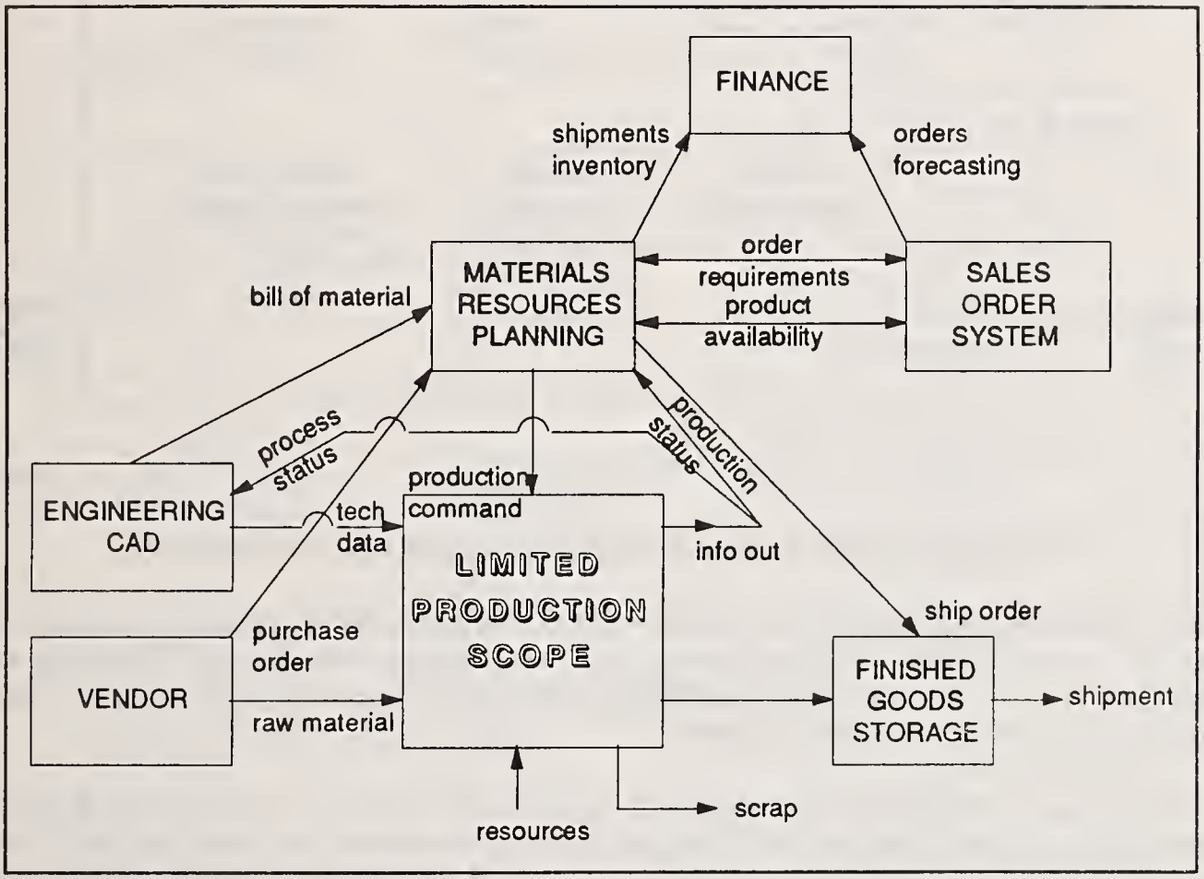


Figure 1. Limited Production Scope and its Environment

At the same time, an internal structure for this Limited Production Scope started to emerge as in Figure 2, with corresponding inputs, outputs and control relationships.

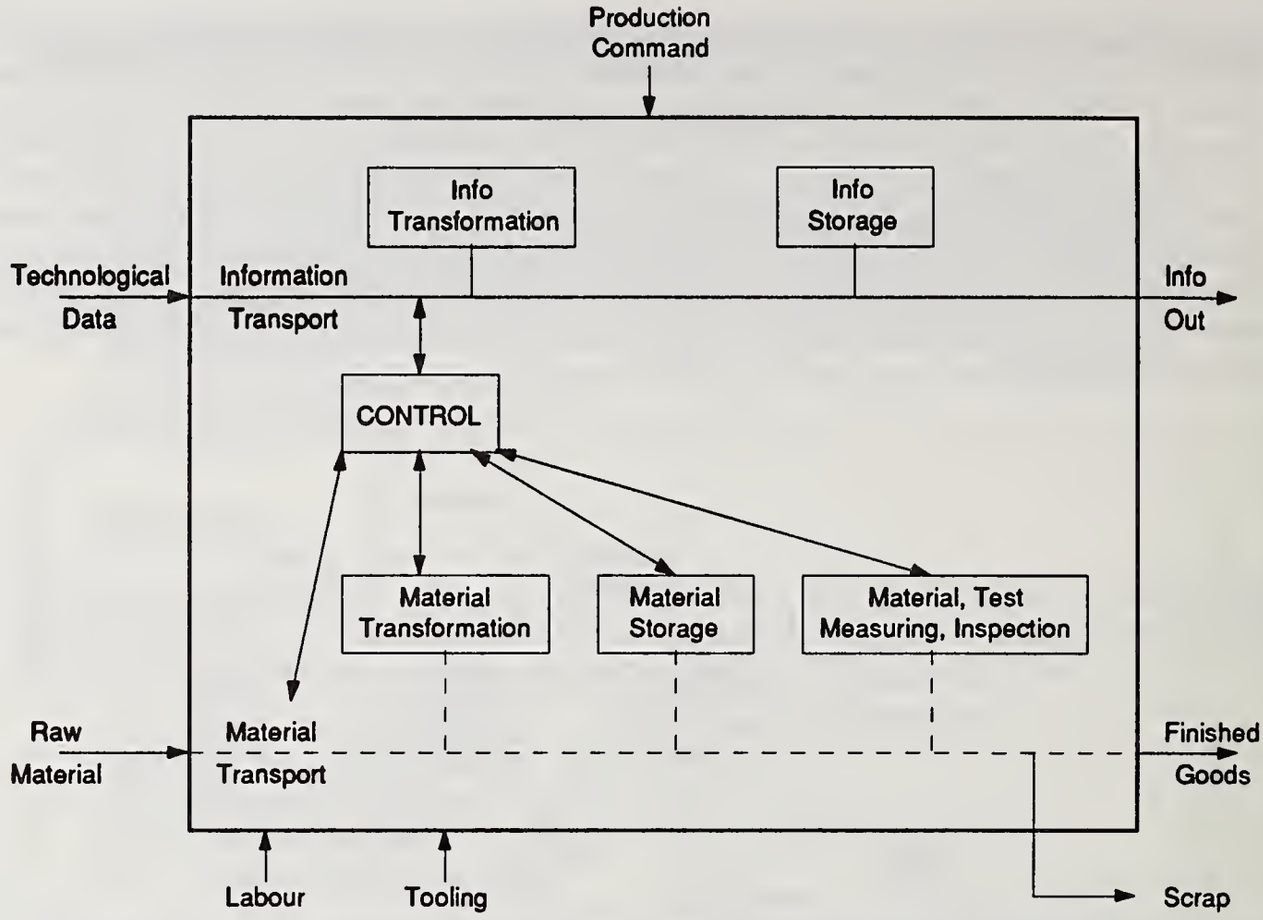


Figure 2. Limited Production Scope - initial internal structure

Later it was recognised that this initial LPS was too restrictive and in particular that there was a need for issues such as feedback, testing and relationships with external databases to be addressed. A more abstract model was needed which could accommodate emerging ideas on systems architecture (as in NBS's AMRF).

There was also a shift in how the model was regarded, from that of a structure which could be populated by existing standards, with the gaps showing where new work was needed, to that of regarding the model as an algorithm which, when applied to a particular manufacturing situation (actual or hypothesised) would identify appropriate standards. Another shift was in the recognition that standards for physical attributes needed to be accommodated, not just standards for Information Technology.

Bringing these ideas together resulted in a layering of the manufacturing process into a Factory Automation Model (FAM) which associated areas of responsibility with identified informational/ control/ physical flows as in Figure 3. This model was checked against a CAM-I analysis of data flows with encouraging results.

LEVEL	RESOURCES/ MATERIALS	CONTROL	INFORMATION
6 ENTER-PRISE		Authorise program	↑ Enterprise projections and achievement reports ↔ Master agreements ↓ Facility/Plant Mission and Objectives
5 FACILITY/ PLANT	Managed Resources	Plan Production	↑ Production and Resource Reports ↔ Technical, Production, Resource ↓ Production Requirements, Technical Data, Resources
4 SECTION/ AREA	Receivables, Work in Process, Shipping	Allocate and Supervise Resources for Production Requirements	↑ Production and Resources Status ↔ Technical, Production, Resource ↓ Production/Technical/Resource Requirements
3 CELL	Work in Process	Coordinate Multiple Machines and Operations	↑ Production and Resources Status ↔ Production Status and Synchronisation ↓ Production/Technical/Resource Requirements
2 STATION	Work in Process, Scrap, Tooling	Command Machine Sequences and Motion	↑ Command, Resource and Production Data ↔ Production Synchronisation ↓ Control Commands and Part Identification
1 EQUIP-MENT	Tooling	Activate Sequences and Motion	↑ Machine Process Condition, Status and Identification

Figure 3. Factory Automation Model

The LPS and FAM constructs are no longer an explicit part of the Reference Model, but have been carried forward, as shown in Figure 4, as a distinction between the collection of activities which are directly engaged in producing parts, that is Shop Floor Production itself, and the enterprise context in which these activities take place.

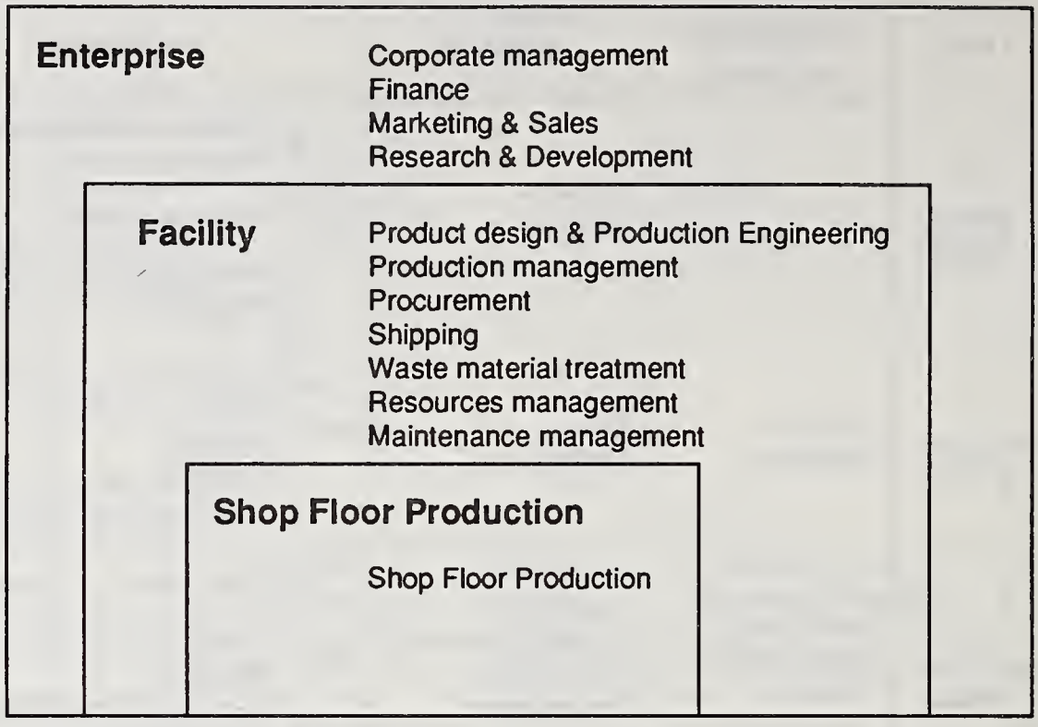


Figure 4. Typical Grouping of Manufacturing

Long (and inconclusive) discussions about the appropriate number of control levels or layers for the Reference Model led to a recognition that, for the Shop Floor Production itself, four levels were adequate for *identifying standards areas* even though particular implementations might require more or less levels.

3. The ISO Reference Model for Shop Floor Production Standards

The following is a brief description of the Model in **[ISO90]**, the use of which is covered by **[N126]**.

3.1 Scope and field of application

The distinction between the necessary characteristics of

(i) a model for identifying standards and standards requirements, and

(ii) a model which is to be the basis of a development method or actual implementation

has been a continuing theme of the ISO work.

[ISO90] states clearly in its definition of scope that it is directed towards (i) above while the European Framework **[ENV90]** is primarily addressing (ii), although the need for standards will appear as the Framework is exercised. **[ISO90]** also makes it clear that it is concerned only with Shop Floor Production, as represented in Figure 4 above, while recognising its wider context (a restriction of its original mandate).

3.2 Standards Viewpoints

The description of the model starts with the selection of nine viewpoints to guide the needs for standards in the manufacturing field, viz:

- Safety
- Environment
- Compatibility
- Performance
- Operability
- Maintainability
- Reliability
- Qualifications
- Description

These Standard Viewpoints are used in the consolidation and filtering of standards areas as described below in 3.5.

3.3 Shop Floor Production Model (SFPM)

[ISO90] then describes an abstract model of Shop Floor Production which represents those concepts of levels of control or responsibility which were found by the Working Group to be common to several systems architectures. This model is presented in Figure 5. With each of the four levels identified (Section/Area, Cell, Station, Equipment) is associated a generic type of production management activity (Supervise, Co-ordinate, Command, Control).

The activities at each of the levels have enough commonality for a general construct to represent them, as described in the next section.

	Level	Sub-Activity	Responsibility
4	Section /Area	Supervise shop floor production process	Supervising and co-ordinating the production and supporting the jobs and obtaining and allocating resources to the jobs
3	Cell	Co-ordinate shop floor production process	Sequencing and supervising the jobs at the shop floor production process
2	Station	Command shop floor production process	Directing and co-ordinating the shop floor production process
1	Equipment	Execute shop floor production process	Executing the job of shop floor production according to commands

Figure 5. Shop Floor Production Model (SFPM)

3.4 Generic Activity Model (GAM)

At each of the levels, a common Generic Activity Model as shown in Figure 6 models the activities at that level. This GAM is an abstraction which is sufficiently general that it can be instantiated at each level, that is for correspondences to be found for specific instances. These correspondences identify specific inputs/outputs (**Subjects**) and operations (**Actions**) corresponding to the general types of inputs/outputs and operations in the GAM itself. (Vertical flows are between levels, horizontal across layers.)

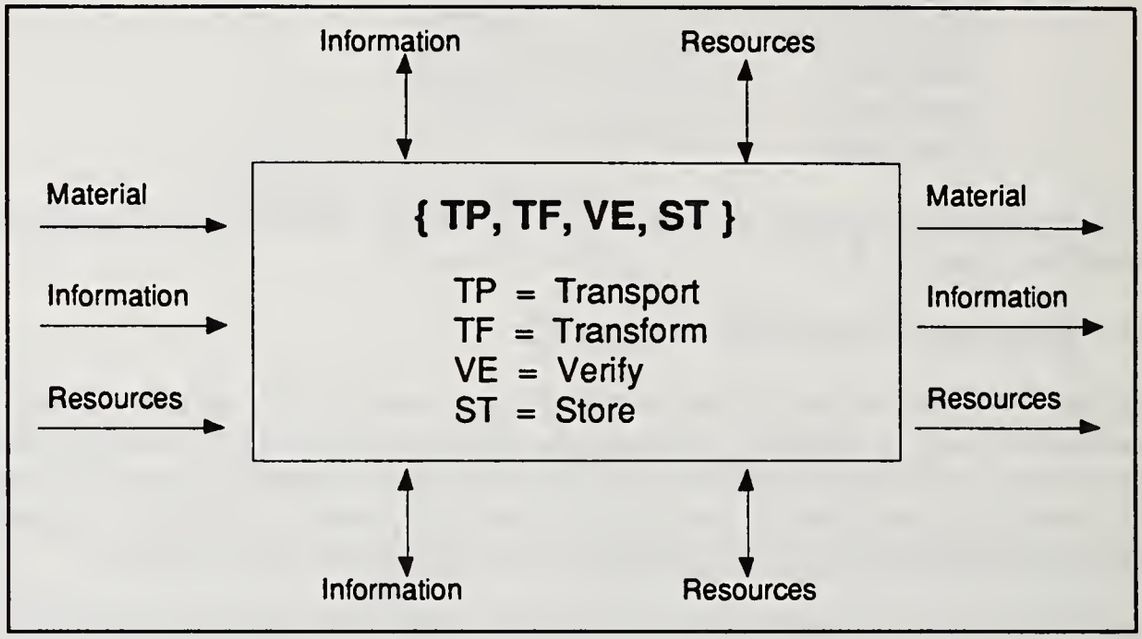


Figure 6. Generic Activity Model

3.5 Methodology

The algorithmic element of the Reference Model, that is the way in which it can be used to identify standards areas, is presented in a methodology made up of five procedures, each of a similar form. Each procedure uses (some of) the concepts of the SFPM and the GAM. The Standards Viewpoints and available manufacturing technology are then used as filters to restrict areas of standards to "those which are realisable with current technology and which reflect the objectives of standardisation" **[ISO90]**.

The procedures are of two types - those concerned with interactions within a SFPM level and those between levels or context. The procedures are represented in summary in Figure 7 and described further in 3.6.

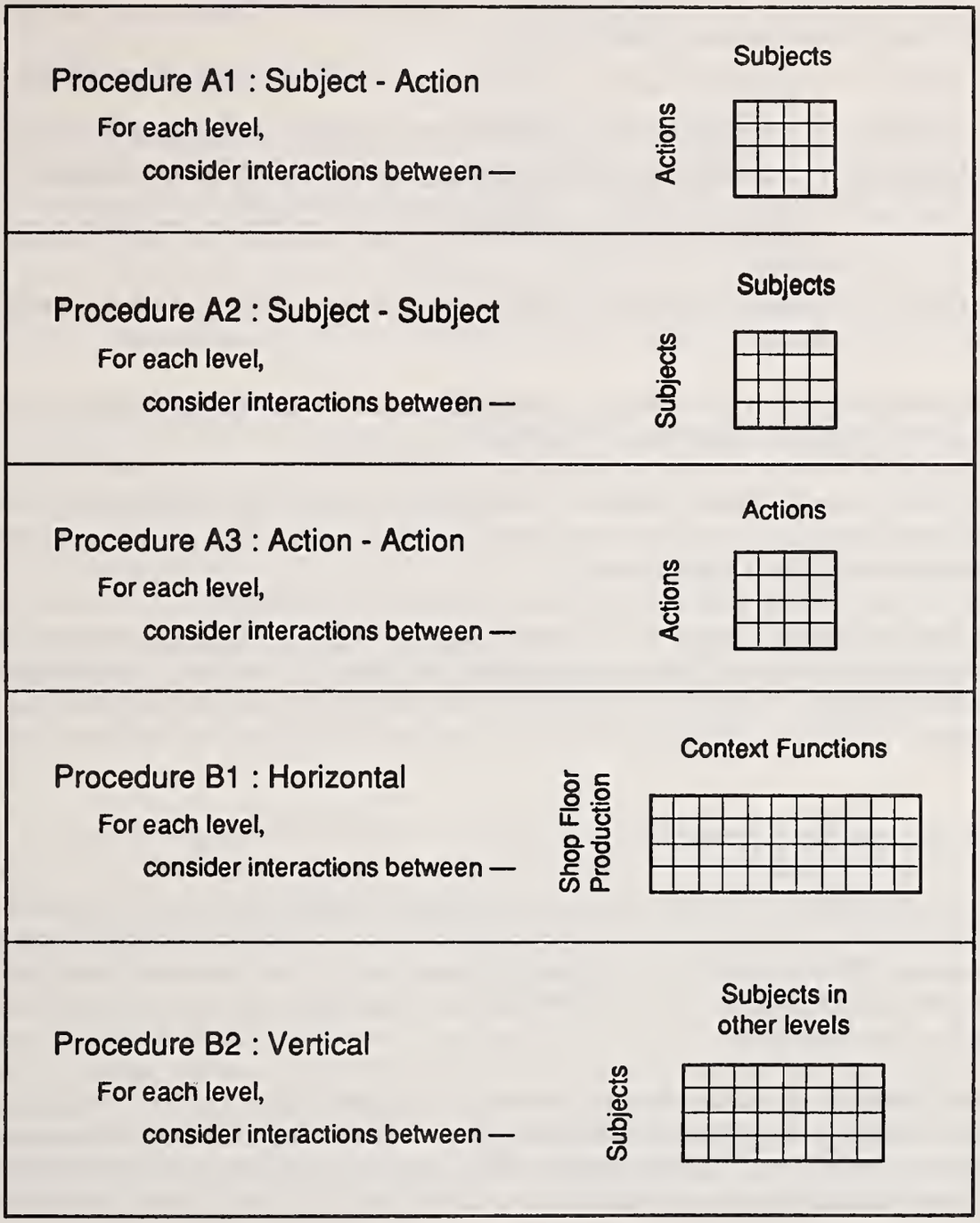


Figure 7. Matrix Representation of Identification Procedures

3.6 Development of Part 2

The Working Group is nearing completion of the second part of the Reference Model **[N126]**. This will review the constructs of Part 1 and reformulate the procedures as instances of a generic structured question of the form

"Are there or should there be **{viewpoint}** standards for significant interactions between **XYZ**, realised in **{Base Technologies}**?",

where **XYZ** represents an "inner question" which depends on the particular procedure as described below and **Base Technologies** include Information Technology, Material / Products Technology, Product / Production Engineering, Instrumentation / Control Engineering and Human Interface Technology.

The specific forms of the inner question are

SQ-A1: interactions between **{subject}** and **{action}** at the **{level}** level?

SQ-A2: interactions between **{subject}** and **{subject}** at the **{level}** level?

SQ-A3: interactions between **{action}** and **{action}** at the **{level}** level?

SQ-B1: interactions between **{subject}** of Shop Floor Production and its manufacturing context?

SQ-B2: interactions between **{subject}** at **{level}** and **{subject}** at the levels above or below?

Each type of procedure is then elaborated in **[N126]** for the various kinds of subjects, actions and levels, with examples of how it can be applied.

Having asked these questions, some way is needed to organise the large volume of information returned in the answers. So **[N126]** also defines an organisation and a labelling structure for the standards areas that have been identified.

At the time of writing, the body of the present draft is being reworked to contain examples of identified standards areas in general form. Specific ISO and IEC standards and standards-making activities will then be listed in an Appendix, whose status will be that of a time-stamped 'snapshot'.

4. European Developments

While the ISO work was proceeding on a Reference Model for identifying standards areas, significant interest had been building up in Europe to work on developing suitable architectural structures for CIM integration and a Working Group on CIM Architecture (WG-ARC) was set up to report to CEN/CENELEC, the Joint European Standards Institution, on the way forward for European standardisation.

WG-ARC started by carrying out an evaluation of major development initiatives and their potential contribution to European Harmonisation, and identified CIM-OSA (Esprit project 688, AMICE) as the most promising starting point for a preliminary European Standard (ENV) in this area. This evaluation has been published as a CEN/CENELEC Technical Report **[RIT89]**.

At the same time the AMICE consortium which had produced CIM-OSA agreed to release certain deliverables to the Working Party, which would use then them as a basis in developing the preliminary standard.

One promising route for development of a CIM system is to use an integrated and coherent model as the basis of an executable system. Because of the scale and range of types of CIM systems, it is important to use standard (common) model elements wherever possible, but not at the expense of meeting the particular goals of the business. So a framework is needed to describe the development process in such a way as to encourage the identification of standard and executable model elements. In the AMICE CIM-OSA project this is the "Framework for Modelling", which has been agreed by WG-ARC to be adequate to express common concepts of a number of different architectural approaches. This has been the starting point for the development of **[ENV90]**, as described below.

5. The European prENV, "Framework for Modelling"

5.1 Background to the prENV

A "prENV" is a preliminary European Norm (standard) published by CEN/CENELEC. It is a provisional standard for fields where a high rate of technical innovation is envisaged, and a review process has to start two years after initial publication. By the end of three years, the prENV has to be converted into a full European Norm, allowed to continue for a further two years (only once), revised into an improved ENV or withdrawn. Because of this revision process, it should be relatively easy to keep the prENV in step with international development. (When an ISO standard is published covering the scope of an ENV or EN, that European standard will be withdrawn.)

The prENV is intended to guide the structuring and development of those standards necessary to achieve *integration* in Discrete Parts Manufacturing. The work is based mainly on fundamental CIM-OSA architectural concepts but has taken note of other work where this was helpful, such as the Generic Activity Model from the ISO Reference model **[ISO90]**.

Formally the work is "to provide a standard for a framework, which will serve as the common basis for identifying and coordinating standards development for computer-based modelling of Enterprises, focussing on Discrete Parts Manufacturing. Models generated by using this framework will ultimately be computer-executable and (will) enable the daily operations of an Enterprise to be run, monitored and controlled by such models". The framework is very general however and WG-ARC considers that many of the concepts may well be appropriate to other types of business enterprise.

A manufacturing enterprise lives or dies in a changing world - it needs to survive in a continually evolving economic and technical environment, with increased integration necessary to achieve fast product cycles and competitive production costs. Information Technology is necessary for CIM, but until now the development methods, infrastructure, products and actual implementations have been insufficiently flexible and incapable of supporting the necessary pace of change. Incompatibilities between partial solutions running on different computer systems are a continuing problem. Producing standards to ease these problems requires a mechanism to ensure that compatible standards can be developed in parallel. CIM users and vendors also need to be able to plan CIM developments. The integration part of CIM needs to encompass physical system intercommunication (OSI), application integration and business integration. Lastly the complexity of the necessary integration and the need for continual evolution demands maintainable computer-based executable models - and this has been accepted as a long term objective of the work.

An Architectural Framework is regarded as a structure of parts such that constructs formed from those parts have certain characteristics, e.g. of consistency and completeness. It is helpful if these parts can be described separately without overlap. They need to be sufficient for the purpose (ensuring the required characteristics) and economic in concept.

This Framework for Modelling (Framework) uses three dimensions (each reflecting a particular concern) and a development process corresponding to each dimension. It assumes the existence of a supporting Information Technology Infrastructure, as illustrated in Figure 8, to design a computer processable model (building on concepts from the Framework) and to control the operations of an Enterprise using that model - however that supporting Infrastructure is not described in the prENV.

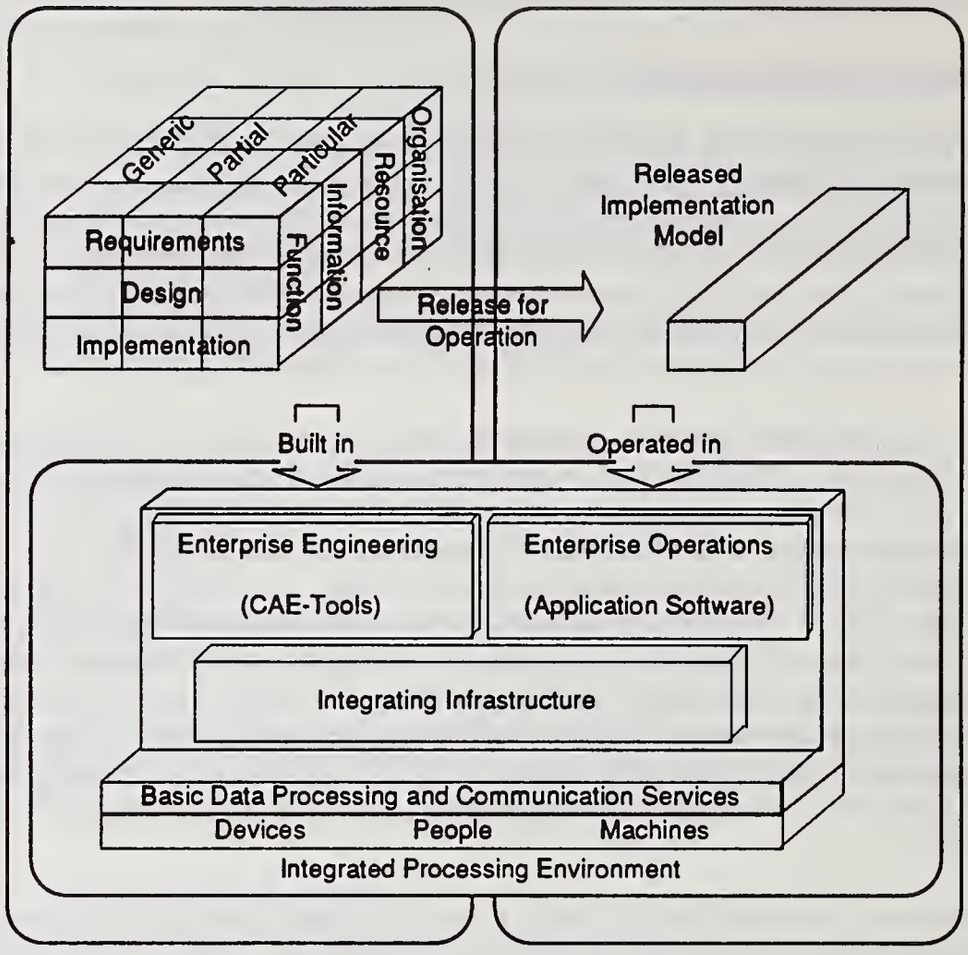


Figure 8. The Supporting IT Infrastructure

Describing the Framework ideally requires the three dimensions to be described at the same time - explaining the concepts of any one dimension generally requires concepts from the other two. The prENV itself includes forward references to address this problem.

5.2 The dimensions and concepts of the Framework for Modelling

(Note - much of the following material is drawn directly from **[ENV90]**)

5.2.1 Architectural Level

This dimension is illustrated in Figure 9. It takes account of the economics of the market allowing CIM suppliers to generate products of maximum applicability while recognising that CIM users need solutions tailored to their specific needs. The most abstract level (*generic* architectural level) defines basic constructs needed for expressing requirements, design and implementation elements, and incorporating functional, information, resource and organisational aspects. The basic constructs themselves are not part of the prENV, but are being developed as part of new standards work, again building on CIM-OSA concepts.

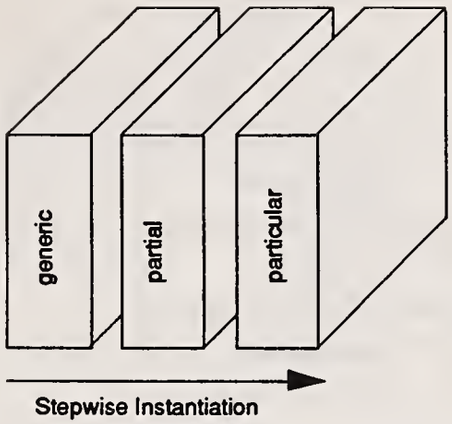


Figure 9. Dimension of Architectural Genericity

The more commercially significant level is that which uses these basic constructs to form models for particular market sectors, the so-called *partial* architectural level. Here partial models will be defined for those requirements, partial designs or implemented packages which are common to a specific market sector. Again these partial models will generally address functional information, resource and organisational aspects.

Lastly the *particular* architectural level is that at which a designer or user of a particular enterprise expresses its specific workings, including requirements, design and actual implementation. As partial models become commercially available, this particular level will increasingly make use of these, adding and modifying components only where necessary.

This process of moving from abstract construct through to partial models through to particular models for a specific manufacturing operation is given the name of "*stepwise instantiation and aggregation*", expressing the idea of increasing specialisation and grouping into sector-specific applications.

5.2.2 Modelling Level

The life-cycle of CIM development is complex and on-going (as new processes become available and new products are required). So the three major project phases of identifying requirements, producing a design and implementation are given explicit expression in the Framework, as illustrated in Figure 10. This allows constructs and techniques to be modelled which recognise the specific requirements of each phase and the transition processes between them.

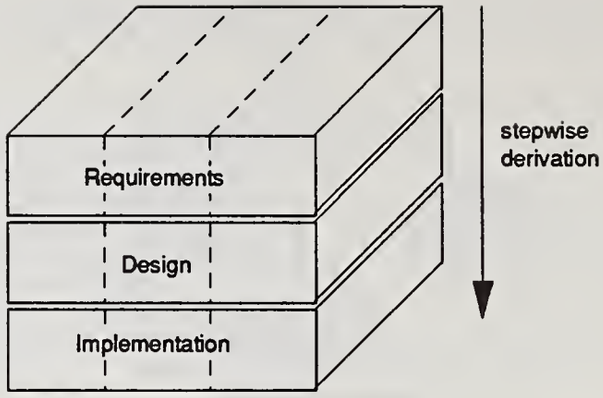


Figure 10. Dimension of Modelling

Firstly the *requirements* modelling level describes the business requirements of the Enterprise and its operations using concepts from the generic architectural level and without (as far as possible) making decisions on designs or implementations to achieve these.

Next the *design* modelling level describes the actions and the processes that will need to be performed to meet the business requirements. It does not say how these are to be executed but it does allow the designer to take into account the interactions between requirements generated from different users, trade-offs, optimisations, constraints etc.

Lastly the *implementation* modelling level selects, or specifies for development, the particular components to be used to realise the design and hence meet the requirements for enterprise operations. These information and manufacturing components are also described in the prENV (see 5.2.4) and are equivalent to the *base technologies* of Part 2 of the ISO Reference Model (see 3.6). They include all the information or material processing elements that operate on information or physical objects and so include human resources, computers, machines, programmes and data etc.

The progression from requirements through design to implementation is called "*stepwise derivation*". As the use of the Framework increases, it is hoped that generic components, partial models and capabilities of actual existing components will increasingly be described in computer processable form, allowing automated assistance to the designer in this progression.

5.2.3 Views

At each level of architectural generality (generic, partial, particular) and at each modelling level, because of the complexity of the interactions between the elements which specify requirements, describe designs or specify implementable elements, it is helpful to focus on a small number of particular aspects of those elements, and to consider relationships between elements from that point of view.

This is not new - many of the existing tools for systems analysis recognise the benefits of separately analysing functional elements, management structures, data or process flows and it is helpful to align the Framework with these existing procedures.

These requirements are recognised in the dimension of views, illustrated in Figure 11. Four enterprise views have been selected, although experience may suggest that others might be required, or indeed that fewer might be sufficient. The current set is described below.

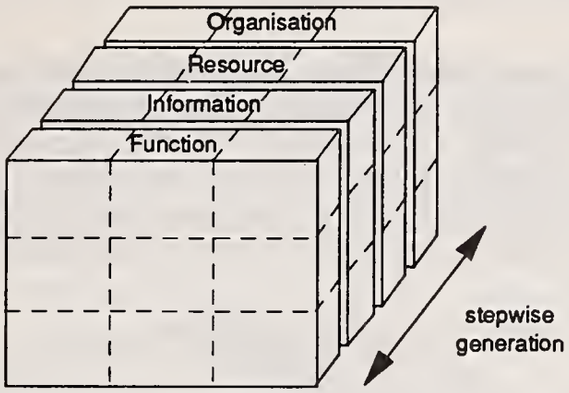


Figure 11. Dimension of View

The *function* view describes the (requirements, design or specifications of) functional processes, their inputs and outputs (including information) and the constructs under which they operate (e.g. pre- and post- conditions). The function view uses other elements of the Framework when it can, e.g. building blocks described at the generic architectural level, models from the partial architectural level etc.

The (requests for, characteristics of) information objects identified at the functional view are further described in the *information* view which characterises the necessary information building blocks and their relationships. How this is done will differ with the architectural or modelling level. At the implementation level for a particular enterprise, the information view will need to produce a hierarchically structured and eventually detailed specification for the information processes and data necessary to realise CIM.

The *resource* view emphasises the resources needed to execute enterprise operations. It also explicitly represents these as information objects (and so there is some overlap with the information view).

Similarly the *organisation* view describes the hierarchy of command - that is the management and control structures, and responsibilities to be undertaken by each management or control function. Again these responsibilities are explicitly represented as information objects.

The process of representing these views ("*stepwise generation*" in the Framework) will in general proceed in an interactive manner because of the inter-relationship between views. A decision taken in one view (e.g. on a specific requirement, on a particular implementation) may well influence decisions that have been or will be taken from another. So iteration will be needed to check consistency and completeness between the views. Explicit representation of resources and organisation should eventually allow automated assistance.

5.2.4 Information and Manufacturing Technology Components

Information and Manufacturing Technology components are required to "transform, transport, store and verify" data, material and products, as recognised in the Generic Activity Model of the ISO Reference Model (see 3.4). These components, such as basic data processing resources, application software, machines with associated control data, robots, human resources etc. are of course an essential part of a CIM system. They are identified as necessary in the Function View, later specified in the Resource View and finally implemented at the Implementation Modelling Level.

5.3 Using the Framework for Modelling

It is not intended that everyone in the CIM community should be concerned with each of the thirty six cells that can be constructed by elaborating the three dimensions - see Figure 12. For example, standards-makers will primarily be concerned with defining and representing objects at the generic level. Vendors of products in specific industry sectors should be using such standard objects in developing and describing new products and partial models. A designer will generate views for a particular enterprise, using standards, products and partial solutions where appropriate.

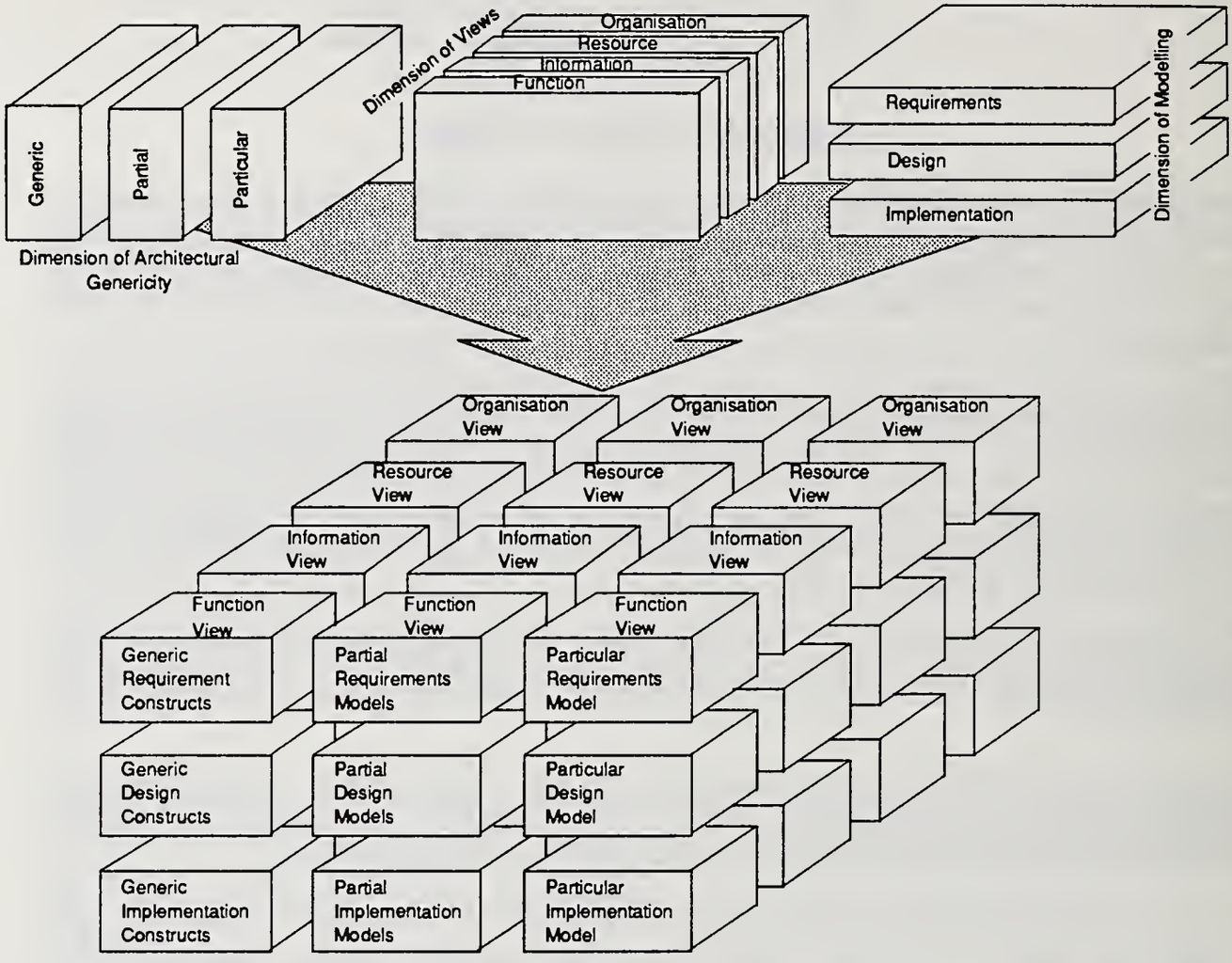


Figure 12. The exploded Modelling Framework

5.4 State of the prENV work

The prENV itself was circulated to CEN/CENELEC members for comment in January 1990 with a closing date of 15 March 1990. At the time of writing, voting is scheduled to take place at a meeting on 18-19 April. The prENV and many of the underlying CIM-OSA concepts were also presented, essentially as described in this Section 5, at a CIM workshop in Brussels on 7-8 March 1990. The general conclusion at that workshop seemed to be that the prENV was accepted as a very general structure which will support the development of more detailed modelling structures at those combinations of genericity, modelling level and view which turn out to be helpful in supporting concerted action to develop standardised elements.

It now seems that, if approved, possibly with modifications, the Framework for Modelling will provide an enabling mechanism for the co-operative development of the necessary standards to support the needs of both user and vendor for CIM integration and CIM evolution.

6. The relationships between the ISO and CEN/CENELEC Working Groups on CIM architectures.

It is important to be clear about the role or purpose of an architecture - what it is intended to ensure (what style, what kind of structure it can generate) and what it is intended to prevent. An architecture in the public domain, especially if it has the status of a standard, must not pre-empt competition (e.g. by embodying proprietary material) or (as far as possible) prevent the use of more advanced implementation technologies than those available today.

At this stage of understanding, a CIM architectural Framework for Modelling, or a Reference Model for identifying Shop Floor standards should be seen as supporting a process, rather than as fixed entities. That is, they should:

- support the formation of an industry (and standards-making) consensus,
- promote a public discussion (using the same words for the same concepts - a vocabulary issue) on the more detailed constructs that are required, and
- encourage parallel and consistent development of these constructs.

From Europe, the current prENV is seen as a start in this process. It has some gaps, eg it assumes the existence of a supporting Information Infrastructure (Figure 7) but does not describe it. The constructs are very general, which is necessary to ensure openness and acceptability today. However, in the future that generality needs to be reduced as a consensus emerges for more detailed constructs, and to increase confidence that (elements of) models generated using the Framework are sufficiently consistent in the sense of capable of a sufficient degree of interworking.

Developing CIM architectural constructs and methodologies which will ensure components can be integrated, modified and realised across a range of implementation technologies is a major challenge which requires a strategy to ensure efforts and other resources can be harnessed to a common goal.

Both the draft prENV itself and other working material have been presented and made available as working documents to ISO TC184 SC5 WG1 in support of the relatively recent work item on CIM Systems Integration. It is hoped that further discussions inside ISO TC184 SC5 WG1 will build on this and other work, identifying any areas where the prENV has deficiencies which need to be addressed in the next revision and identifying areas where new work items are necessary. In this way the prENV should be a basis for international collaborative work, gathering up contributions within a common framework and speeding the development of new and necessary standards.

7. References

[ENV90] CIM System Architecture Framework for Modelling, Draft European Pre-standard, CEN/CENELEC prENV 40 003, January 1990

[ISO90] Technical Report TR10314 - Reference Model for Shop Floor Production Standards, Part 1, ISO 1990 - to be issued

[N126] Document N126 is a draft working paper of the Reference Model for Shop Floor Production Standards, Part 2, on the application of the Reference Model for Standardisation Methodology Industrial Automation Shop Floor Production Standards - currently under development by ISO TC184 SC5 WG1

[RIT89] Evaluation Report on CIM Architectures, CEN/CENELEC Report R-IT-01, 1989

DESIGN TO PRODUCT AND ESPRIT 384 TWO ROADS TO OPEN CIM

P.A. FEHRENBACH AND S.P. SANOFF
GEC-MARCONI RESEARCH CENTRE, UK

Abstract

Design to Product and ESPRIT 384 are two European projects that have developed open CIM (computer integrated manufacturing) architectures over the last five years. During that time the CIM world has changed a great deal and the two projects have adapted their architectures to suit. Design to Product changed from being closed and monolithic to an open, distributed architecture. ESPRIT 384 capitalised on the flexibility of its rapid prototyping approach to incorporate change quickly. Technical, financial, managerial and cultural factors all influenced the roads that the projects took and all of these are discussed in the paper in the context of manufacturing industry at large.

1. Introduction

In the past two or three years the ideal of the *open* system that can incorporate software and sub-systems from a wide variety of sources, all communicating via a standard interface, has gained acceptance and popularity in many areas of computing, including computer integrated manufacturing (CIM). It has not always been so. Co-operation between vendors over standards, such as the manufacturing automation protocol (MAP), is a recent development, prior to which each was happy to lock customers into their own proprietary standards and systems. The change in attitude of vendors and other important changes in the computing industry and manufacturing industry in Europe and North America at large, have occurred during the course of two important CIM projects in which the authors have been involved. The projects have been greatly influenced by the changes taking place around them and by their own experience and tell a story of adaption and development that we believe holds valuable lessons for the future.

The GEC-Marconi Research Centre (MRC) is the research arm of GEC-Marconi Ltd., the United Kingdom's largest manufacturer of electronic systems for all types of civil and military application. It has been involved in robotics and automated assembly systems since 1979 and in the wider aspects of CIM since the early 1980s. GEC-Marconi is a large organisation manufacturing a very wide range of high value products, often in small batches. It has substantial investment in existing computer aided design and manufacturing systems from a number of different vendors and would wish to incorporate these in any all-embracing CIM strategy that it adopted. At MRC we take the view that CIM should encompass all facets of a manufacturing enterprise's activities, not just design and manufacture, and we try to design systems that, although not comprehensive in themselves, fit in with this ideal and can be extended as and when required to meet future needs.

We are in the business of designing and prototyping CIM systems and sub-systems that we believe would be of use to GEC-Marconi or to outside customers. These prototypes are presented as demonstrators to stimulate interest from the company's product divisions but are not themselves put to use. Our demonstrators need further work to turn them into robust systems for real use. In this paper we describe two such demonstrators; both were collaborative projects, one of which we led.

2. The Alvey *Design to Product* large scale demonstrator project

2.1 The Alvey programme

The Alvey Programme of pre-competitive, collaborative research in advanced information technology was set up by the UK Government in 1983 with a budget of £350 million to be spread over

five years. The programme was focused on the enabling technologies of intelligent knowledge based systems, man-machine interfaces, software engineering and very large scale integrated circuits. In addition there were a small number of large scale demonstrator projects that were intended to pull through the results of the enabling technology work and demonstrate them in important applications. One of these was called *Design to Product*, DtoP for short, and was in the field of CIM.

2.2 Genesis of the project

In 1983 MRC and the Department of Artificial Intelligence at the University of Edinburgh had been trying for a year or so to secure funding for a joint research proposal called *Design and Make*. The idea was to develop a new type of artificial intelligence (AI) based computer aided design (CAD) system and link it to a robotic assembly system. The CAD system would be used to design assemblies of components based on knowledge of component function and the robotic assembly system would then automatically assemble the design.

One of the funding bodies approached suggested that the idea would form a good basis for a large scale demonstrator proposal under the embryonic Alvey Programme, provided that it was expanded in scope and participation. So the process of forming a consortium to propose the large scale demonstrator began.

2.3 The proposal

The project proposal was written over a five month period in 1983-84, during which the size of the project consortium grew to a maximum of eleven members before settling down to eight. There were major structural changes in the consortium as it was formed. Responsibility for managing the project was handed over to a company that designs and sells manufacturing systems and that would be in a position to exploit the results at the end of the project, as required by the Alvey Programme. A user collaborator was appointed to provide a demonstration base facility and example product for the project to focus on. Collaborators were drawn from the industrial and academic spheres and joined the consortium on the basis of having some past experience or new ideas to offer that filled an otherwise awkward gap in the consortium's credentials. Design of the consortium was very much *bottom up* rather than *top down*.

The proposal was for a five year project costing £8.7 million. But five years is a long time for eight collaborators to stay together on the basis of a plan worked out before they started, so we divided the project into two halves and only planned the first half in any detail. We expressed the goals of the project in suitably vague terms so as to allow flexibility. The most important goal for us was to demonstrate support for the lifecycle of light, electro-mechanical products from design, through manufacture to service using the techniques of artificial intelligence. The user collaborators were manufacturers of diesel fuel injection equipment, so the chosen product on which to focus our attentions was a fuel injection pump comprising several hundred components.

2.3.1 The first system concept: a monster is born

In 1983-84 the talk amongst people with interests in CIM was all about automated, unmanned factories that would operate 24 hours a day, 365 days a year. These factories would accept raw materials through one door and despatch finished products through another. They would be flexible, efficient and trouble free. We were party to this vision of the future and wrote it into the DtoP proposal. We produced a concept drawing showing a spacious factory with futuristic machines tended by automatic guided vehicles. The only signs of human presence were some empty chairs in the design office. We also built a wooden model to make the point more forcibly.

All the project members were technologists and we proposed a wholly technological solution to the CIM question. What is more, we proposed a solution that, although complete in itself, relied on every part functioning correctly in order to work and that would find it difficult to communicate with systems outside. Our first stab at a system block diagram is shown in figure 1. This was originally compiled as a representation of the CIM problem but came to be presented as a possible system solution. It is remarkable not for the careful thought that went into deciding what each box should contain and how they should be connected, but for the sheer number of boxes, the complexity of their interconnection and the lack of any apparent structure. We had proposed a monster and did not yet know that it was of the dinosaur variety.

2.4 A new collaborator brings a human face

As a condition of funding the project the Alvey Directorate insisted that we take aboard an additional collaborator expert in human factors to bring a more humane approach to our goals. We duly signed up a new collaborator in the form of an academic research centre specialising in human sciences and advanced technology. They used strange terms like *usability*, *utility* and *socio-technical* that we did not understand. They asked how we were going to care for the people who would work with our CIM system and we said that there were no people. It was clear that there were going to be some arguments.

Our new friends conducted a human factors feasibility study to carve out a role for themselves in the project and got the budget increased to cater for it. They identified dozens of areas where they felt they could make a contribution and presented their findings to the project. The reception was polite but not enthusiastic and our new friends began a long task of persuading and cajoling the rest of us into accepting their ideas.

2.5 The pilot phase

The first half of the project was dubbed the pilot phase and got underway early in 1985 after seemingly endless negotiations over contracts and collaboration agreements. Our objective during this period was to demonstrate independently the viability of the different product support sub-systems that would eventually link to form the integrated Design to Product system. The consortium had been put together bottom up with each member corresponding to a particular phase of the product lifecycle, e.g. design, process planning, assembly, maintenance. So too, therefore, was the workplan for this first phase. Each collaborator got on with their own part of the system in their own way and at their own pace. Close collaboration was not required for individual success and was therefore not manifest. Collaboration proceeded at the level necessary to ensure that the project would meet its overall objectives at the end of five years.

The vagaries of the funding arrangements for the academic collaborators meant that some were able to start work much earlier than others. Those who started early then needed to take decisions before the project as a whole had caught up with them and this was to prove a problem for us when we came to integrate our efforts later. Initially some of the industrial participants found the academics difficult to understand and some academics found the industrials frustrating. The pilot phase brought all the collaborators closer together in understanding and engendered a common outlook amongst us. It ended in late 1987 with successful demonstrations of all the system components.

2.6 The demonstrator definition phase

The pilot phase did not produce an architecture for the DtoP CIM system; it had no need of one itself (though more on this point later) and was about demonstrating not defining. We believed the tasks of defining the system architecture and the work of the second phase of the project so important that we set up a separate activity, called the demonstrator definition phase, to do them. This activity ran for two years from the beginning of the project and was staffed by one person with specialist

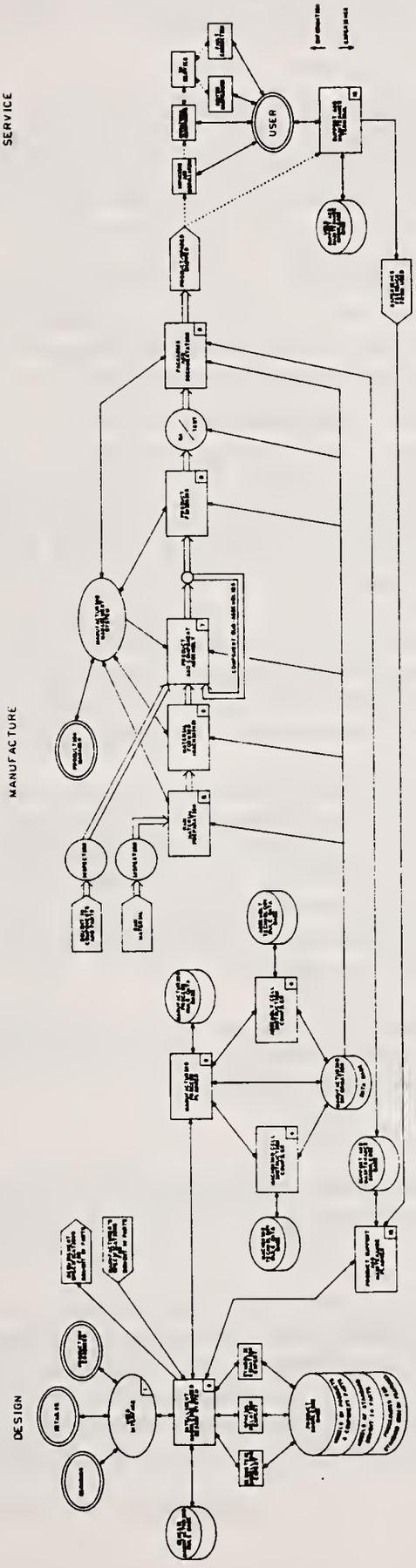


Figure 1. The first DtoP system concept

help as required. This person also had responsibilities for liaison with other projects and for spreading technical information round the project from outside, so it was not two person years on architecture and workplan alone.

The demonstrator definition phase needed to produce a new system architecture for use in the second half of the project. We needed a practical architecture that it was realistic to build in the remaining time with the resources available. Our monitoring of the world outside the project showed that open, distributed architectures were in the ascendent and that large, monolithic systems were rapidly falling out of favour. The costs of building large CIM systems were also being appreciated more acutely throughout the industry and the trend was distinctly towards modular systems that could be built up incrementally from small beginnings. This led to a scaling down of our aspirations towards automation in all areas of product support, particularly manufacture and service. Our human factors collaborator had been hard at work and the importance of all types of system users was beginning to be realised, their presence being very necessary as the goal of total automation receded.

2.6.1 The second system concept: the monster is no more

Our new system architecture is shown in figure 2. It is very different from figure 1 and the changes reflect those in the world and in our own thinking. First, it is much simpler. There are fewer boxes and fewer interconnections. Some of the boxes are floating, apparently without any connections, reflecting a move towards a blackboard style of working. Second, the knowledge bases and control modules that are the necessary infrastructure for a complex system take a more prominent position. Third, the users are now considered to be part of the overall system. Fourth, the product design functions appear to be assuming dominance over the manufacturing and service functions, which occupy a comparatively small fraction of the diagram. This is because we found that our original ideas for manufacture and service support, which required the purchase or construction of a large amount of hardware, were too costly for the project to support, so work in those areas was reduced.

Some of the changes highlighted above are reinforced by the system design concept shown in figure 3 that is contemporary with figure 2. This gives more details of interconnection between parts of the system and strongly emphasises the central role of knowledge bases and their management systems. Both figure 2 and figure 3 show very little structure to the system, in keeping with the blackboard paradigm. This has the effect of reducing dependence of each part of the system on the others and also makes it easier to extend because the interconnections are not too complex. Thus the goals of modularity and openness are achieved.

We therefore entered the second half of the project with our aims intact, if reinterpreted to scale them down, and a system architecture appropriate for our purposes and in accord with, indeed we would like to think ahead of, the wisdom of the time.

2.7 The full demonstrator phase

The second half of the project had the tasks of integrating the results of the first half, which had been developed separately, and demonstrating them as a single, coherent system. It was called the full Demonstrator phase.

The new project system architecture (figures 2 and 3) depended on centralisation of knowledge and knowledge management while encouraging distribution of system function. During the pilot phase we had not done anything towards implementing this system infrastructure and so work had to proceed in parallel with the integration during the demonstrator phase. We were given a head start in the information management area by basing our work on an AI toolkit known as KERIS [POU89] that we were able to modify, but most of the infrastructure work had to start from scratch.

Not all the work done during the pilot phase was amenable to integration into the new architecture. To overcome this we did some reimplementation of pilot phase work and modified the architecture where necessary. As real implementation problems were faced, as opposed to conceptual ones, many decisions were taken that affected the architecture. A decision to adopt Sun Microsystems's network extensible windowing system (NeWS) meant that the system could easily be run over a network of computers and that software functions could be split into smaller units and run separately. Thus the user interfaces for a number of the software modules were separated from the rest of the software to facilitate change in future.

It was not long therefore before a revised architecture was produced by the team building the system infrastructure to reflect these changes.

2.7.1 The final system concept: deceptively simple

The revised and final system architecture shown in figure 4 gives central status to the *Tool Manager*, reflecting the preoccupations of its progenitors. The tool manager is like a software telephone exchange that puts other parts of the system, known as tools, in touch with each other. The most important tool is the information management system, which presides over the system's product description and engineering knowledge bases. It is here that all the information on products and how to support them is stored. We have grouped other tools in the diagram to try to given an impression of system structure, but the tool manager has no favourites and treats all tools equally.

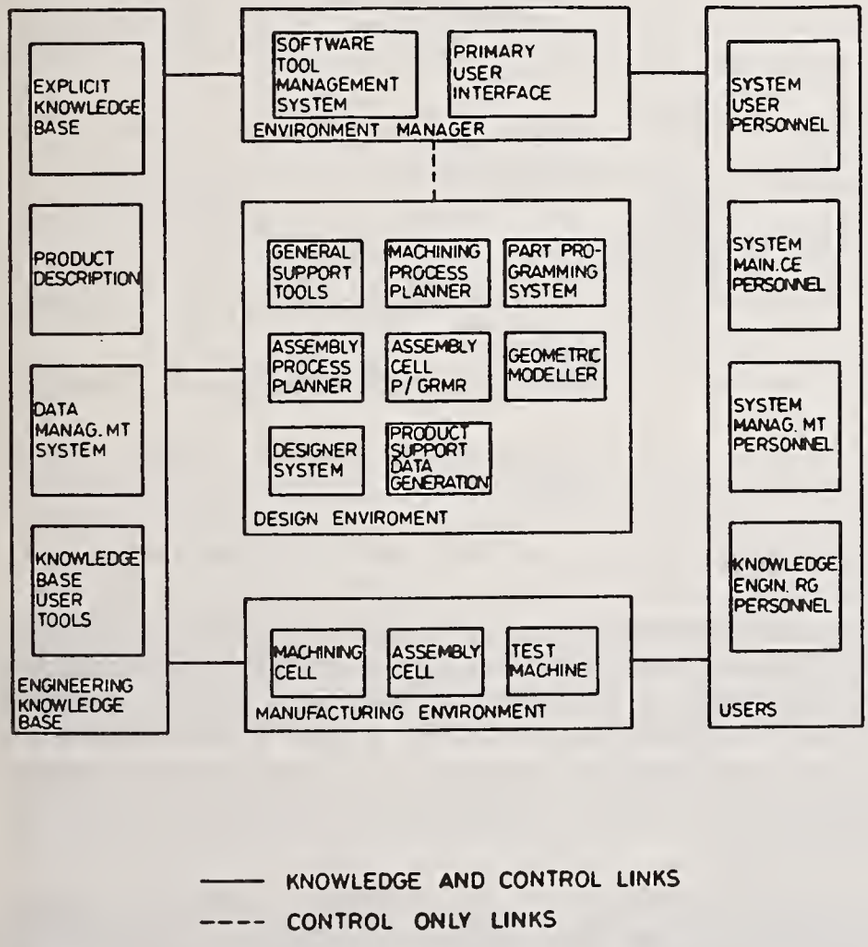


Figure 2. The second DtoP system architecture

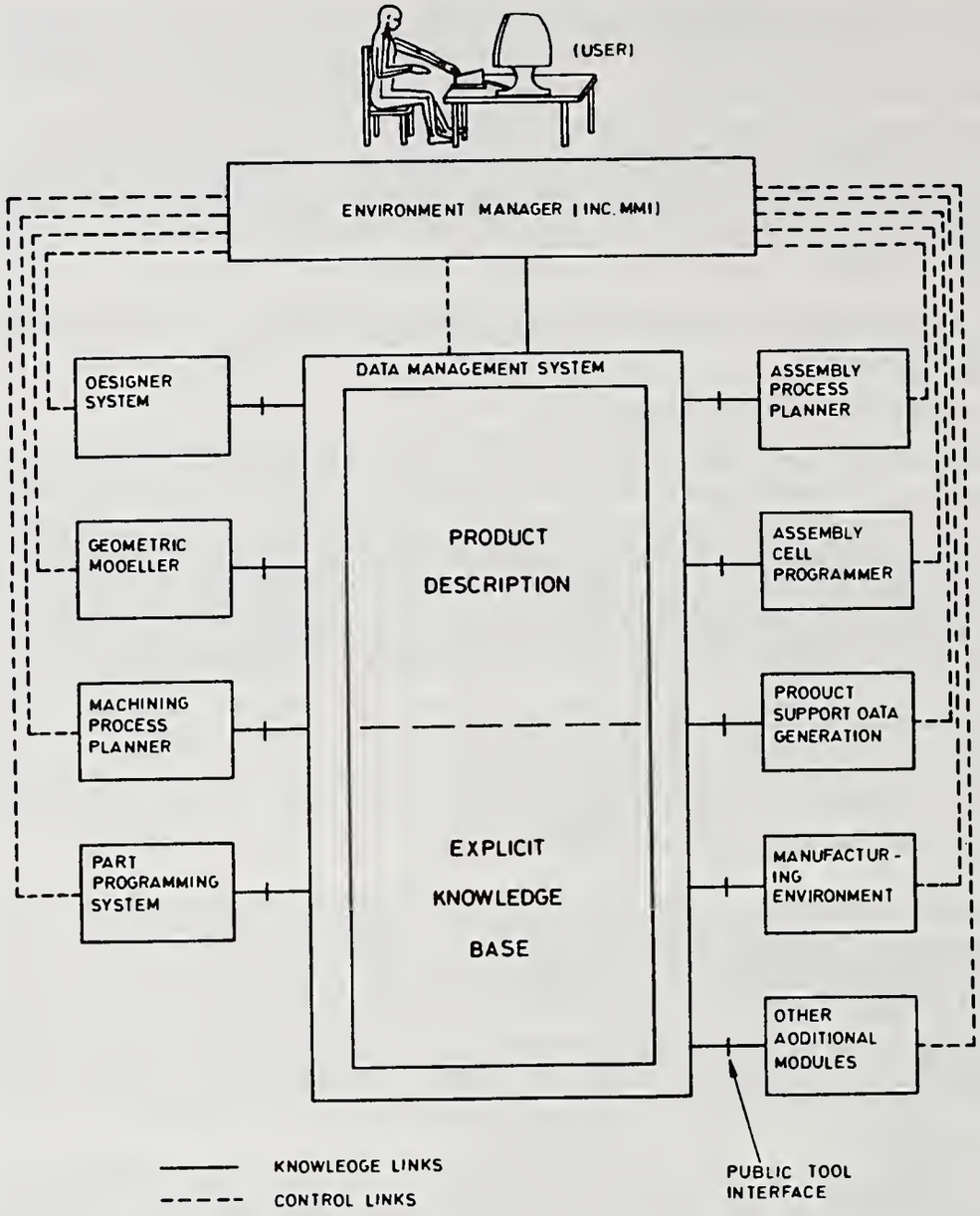


Figure 3. The second DtoP system design

The major group of tools are known as the design support tools and have consumed the greater part of the project resources. They support product specification, design, machining process planning and numerical code generation, assembly planning and robot programming, geometric modelling and documentation functions. Each tool has a user interface which is itself a tool and shown in another group. The separation of the tool user interfaces is a major development from the previous system concept. It allows the interfaces to be modified more easily to fit in with changing user requirements or to conform to a house style.

The manufacturing environment, which comprises a factory area controller and cells for machining and assembly, is another tool in the system, though quite different to the purely software design support tools. Finally we have included some utility tools that provide help, error reporting and other functions.

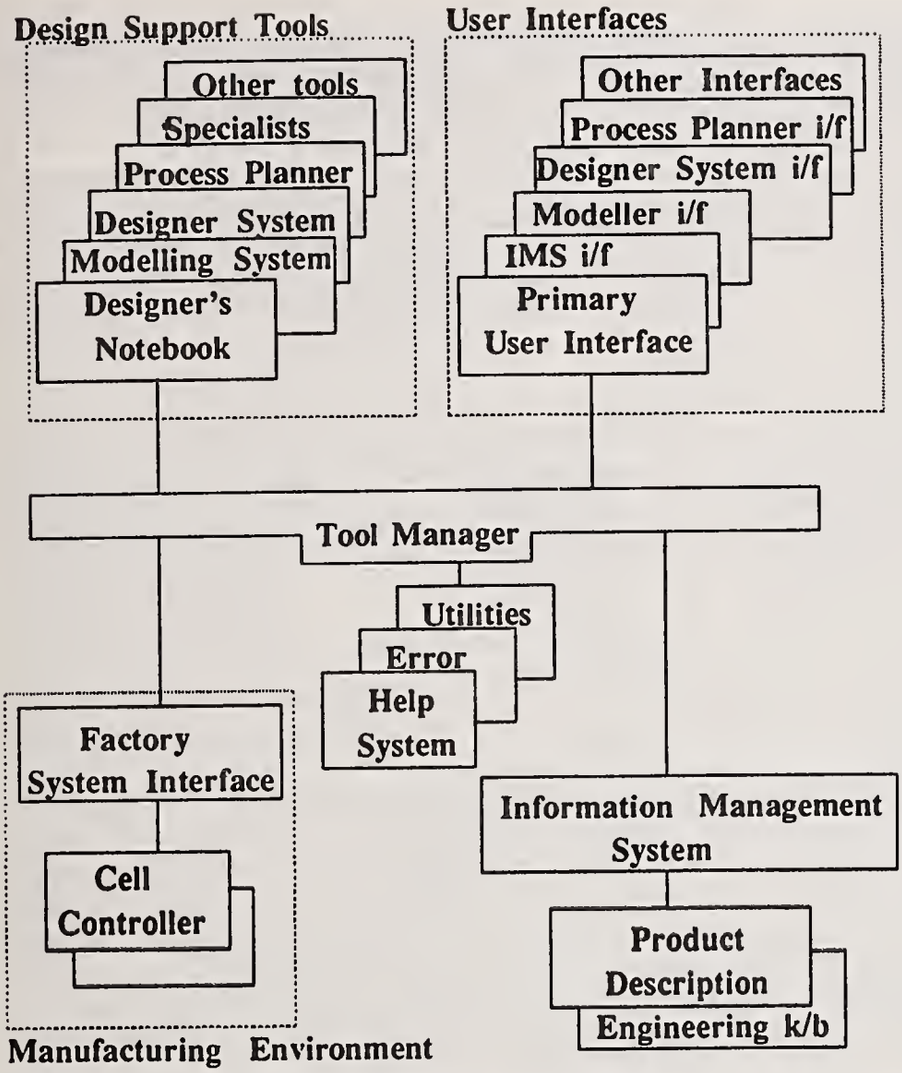


Figure 4. The final DtoP system architecture

No two tools in the DtoP system are alike. They are written in different languages (PROLOG, FORTRAN, C...), run on different machines (Sun 3, VAX, PC...) and contain a huge variety of things from databases to machine tools. They include interfaces to third party systems as proof that the DtoP architecture is open. The two things that all the tools have in common is that they communicate via the tool manager and use the contents of the information management system's knowledge bases. Indeed, these are the only two parts of the system that you really need. The tools to be found in a DtoP system would depend upon the application and no two systems will be the same. This also means that a system can start small and grow by the addition of new tools as the need arises or funds allow. The architecture is truly modular and extensible.

At this point the reader may be thinking that this is all too good to be true and wondering whether we have really done all we claim. Remember that DtoP is only a demonstration or prototype system, so it is not robust, is full of holes and patches and is not something we would want anyone to use for real and come to depend on yet, that requires more work. As, admittedly not very convincing, evidence that it does all exist, figure 5 shows a screen dump of the system's primary user interface that gives first access to all the tools and represents each one by a button.

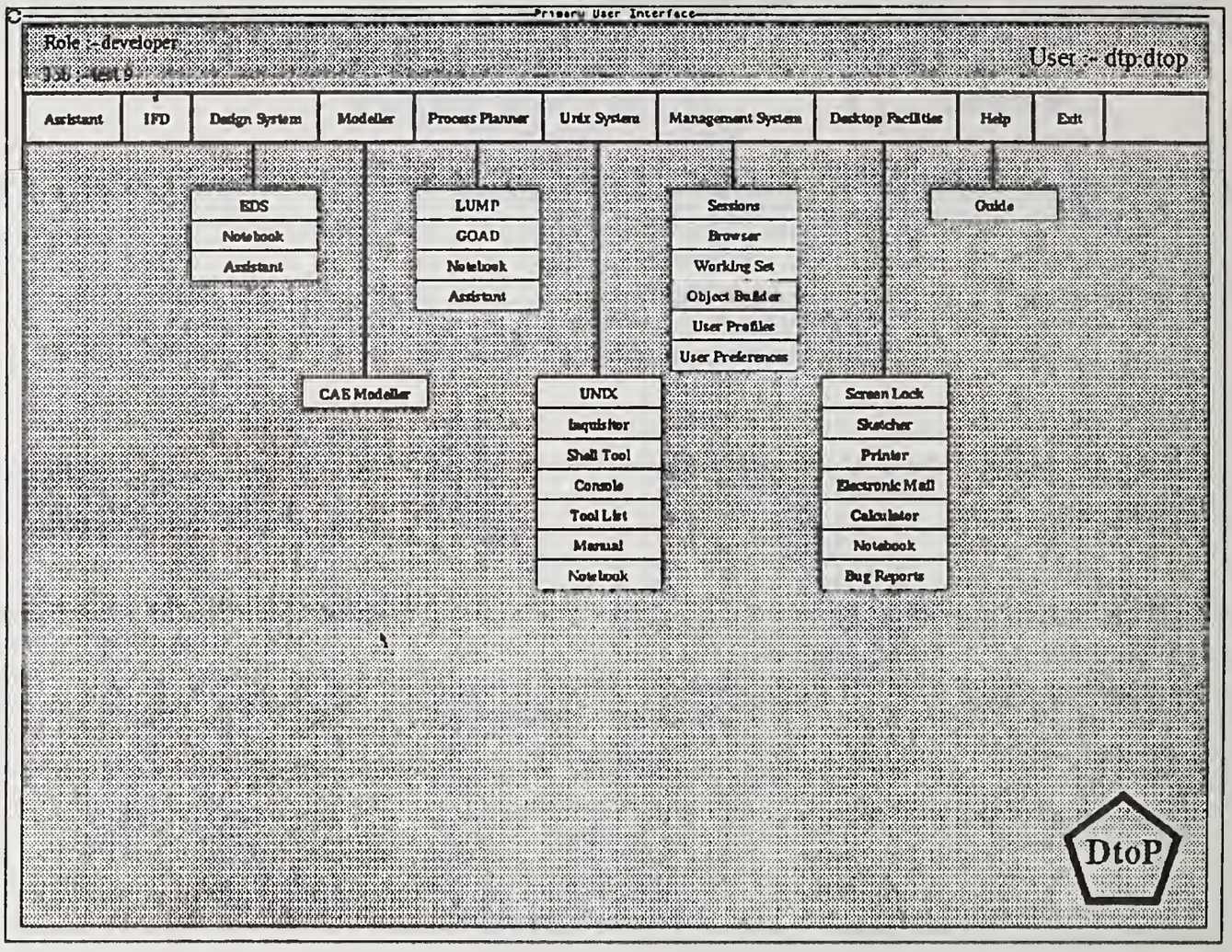


Figure 5. The DtoP primary user interface

2.8 OK, so how does it work?

There is space here for only a brief description of how the DtoP system works. For further details the reader should refer to [FEH90].

When the DtoP system is turned on all the tools register their presence with the tool manager and list the functions that they can provide in a standard tool call format recognised by the tool manager. The knowledge bases supervised by the information management system (IMS) will previously have been loaded with two kinds of information: static knowledge about product lifecycle support, such as codes of practice, standards, tables and rules, and dynamic knowledge about particular products that the system is used to support. The former is in an engineering knowledge base and the latter in a series of product description knowledge bases, each one for a different product. When a new product is started a new product description knowledge base is created. It is initially empty but over the entire life of the product accumulates information of relevance and provides a complete record of product design, manufacture and service history, assuming that the DtoP system continues to be used for those functions. In DtoP's present domain of fuel injection equipment most design is variant, drawing on old designs for inspiration, so a user is likely to want to work on an existing product description.

The user is first presented with the primary user interface, see figure 5, and from it selects the browser tool to locate the desired product description in the IMS. The user can then move between combinations of the design support tools to modify or add to the product design and perform manufacturing planning and other tasks. The user's activities are limited only by the capability of the tools and the information in the IMS. All the user's decisions are recorded in the product description, with reasons if he or she chooses. When design and manufacturing planning are complete the appropriate programs and resource information can be loaded into the manufacturing environment for production to begin. Feedback from the manufacturing environment, which is unique among the DtoP tools in that it may continue to operate with the same product information repeatedly for years, to the product description of modified programs or manufacturing statistics is possible.

When a user has finished what they wanted to do they return the product description they have been using to the IMS for later retrieval and leave the system. The DtoP system is not for completely novice product designers or other users. It contains a great deal of knowledge, and is accumulating more all the time it is being used, but is not an expert system. It automates some mundane tasks and supports users in more complex ones, but most of the time the user is in control and has to do a fair bit of thinking.

2.9 External factors that affected development

There were many external factors that affected development of DtoP's CIM architecture. Some we have mentioned already.

The biggest single influence was the marked trend across all industries away from large, monolithic CIM systems to smaller, modular ones. This was partly for technical reasons, because the large systems proved harder to build than anticipated, but the overwhelming pressures were economic. The large systems were expensive and were an all or nothing investment, there was no easy way to build them up in stages. This put the systems out of reach of small and medium sized companies and meant that large companies that could afford them either had to throw out the existing contents of their factory first or set up on a new site. Clearly this made the potential market rather small. Companies large and small want to acquire systems in stages, phasing investment to match needs and resources. They also want to be able to continue to use their existing systems, which means that they require open systems that are modular and can be built incrementally.

The rise of open systems has not been confined to CIM but has been a theme throughout the computing industry in recent years. DtoP has interfaced to systems from Computervision and Applicon as proof that it is an open system. This development has gone hand in hand with a change in the engineering computer hardware marketplace. Superminicomputers, such as VAXes, were all the rage when the project began and we planned to use them. However, we could see that single user workstations were the coming favourite and decided to switch to Sun 2 workstations during the pilot phase. When Sun 3s became available some collaborators bought those and finally the whole system was ported to Sun 3s, two or three of which are used in a network under NeWS. The emergence of windowing environments like NeWS and X-windows has made a big difference to the way DtoP is implemented and run on networks of workstations.

2.10 Internal factors that affected development

The single biggest internal project factor that affected development of the DtoP architecture, and more particularly the system, was the budget. We started off with very big ideas and were continually scaling them down during the project as the cost implications, and manpower skills requirements in some areas, became clear. Our ideas were still at the end bigger than we could implement and we came to view the DtoP demonstration system as only a partial implementation of the DtoP architecture in its full form.

More money and therefore more people would have been nice, but would we have been able to control it all? As it was, changes of personnel, for many reasons, over the life of the project caused some problems. The academic staff were mostly on contracts of three years or less that did not encourage project loyalty and turnover at some collaborators was high. In some cases the loss of a particular individual drastically affected work on some part of the system and altered the course of development.

Management of the project and strategic direction was in the hands of the industrial collaborators who had the job of exploiting the results afterwards. The companies involved had concerns greater than DtoP and the project was affected by some changes in company direction. Thus did the product that the project was focusing on change half way through and the product service support functions reduce in significance.

The rise in importance of human factors in the project following the appointment of our ninth collaborator had a large and beneficial impact on the system architecture. Although it took some time for the human factors message to be accepted and to make its mark, the influence was significant. To be successful, systems must do what their users need. The system developers must therefore find out what the users need and design to satisfy them, in terms of function and mode of operation. A system that does not fill a need or is cumbersome to use will not be used.

2.11 The biggest problems

The project met and solved many problems. A large proportion of those were not connected with what we were trying to do, i.e. design and build a CIM system, but stemmed from the way we were doing it, i.e. collaboratively between nine parties. These problems we have excluded from consideration in this section because they were peculiar to our project.

The biggest problems were caused by not having a proper system architecture at the beginning of the project on which to base all the work. Our first concept from figure 1 was never seriously pursued and had been put forward in the project proposal as a suggestion for a possible architecture, nothing more. This was not seen as a problem to begin with, indeed the absence of an architecture was welcomed as an opportunity to do our own thing, which was itself one of the problems. During the first half of the project a great deal of work was done with little or no thought for how it would later be made to work together. By the time that a workable architecture emerged, after two years, it was too late. We were too far down the road and the damage had been done. We could not afford to start again but knew that integration would be more difficult than it need have been. Ironically the fact that we succeeded in integrating the different parts of the system testifies to our claim that DtoP is an open architecture. Nevertheless, it was a problem that we could have done without.

A second major problem was coping with all the information that we had to absorb. External information about CIM, design and manufacturing in general, computing and a host of other subjects flooded into the project and we had difficulty keeping on top of it. Added to this were internal reports and specifications, minutes and memoranda. The temptation to give in, don blinkers and get on with the job was great, but had we done so we would not have produced such a flexible system so suited to the needs of potential users.

Two final major problems were getting the system to work and documenting it, but there is nothing new in either of those. It is nearly always more difficult than expected to get the different parts of a large system developed by a dispersed team to work together and engineers are never keen to document their work in quite the way that is most useful, if at all.

With those words of wisdom, we now leave Design to Product and move on to the other major CIM project we have been involved in : ESPRIT project 384, which we have led.

3. ESPRIT Project 384 - *Integrated Information Processing for the Design, Planning and Control of Assembly*

3.1 Introduction

The European Strategic Programme for Research in Information Technology (ESPRIT) was the first of many research programmes instituted by the European Commission with the aim of encouraging cross-border collaborative research in Europe. Over a period of five years, hundreds of companies, universities and research institutes, attracted by the fifty percent subsidy, have participated in pre-competitive research into five areas of Information Technology, of which CIM is one.

By the very nature of the conditions for obtaining funding, ESPRIT has succeeded in bringing together industrial and academic engineers throughout the European Community, though not always with the *synergy* and *harmonization* so loved by Commission scribes. There have been notable successes in the development of standards, in semiconductor design and in other enabling technologies. Research in CIM, under the aegis of ESPRIT, has spanned every area from the design of sensors for robot grippers to the development of CIM architectures. Our own project, involving initially five and later six partners, had the stated aim of "demonstrating the feasibility of an integrated manufacturing system covering all aspects of electro-mechanical assembly, from design to control". We started work in 1985 and the project ends in June 1990.

3.2 Project history

Due to factors internal and external to the consortium (discussed in the next section) the work carried out during the lifetime of the project took a course which had not been anticipated at its inception [SAN87]. In this section, we describe the major packages of work which were actually undertaken. Briefly, these are:

- Survey of assembly technology and project definition
- Selection and training in the use of software tools
- Development of target CIM system concepts
- Development of First Integrated Prototype (FIP)
- Evaluation of FIP and design of next prototype
- Implementation of Last Integrated Prototype (LIP)
- Documentation of software using CASE tools.

3.2.1 The first year

The plan for the first year of the project was to carry out a survey of the state of the art in assembly related software and hardware technology and to define the workplan for the rest of the project.

These tasks were carried out but, in truth, the first year became as much a team building exercise as anything else. There were differences of company philosophy to understand; we had to discover people's motivations and their ways of working. There were also some minor language problems to overcome, as typified by one fellow's remark that his english had improved since he had been on an english *curse*. But such *glitches* were inconsequential when compared with the different meanings people attached to simple words such as *cell*. This led to our writing a *glossary document* of some thirty pages. The state-of-the-art survey was useful not because it revealed anything

previously unknown to us, but because it ensured that all partners had a common basis for discussion. An analysis of selected products manufactured by the industrial collaborators similarly provided only a focus for discussion.

Those discussions revolved around the problems inherent in assembly and what contribution we could make to the field.

During that first year it was also agreed that we would develop some pieces of software to demonstrate our ideas on integration. These ideas involved three models: one of the equipment in the factory, one of the assembly processes the equipment could perform, and one of the products that could be assembled in the factory. All the modules available to support design and production activities in the CIM system would use those models and also each other's outputs.

3.2.2 Software tools

All partners were interested in applying Artificial Intelligence (AI) techniques in the project. At the time, AI was flavour of the month and we all believed that the technology would allow us to implement solutions where none had existed before.

We searched for the best AI toolkit money could buy, on the basis that we would recoup the money through increased programmer productivity. In the end, we had a shortlist of two systems: IntelliCorp's Knowledge Engineering Environment (KEE) and the Carnegie Group's Knowledge Craft. In hindsight, it is obvious that we would eventually choose KEE, as two of the partners already had it on their sites. However, a period of about three months was spent evaluating and comparing the two systems, principally because each one had a determined and loyal supporter in the consortium.

The management structure in early ESPRIT projects is such that the partner who is *Prime Contractor* has no executive power. Its role is to act as a focal point between the other partners and the Commission. Decisions have to be made democratically by the consortium and there is no formal mechanism to resolve differences. As it was essential that all the partners use the same programming environment, the choice of toolkit required the consortium to make a decision between one or the other, which led to considerable anguish. Subsequently, however, every important decision made by the consortium was to involve a compromise between the partners. Whether this is a bad or a good thing, we leave as an exercise for the reader.

3.2.3 The first integrated prototype

After the toolkit selection debacle, we spent a great deal of time defining the contents and structure of our three models. We also attempted to define the extent to which the functional modules would be integrated amongst themselves and to clarify the role of the user. Could the modules exchange information at arbitrary times or only when they had completed their task? What would be the best compromise between supporting users and doing the job for them?

During this period, differences in methodology began to emerge. Some partners liked to plan their work very carefully, considering all the consequences of their decisions. Others preferred to start *hacking*, testing their ideas in software and making changes as they went along.

To focus discussions, a simulation was carried out where the roles of both the CIM system and its users were played by people. The results were recorded on paper and extensively altered to arrive at a definition of our target CIM system. This illustrated the flow of information involved in the design and manufacture of a simple assembly provided by one of the partners.

Having completed this exercise, we felt that the document thus produced contained enough information to serve as a design specification for a first software prototype of the whole system. We reached this view because the focus of the project was integration, rather than the development of particularly capable modules such as planners and schedulers. The result would be a broad, rather than deep, demonstration system.

Over the next few months, the collaborators implemented all the components of the system. Despite the trauma involved in integrating this software (which was accomplished in a matter of days), the most visible flaw in the demonstration system was the lack of a common approach to the user interface. Each module operated satisfactorily and actually used information produced by the others. Of course, the software was limited in that it only supported activities related to the test product and, behind the glossy interfaces, there were some ugly last minute patches introduced to get the whole thing to work.

3.2.4 Evaluation and re-design of prototype

Rapid prototyping had thus provided us with a limited but working software system. The temptation was to develop this further, expanding its capabilities without making any radical changes. However, the essence of successful prototyping is in resisting this temptation, using the prototype merely as an investigative tool which supports the design and implementation of a new, better, software system. Consequently, the following nine months were spent evaluating the first integrated prototype and designing a new system architecture.

Whereas each model and module in the FIP had been developed by a single partner, taking into account the known needs of the others, it was felt that the evaluation and re-design of the system should involve all partners equally. Therefore, five groups were created: one for each of the three models, one with overall responsibility for software quality issues and the architecture of the system, and one to define the man-machine interface and user requirements, based on our new understanding of what could be achieved.

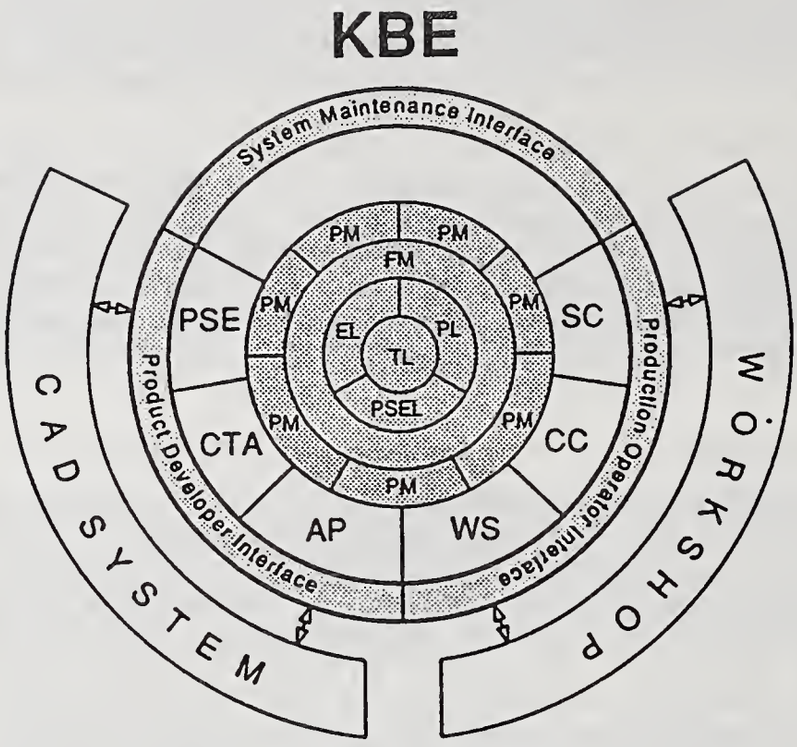
The creation of the groups was a management coup. With just one technical representative from each partner, an intense schedule of meetings and well defined goals, the groups became the focus of the project. Where before there were disagreements between collaborators, the tendency was now to have them between groups, a sign that the arguments were of a purely technical nature.

As further evidence of the success of the groups, a French team member and a German engineer were united in matrimony!

At the technical level, the FIP evaluation highlighted the fact that the KEE toolkit had been used primarily as an object oriented programming (OOP) system and it was therefore agreed that OOP principles would be followed more strictly in the next prototype. Software development guidelines were created and a suite of software written to check compliance with the guidelines. The functionality of the CIM system was defined and its architecture developed (see figure 6). The structure of the models was substantially improved and, finally, a user interface specification was written to ensure the consistency of the interfaces.

3.2.5 The last integrated prototype

At the time of writing, the LIP software is nearing completion. A first attempt at integrating the latest models and modules revealed a few minor deficiencies which are currently being addressed.



TL:	Technological Library	PSE:	Product Structure Editor
PL:	Process Library	CTA:	Connection Techniques Advisor
EL:	Equipment Library	AP:	Assembly Planner
PSEL:	Product Structure Element Library	WS:	Workshop Scheduler
FM:	Factory Model	CC:	Cell Controller
PM:	Product Model	SC:	Station Controller

Figure 6. The ESPRIT 384 system architecture

Functionally, this software places more emphasis on the system operators than the previous prototype. As a result of feedback from potential users, some of the more advanced features in the FIP were removed and more effort placed on providing facilities that allow the product developer to manually edit product and production information. In addition, the LIP deals with a range of products rather than just one and work has been done to facilitate access to external databases.

3.2.6 Use of CASE techniques

In parallel with the later stages of development of the LIP, the collaborators were asked to use a computer aided software engineering (CASE) tool to document the software produced. For practical reasons, the tool chosen had to be already available within the consortium, which resulted in our selecting KnowledgeWare's Information Engineering Workbench. In particular, the AWS (Analysis Workstation) module is currently being employed.

The first thing to say is that AWS is not well suited to describing OOP software, but then, no CASE tool we have seen is really satisfactory for that purpose either. However, it is clear that such a tool, had one been available, would have provided an excellent alternative to our paper simulation approach to defining the architecture of our system. Whether CASE could have avoided the need for the first prototype is much less obvious. In principle it could, but in practice we suspect that

there was a real need to get on with some practical implementation work in order to keep engineers motivated and senior managers supplied with demonstrations, and to prove the feasibility of the innovative ideas being proposed.

As we near completion of the work with AWS, the suspicion is that we will rediscover conceptual differences between the partners' views of how the target CIM system should operate. Although we have implemented a fully working system, rapid prototyping encouraged the global view to be forgotten while small interconnectivity problems were resolved. In addition, the modular OOP nature of the software did not necessitate the overall architecture to be finalised in order to demonstrate the modules and their connections with other modules and the models.

3.3 Achievements

The time has come to describe the software developed by the consortium. This will be, of necessity, only a brief overview.

The system is designed for use in small batch manufacturing environments. It is assumed that the assembly workshop contains a number of cells linked by a common transportation system. Each cell is composed of a number of automatic assembly stations, typically consisting of a robot. A common transportation system also links all the stations within a cell. By definition, at any instant of time, each station may only work on one product, though others may be in the station waiting to be worked on. In contrast, under normal circumstances, all the stations within a cell work concurrently.

The key element of the system is the Knowledge Based Environment (KBE), shown in figure 6, which is designed to support the development of products which may be manufactured in the workshop. The KBE consists of the following integrated models and modules. The Equipment and Process Models describe in considerable detail the capabilities of the workshop and contain taxonomies and tools that enable them to be modified for use in other factories [SAN89a]. When a product developer uses the KBE, (s)he refines and optimises the product design and its associated manufacturing information. This is held in a Product Model which comprises the product structure, components, their connections, references to the processes and equipment to be used, and an assembly graph which defines the precedence of assembly operations. Some of this data is held outside the KBE in a CAD system; the two are linked in such a way as to ensure data consistency and so that some of the functions of one system can be invoked from the other, making for a productive working environment.

In principle, any number of functionalities could be added to this basic framework. The following ones have already been prototyped. A design support system which allows parts of previous designs to be reutilised. A product structure editor which allows the relationships between components and subassemblies to be defined. An expert system which helps the user select the most suitable connection techniques to use in an assembly. A planner which helps in structuring and detailing of the process plans for a product. A workshop scheduler which combines orders into lots to be assembled in the various cells available in the factory. A cell scheduler which works together with the workshop scheduler to produce a detailed schedule for the stations in each cell [SAN89b]. Finally, a station control system, implemented partly in KEE and partly in a proprietary robot control language, which utilises process plans to transform parts into assembled products [DWO89]. The system has been tested using simulators and one physical station.

It must be stressed that the achievement is not the development of software modules but, rather, their integration. Indeed, had suitably *open* packages been available commercially, we would have been contented with integrating those. As it turned out, we made use only of a commercial CAD system and a low-level control system, the rest of the software being developed from scratch with integration in mind.

3.4 Problems encountered

In this section we discuss the problems which most hindered the smooth progress of the project.

3.4.1 Technical difficulties

Surprisingly, these have been relatively minor, compared to *people problems*. Sure, linking the KBE to the CAD system presented difficulties, as did the programming of the assembly station. In other areas too, naturally, there were problems. Given the goal of producing an integrated system, the tough issues have concerned information processing; the engineering problems inherent in assembly rarely made it to the surface. At times it felt as if we were solving only the problems we ourselves had introduced. Given the project's emphasis on building software prototypes, it is not surprising that many of the day to day difficulties we encountered were of the programming variety. KEE, and the Symbolics Lisp Machine programming environment we have used, are simply superb for rapid prototyping due to their richness and flexibility. The latter is invaluable when used by a small tight knit team of programmers but can be counterproductive otherwise.

3.4.2 Personnel matters

The following problems will be familiar to anyone managing projects of long duration.

Over the lifetime of the project, there have been numerous changes in personnel. Apart from the problem of training, which is particularly acute in the case of sophisticated software tools, engineers were replaced with little consequence. More unfortunately, eight changes of management have taken place at the commission and the various sites. This has affected the direction and style of the work. For those who have stayed the course, this has perhaps been helpful in reducing loss of enthusiasm for the work, but the project as a whole has suffered.

We cannot end this section without referring to the problems caused by having in the consortium many individuals possessing equally strong personalities. *Group dynamics* are interesting to observe, but a nightmare to deal with.

3.4.3 Company related issues

It is extraordinary to note that in a five year span, three of the original five partners have been taken over by other companies. Luckily, this has not had as great an impact on the project as one might have expected.

A greater problem has been the lack of involvement in the work by *front line* designers and engineers; those with the greatest need of a CIM system have the least time available to contribute to its development. Throughout the consortium, company support for the work has been patchy, both financially and in terms of human resources. The time and cost involved in cross-border travel has also been detrimental to progress. As an alternative, electronic mail was used for a short while but found lacking. Faxes, which allow diagrams to be transmitted, have been used extensively. If we were starting the project now, we would specify that all the partners had to use a common word processor and graphics package. With the current proliferation of e-mail in Europe and greater standardization amongst the partners, communication of text, program code and diagrams ought to be possible.

As a result of the overheads imposed by travel and meetings, we would guess that, given comparable resources, any one of the partners could have achieved much more than the consortium has as a whole. But, considering each partner's actual financial commitment to the project, collaboration has allowed each company to influence and share results that would just not have existed without it.

4. Conclusions

If we tried to sum up all the lessons were have learned on DtoP and ESPRIT 384 in a few pithy phrases we could not. Being involved in the projects has been an education in many ways and the experience will be with us for the rest of our lives. Some of the most important points that we would like to pass on as valuable lessons learnt are:

Managing a project

- Do not underestimate the technical difficulties and adjust either the objectives or the budget accordingly. Everything may seem possible at the start, and probably is, but the price may be higher than you think.

- Keep yourself informed of what is happening in the world outside, both to your potential users or customers and in CIM circles.

- Five years is a long time; try to keep projects shorter to avoid changes in strategic direction and reduce personnel turnover.

- Strong and respected project management can smooth the path greatly.

- CIM projects are great fun, but tiring too.

Developing a system

- Start with a clear idea of where you want to go, by adopting an architecture either before you write the proposal or as soon as the project starts.

- Do not think that technology can solve everything, you will need people to use and work with your system.

- Find out what your future users really need and design for it. This is more difficult than it sounds because users, even if you can identify them, may not know what they need.

- If you do not have human factors expertise on your team, find some and use it.

- Rapid prototyping and CASE are useful techniques for exploring problems and possible solutions.

- If possible, develop systems incrementally. Delivering partial solutions early on motivates sponsors and engineers alike.

- Use the best quality software development tools you can afford. Avoid developing your own as this diverts effort from the true objective of the project.

Demonstrating a system

- Explaining complex systems to people who have not been involved in developing them is difficult. Find someone who can do it and make them your front man, keep the other engineers out of the way.

- Demonstrating software convincingly using only a workstation screen, keyboard and mouse is impossible. The quantity of information people can absorb is miniscule.

Collaborating with others

- If you must collaborate to obtain all the necessary expertise, keep the number of partners as small as possible.

- Make sure *your* objectives are clear, or you will end up servicing the others' requirements without extracting anything useful from the work.

- Allow time for the process of collaboration.

We do not advance either of our two CIM architectures as panaceas for manufacturing industry's problems. They are appropriate to the particular types of problem that the two projects addressed and we believe them to be adaptable to a wide range of CIM applications. However, no two manufacturing enterprises are quite the same and while the adoption of standards has distinct advantages in some areas, such as communications and information management, the appropriateness of a solution, in terms of complexity, cost and other factors, to any given problem must always be an important factor in system design. Only one thing is certain: be guided by our architectures and by our conclusions if you will, but do not follow our development roads: they were long, tortuous and full of potholes.

5. Acknowledgements

We wish to express our gratitude to all the partners in both projects and, in the case of the ESPRIT project, to the reviewers for their comments and guidance. We of course also wish to thank the European Commission, UK Department of Trade and Industry, Ministry of Defence and Science and Engineering Research Council for their financial support.

Our collaborators in the Design to Product project were: GEC Electrical Projects Ltd., GEC Avionics Ltd., Lucas Diesel Systems Ltd., University of Edinburgh, University of Leeds, University of Loughborough, HUSAT Research Centre and National Engineering Laboratory.

The ESPRIT 384 consortium includes Daimler Benz (Frankfurt), IPK (Berlin), Investronica (Madrid), TNO (Apeldoorn) and La Telemecanique Electrique (Paris).

6. References

[DWO89] B Dwolatzky and S P Sanoff, "A Product-centred Controller for Flexible Assembly Cells", Proc. of the 10th Int. Conference on Assembly Automation, 23-25 October, Kanazawa, Japan. 1989.

[FEH90] P A Fehrenbach ed., "The Alvey 'Design to Product' Demonstrator Project - An Integrated, Knowledge Based Approach to Product Lifecycle Support", project document DTOP/EXT/GECR/1/1, available from the author, 1990.

[POU89] K Poulter et al., "The KERIS Reference Manual", available from GEC-Marconi Research Centre AI Division, 1989.

[SAN87] S P Sanoff et al., "Integrated Information Processing for Design, Planning and Control of Assembly", ESPRIT Conference '85, 23-25 September, Brussels, Belgium, 1987.

[SAN89a] S P Sanoff and B Dwolatzky, "Intelligent Models for Assembly Design, Planning and Control", ESPRIT Conference '87, 28-30 September, Brussels, Belgium, 1989.

[SAN89b] S P Sanoff and D Poilevey, "Integrated Scheduling and Control of Manufacturing Cells", Proc. of the 5th CIM-Europe Conference, 17-19 May, Athens, Greece, 1989.

THE DEVELOPMENT OF A CIM ARCHITECTURE FOR THE RAMP PROGRAM

ERIC E LITT

MAY, 1990

Abstract

This paper describes the development of the RAMP Architecture and presents a top level view of it from functional, information, and control perspectives. It also presents some of the considerations in the development of a CIM Architecture such as the role of standards, commercial versus custom code trade offs, and a brief discussion of lessons learned from this development effort.

1. Introduction

The Rapid Acquisition of Manufactured Parts (RAMP) program was initiated by the Naval Supply Systems Command (NAVSUP) to address the time that it takes to acquire replacement spare parts. NAVSUP funded a study to determine the average time that it takes to get a replacement part from the time the demand is recognized to the time that it is delivered. The results of that study showed that it takes between 300 - 600 days to fill most orders.

Furthermore, it concluded that there are three distinct time periods in the acquisition process that account for the total time. The first period is called Procurement Administrative Lead Time (PALT). This covers the period from when a demand is recognized until an order is placed with a vendor. The second period is called Manufacturing Administrative Lead Time (MALT). This covers the period from award of contract until the time that work is released to the shop floor. The third period is called Manufacturing Lead Time. This covers the period from the time an order is released to the shop floor until it is shipped to the customer.

The initial phase of the RAMP program developed the requirements specifications for addressing all three of these time periods. This paper describes the development of the RAMP Architecture and presents a top level view of it from functional, information, and control perspectives.

2. What is an Architecture?

There are many parts to defining an architecture of a CIM system. We have broken them down into the following categories:

1) functional relationships,
2) information relationships,
3) processing sequence relationships, and
4) control relationships.

As one can see, the common denominator is that all of the categories define a set of relationships. In fact, they all describe the system with the only difference being the perspective from which they view it. The Architecture provides the framework under which the detailed system design is developed. A well thought out architecture provides for modularity, flexibility, and asynchronous design. It should provide the designers with a good roadway to follow, and enough boundaries on them that they can all work fairly independently without a high risk of design conflict.

A good analogy is that of a building. Once the foundation is in and the shell is erected, the basic framework of the space is defined. The interfaces between the floors are defined, the utility services are defined, the boundary to the outside world is defined, and the basic functionality of the floor is defined. The variability is in how each floor is configured to meet its functionality. In this regard the designer has a considerable amount of flexibility in how he does his job, yet he does have constraints as to how he fits in with the rest of the building. Over the life cycle of the building the floor may be remodeled several times, in fact even its functionality may be changed yet it is not necessary to tear down the floors above it to accomplish these modifications.

In the design of a CIM system architecture one must have the foresight to consider the types of changes that may occur to the system over its life cycle and provide for mechanisms that will enable the modification of the system without a complete redesign. Since no one can predict the future changes to the system, it is crucial to have a modular design with well defined interfaces such that the system can be upgraded in a modular fashion.

3. The Role of Standards in Developing a System

Standards can play an important role in insuring that the modularity of the system is achieved. There are published standards for many of the components that make up a RAMP system. Examples include the common database which is an ANSI compliant SQL DBMS, and the backbone communications network which is OSI compliant.

The value of adhering to these standards is particularly evident in an environment that one is using a conglomeration of commercial products. By using products that adhere to the standards there is a greater chance of success in integrating the products because the vendors have some common ground. The SQL DBMS is a good example of this in that when sharing information between applications from different vendors, it is easier to do so if each talks a common language. In this case the common language is SQL.

Standards are only effective however if they are endorsed by industry. MAP is an example of a standard that has not had the wide spread endorsement of industry. Though the goal of MAP will benefit the customers, many vendors have been reluctant to provide products that interface with it. As a result, it may never reach its potential until there is a stronger commitment from industry to support it. This is unlikely to occur without the customers demanding it en masse.

4. Approach to Defining Architecture

4.1 Design Team

The South Carolina Research Authority (SCRA) is the prime contractor on the RAMP/RTIF program for the Navy. However, all of the technical work is performed by one of four subcontractors. The four subcontractors and SCRA make up an organization called the American Manufacturing Research Consortium (AMRC). The membership of the Consortium includes ARTHUR D LITTLE, BATTELLE, GRUMMAN DATA SYSTEMS, SEACOR, AND SCRA.

4.2 Design Approach

This program used a five phase top down design approach. It should be noted that these "phases" do not correspond with any contractual nor implementation requirements or documentation. However, they do document the process that was used in developing the systems.

Planning (Phase One)

During this phase a comprehensive set of plans were generated in order to meet the Navy's objectives of designing a system that would result in the desired lead time reductions. Examples of the types of plans developed include:

Program Master Plan,
System Engineering Master Plan, and
System Integration Plan.

Establishment of System Specifications (Phase Two)

During this phase the functional and design specifications were developed using the system requirements identified in the planning stage. Various methodologies were used to develop the specifications. These methodologies included IDEF0 Function Modeling, Yourdon-Demarco Data Modeling, Entity Relationship Modeling, and Process Flow Diagrams. Most of these modeling techniques were supported by Computer Aided Software Engineering (CASE) tools. These tools were used to ensure consistency and to perform error checking within the models. Examples of the types of documents developed during this phase include:

IDEF0 Function Model of a complete enterprise,
Generic Information Model,
Control Architecture,
Generic Interface Requirements Specification, and
Type "B" Specifications.

Design (Phase Three)

During this phase the detailed design work was accomplished. This included writing detailed component specifications and test plans as well as the design of custom components and interfaces. Type "C" specifications and procurement packages were prepared and released for bid. Examples of the types of documents developed during this phase

include:

System Acceptance Test Plans,
Component Test Plans,
Failure and Corrective Action Program Plans, and
Type "C" Specifications.

Construction, Integration and Testing (Phase Four)

This phase was done incrementally resulting in the build up of the complete system. The first activities were to procure the hardware and software necessary to write the custom software code. While the code was being written and debugged, the long lead time items such as the shop floor equipment was being procured. The System's Top Level Components (TLC's) were implemented in parallel by component, starting at the lowest levels of the hierarchy. For the manufacturing cell, the equipment was installed and tested first on an independent basis, and then as a part of a workstation, until all of the workstations were installed. The Manufacturing Cell and the other TLC's were then tested on a stand alone basis to ensure their functionalities, and finally the complete system was tested.

Installation (Phase Five)

Efforts are now under way to implement the system at three different sites. These sites are Cherry Point, NC; Naval Avionics Center, IN; and Charleston Naval Shipyard, SC. During this phase the system will be shipped to these sites and installed in their facilities. The external interfaces to their existing systems will be tested and verified, and a complete site acceptance test will be performed.

As was mentioned under Phase Two (Establishment of System Specifications), the information and control architectures were defined using the requirements identified in phase one as a baseline. The approach was to consider the complete enterprise, and model its functions and their interrelationships. This was done using the IDEF0 methodology. At that time there were no CASE tools available to assist in creating the models, so it was done by hand and input into a CAD system. Since then however, several automated tools have become available on the open market, which we have used on other projects. Once these relationships were clearly understood, an information model was developed using the Yourdon-Demarco methodology implemented in a CASE environment. This model detailed the information requirements for each of the processes identified in the function model.

Once the concept for RAMP was developed in the context of an enterprise, it was decided to scale back the initial implementation to focus on those areas that would have the largest return on investment for the sites. The enterprise functions were grouped into two basic categories; those that directly supported manufacturing, and those that were support functions. It was decided that most of the support functions such as maintenance, payroll, accounting, etc. would continue to be performed by the site using their existing systems. The remainder of the functions comprised the baseline design for the first RAMP implementations.

This baseline design was documented in an information and function model using the same methodologies. It also identified the data storage requirements and the relationships between the various data stores. This served as the basis for the Input Process Output charts and the database design in the Software Requirements Specification (SRS) and the Software Top Level Design Document (STLDD).

5. The Ramp Architecture

5.1 Information Model

Figure 1 shows the top level of the RAMP Generic Model. One will note that there are four processes identified at this level. They are:

1) Production and Inventory Control,
2) Manufacturing,
3) Manufacturing Engineering, and
4) Quality.

These correspond directly to four of the seven TLCs of the RAMP System. In addition to these functions, all segments of the RAMP System are integrated through a common architecture consisting of three other TLCs:

5) Information Management,
6) Communications, and
7) Control.

TLCs 5 through 7 comprise an information processing "shell" into which Production & Inventory Control (P&IC), Manufacturing, Manufacturing Engineering, and Quality are integrated.

The following text describes the functional requirements to be carried out by each of these TLCs, and from a top-level viewpoint, the lead time reduction benefits that are achieved by each TLC. The partitioning of these TLCs into Lower Level Components (LLCs) is also described.

5.1.1 Production and Inventory Control TLC

To enable the reduction of lead time, the P&IC TLC provides administrative tools for achieving semi-automated, closed-loop workload control. There are four LLCs which comprise P&IC. The LLCs which incorporate this functionality include Capacity Requirements Planning (CRP), Production Control, Order Entry, and Material Inventory Management (MIM).

Capacity Requirements Planning

> CRP helps ensure that Required Delivery Dates are met by reserving and allocating a period of time for orders to be processed by RAMP resources. CRP also ensures that the workload reserved and allocated for the resources does not exceed the finite capacity available, thus preventing the build-up of work-in-process inventories.

Figure 1 Information Model

Production Control

Production Control provides the mechanism to release an order to the shop. This occurs when it determines that all items and process plans are available and shop capacity permits. In addition, it assigns priorities to the orders, releases the orders, and updates the Order when all RAMP processing is complete.

Order Entry

Order Entry provides the capability to determine order status conditions that require site attention, provides two-way communications to monitor, send and receive order-related information to/from the site support functions, and provides the capability to initiate orders.

Material Inventory Management

MIM acts as the point of contact between the site and RAMP to requisition, track and receive direct and indirect items, and to obtain outside processing services. As a result of the requisition, this process also receives projected delivery dates of items from the site for use by RAMP.

5.1.2 Manufacturing TLC

The Manufacturing TLC enables the achievement of lead time reduction by providing an integrated methodology to reduce the time parts spend waiting to be processed on the shop floor. Industry statistics have shown that in a conventional shop, parts spend an average of 5% of their overall time in the shop being processed, and the remaining 95% of the time in queue to be processed, in setup, or waiting for a problem to be solved. The Manufacturing LLCs, include Schedule Manufacturing Cell, Manage Maintenance, Coordinate/Monitor Manufacturing Cell, Manage Indirect Inventory, Workstation Control, and Transportation Control. They provide mechanisms to reduce this waiting time.

Schedule Manufacturing Cell

Schedule Manufacturing Cell provides the mechanism for preparation, interactive update, and limited analysis of resource work agendas, as well as initiation of control tables used for "pulling" Shop Work Orders (SWOs) through the shop floor.

Manage Maintenance

Manage Maintenance interfaces with Site Maintenance to request timely support and receive commitments and schedules of the planned execution of the requests. This information is used for scheduling and coordination of machine outage repairs and preventive maintenance.

Coordinate/Monitor Manufacturing Cell

Coordinate/Monitor Manufacturing Cell has the control logic to manage the activities of all workstation level controllers in order to achieve a methodology of shop floor control to pull SWOs through the shop. It downloads commands and detailed instructions, and processes uploaded status/state updates.

Manage Indirect Inventory

Manage Indirect Inventory provides the information processing support capability for filling requests, inventory maintenance, and directing returns/receipts of all indirect items.

Workstation Control

Workstation Control provides all of the information processing required to support the timely execution of production tasks at the workstation level.

Transportation Control

Transportation Control provides all of the information processing required to support the transportation of items between workstations.

5.1.3 Manufacturing Engineering TLC

Manufacturing Engineering achieves lead time reduction through the semi-automated generation of macro and micro process plans. The RAMP uses RAMP PDES to generate macro process plans when feasible, which will enhance the quality of end items. The LLCs of Manufacturing Engineering include Create Process Plans, Evaluate Problem Cause, and Generate RAMP PDES.

Create Process Plans

Create Process Plans includes verification, extraction, and conversion of the RAMP PDES, creation of the Macro and Micro Process Plans, creation of the final test or inspection plan, maintenance of databases used in process planning, and coordination of process planning.

Evaluate Problem Cause

The Evaluate Problem Cause LLC provides interactive resolution of discrepant part problems and Engineering Service Requests and Completion notices.

Generate RAMP PDES

Manufacturing Engineering is capable of limited generation of RAMP Product Data Exchange Specification (RAMP PDES) based on a Level III Part Technical Data Package (L3PTDP).

5.1.4 Quality TLC

The Quality TLC performs the generation of Quality Reports, the coordination of the dispositioning process surrounding discrepant/quarantined parts, the arrangement for quality services not found within the RAMP, the assembly of Part Pedigree Reports, the generation of Part Quality Records, and the monitoring of Resource Certification. By supporting first-run quality from both a part inspection and process monitoring aspect, the Quality TLC will support lead time reduction by reducing the percentage of scrap and rework incurred on the shop floor.

Generate Quality Reports

The Generate Quality Reports process accesses and retrieves Part Quality Data and Inspection Results generated during the actual manufacturing execution of the process plan operations. It computes process control measurements of the total manufacturing process for inclusion into Quality Reports.

Coordinate Disposition of Quarantined Part

This process coordinates the disposition process associated with parts determined to be in a discrepant condition, and placed in a quarantined status. It also arranges for quality services not found within the RAMP.

Assemble Part Pedigree

If required by the order requirements, the Assemble Part Pedigree process assembles a complete component/material pedigree for a given part.

Generate Part Quality Record

Generate Part Quality Record compares actual Quality Reports to order quality report requirements included in the process plan. The objective is to assure that all order requirements have been met and that the required documentation is created.

Resource Certification

The Resource Certification LLC ensures that all manufacturing equipment and personnel maintain their prescribed calibration and/or certification, and only certified equipment and personnel are utilized in the manufacturing processes.

5.2 Information Management TLC

The Information Management TLC manages all data shared between more than one RAMP TLC using the Common Database (CDB) for the storage of all shared data, while assuring data security and integrity. Information is transferred either as a message or a file.

The Common Database Manager (CDBM) interfaces with the Command Status Services (CSS) to transmit all data. CSS supports bi-directional communications providing CDBM with commands and output files to initiate or respond to other RAMP components. At each application node, CSS transmits data into the node's mail box and receives status data from the node's Application Control Interface (ACI).

The ACI provides for the integration and control of the various COTS applications programs in the RAMP using messages from the CSS. Messages interconnect the ACI Application Dispatcher, Execution Monitor, Upload Monitor, and Download Monitor LLCs within the CSS. When a Controller at any level of the hierarchy applies its control logic to assign a task to a subordinate, these LLCs perform the data downloads and uploads.

5.3 Communications TLC

The Communications TLC provides for Application File and Message Exchange, Data Handling, Interfacing to other networks, and Data Communications Management.

The Application File and Messaging Service provides file and message exchange services to all application functions using mailbox communication services. The user application requests file/message transfer service and provides valid source and destination names. The local/network operating system validates and processes the application service request. Access is made to the requesting application for valid service requests, otherwise an error message is generated.

Data handling functions determine whether the message/file request is local or remote, transparent to any user applications. Local requests are processed by the local operating system node and remote requests are processed by the network operating system node. Application-to-application valid data transfer, retransmissions, or error/fault status messages are generated as required.

Provisions to interconnect to other networks is provided through the network-to-network interface for file/message exchange support services. The network-to-network interface facilities will be standard bridges, routers, or gateways. This function is currently under design for the Cherry Point Implementation.

Communication Management performs the following functions:

1) Maintain/update Network Configuration and User/Network Address Database,
2) Control network security access and operation,
3) Provide alarms, and
4) Provide data communications statistics.

When a Controller at any level of the hierarchy applies it's control logic, to assign a task to a subordinate, the Communications TLC provides the facility for data transfer.

6. Control Architecture

6.1 Control TLC

The purpose of the Control TLC is to provide a mechanism for communicating timely and accurate control information and task processing information to and from RAMP resources (artisans, equipment, and information processing devices) that perform the P&IC, Manufacturing, Manufacturing Engineering, and Quality functions. The Control TLC instructs the RAMP resources when and where to perform a value-added task on an order or SWO. The RAMP resources then follow the instructions and report the status/state resulting from the performance of the value-added task.

As illustrated in Figure 2, the Control TLC is organized in a five level hierarchy with the highest level being the enterprise followed by the system, cell, workstation, and equipment levels. The current RAMP implementations implement the Manufacturing System, Cell, Workstation, and Equipment levels of this hierarchy.

This control schema is based on the premise that all processes or resources receive commands from their supervisory controllers, and dispatch commands to their subordinates. In addition, they receive status/state information from their subordinates and send status/state information to their supervisory controller.

This closed-loop control will enable the achievement of the lead time reduction goals, by avoiding situations where RAMP resources lack information, perform the wrong task, or do not coordinate with one another properly, and by providing a proper startup/shutdown/restart mechanism. The design of the RAMP system minimizes the anomalies (errors, failures, irregularities) that occur during the processing of orders, but when an anomaly occurs, proper control ensures the proper recovery from the anomaly.

6.2 Process Flow

Figure 3 shows part of the process control flow diagram. It shows that the sequence in which the processes are activated is based upon state changes. This control logic is embedded into state tables which, when utilized, execute predefined actions. The logic that was input into the state tables was derived from the process control flow diagrams. As one can see, the sequence of events is predetermined based upon the results of the completion state of the previous action.

We have found that the best tool in presenting the system design is Figure 4. Figure 4 shows the physical representation of the components that make up the system in their respective hierarchical positions. However, this figure by itself is not enough to have a full understanding of the architecture. It is necessary to have a thorough understanding of all of the logical diagrams to understand the significance of the physical diagram.

Figure 2 RAMP Hierarchy

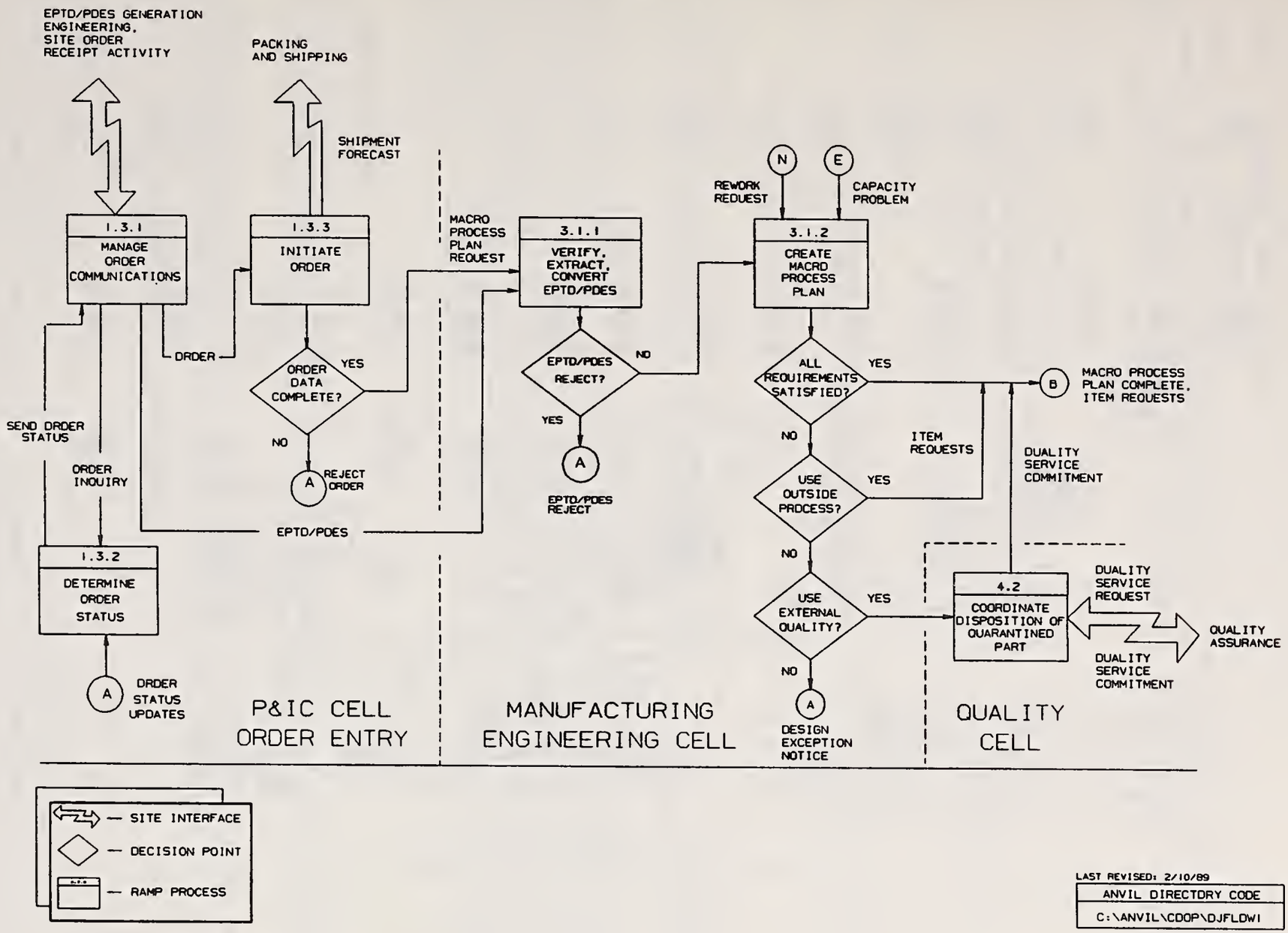


Figure 3 Process Control Flow Diagram

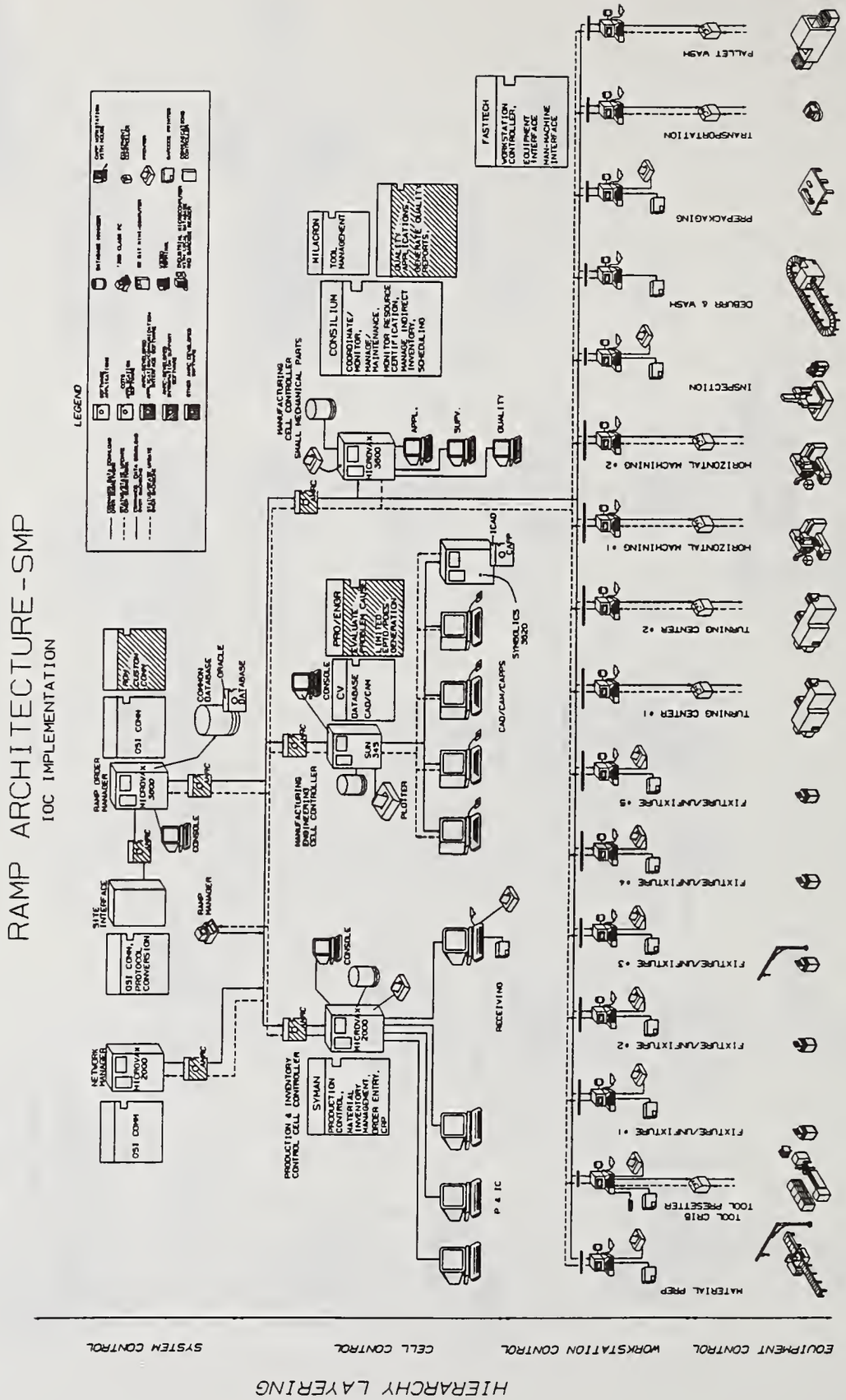


Figure 4 Implementation Diagram

7. COTS vs. Customer Code Trade Off

There is a significant decision to be made when implementing a CIM system as to whether the code should be custom written for the application, or whether commercial products should be used. The advantages and disadvantages of using custom code are outlined below.

Advantages:

1) functionality is performed as specified,
2) no overlap between software modules,
3) one to one correlation between design and implementation, and
4) developer has complete control of system configuration.

Disadvantages:

1) high risk associated with a developed software package, and no installed base,
2) cost to develop, and
3) support required to maintain it over its life cycle.

The development concept for RAMP was based upon using Commercial Off the Shelf (COTS) software, computer hardware, and equipment as much as possible. All of the RAMP computer hardware, equipment and application software is COTS.

When evaluating COTS for it's applicability to a particular situation, it is important to have a well defined set of criteria for evaluation. This criteria should be used to asses the functionality, as well as the performance of the COTS in question. There should also be a rating scheme by which relative weights can be assigned to the importance of the function or its performance. Also, when making decisions on software that will be integrated with other applications, it is necessary to consider the degree of difficulty for integration.

8. Lessons Learned

A top down approach to system design is an extremely beneficial exercise to go through, even if the whole design is not implemented.

When evaluating a COTS product it is important to make SITE VISITS and observe the use of the product in an actual production environment.

All evaluations should be against a standard.

Selection should be based upon an explicit understanding of the requirements.

IMPLEMENTATION OF THE RAMP ARCHITECTURE AT AN ESTABLISHED SITE

DAVID W. JUNG
MAY, 1990

Abstract

This paper provides insight into the lessons learned in adapting the generic Rapid Acquisition of Manufactured Parts (RAMP) architecture, to create a conceptual architecture for the Cherry Point Naval Aviation Depot (NADEP). AS-IS IDEF0 models were successfully used to document the existing site operations. Baseline RAMP data flow, process control flow, and physical models were then adapted to allocate functions and interfaces to the TO-BE Cherry Point RAMP Manufacturing System (CPRMS). As a result of the adaptation, the time and manpower required to create the initial conceptual architecture was reduced. However, it was also learned that conceptual architecture development is an on-going, rather than static, activity.

1. Introduction

The RAMP Manufacturing System (RMS) is currently moving toward the production test phase at the RAMP Test and Integration Facility (RTIF), Charleston, South Carolina, and will meet its primary mission of producing Small Mechanical Parts (SMPs) for the Navy. A second goal of the RAMP program is to transfer the developed architecture and technology to established sites. Initially, RAMP technology will be transferred to Navy and Marine Corps sites which have the mission to support fleet readiness. These sites are therefore key candidates for realizing the benefits of replacement part leadtime reductions, as well as other RAMP productivity improvements, such as improved quality, inventory reductions, and an increase in machine utilization.

The RTIF construction, integration, and testing phase was underway, but not completed, when the effort to transfer the generic RAMP architecture started in early 1989. This paper concentrates on describing the tasks that were performed during the period from January, 1989 through June, 1989, to create an initial conceptual architecture for the CPRMS.

A fundamental goal from the start of the CPRMS architecture development was to leverage and adapt the RAMP architecture to Cherry Point NADEP needs. All of the program management plans and models that were produced in developing the RAMP architecture were available to the American Manufacturing Research Consortium (AMRC) systems engineers for development of the CPRMS architecture.

Although a significant portion of the RAMP architecture was directly transferred to the CPRMS architecture, there were also site specific factors which required additional architecture development. The major drivers of the CPRMS adaptation of the generic RAMP architecture included implementation strategy, enterprise integration, and shop floor integration.

The first task performed in developing the CPRMS conceptual architecture was creation of AS-IS IDEF0 models to communicate the site survey findings. Completion of the AS-IS models led to the development of conceptual TO-BE IDEF0 models, Yourdon-Demarco data flow models, process control flow models, and physical models. These conceptual models were developed to guide the adaptation of the baseline RAMP functions and interfaces to the TO-BE CPRMS conceptual architecture. A proprietary modeling technique, the allocation model, was developed to communicate the adaptation requirements to the customer. This allocation model was later expanded and used for illustrating the reusability of AMRC-developed software at the unit level and application of Commercial-off-the-Shelf (COTS) software modules to functional requirements.

One of the findings of this experience was that time and manpower required to complete the initial CPRMS conceptual architecture was significantly reduced by applying the generic RAMP baseline. A key lesson learned was that communication between the customer and systems engineers is essential in performing the adaptation from a generic design to a site specific architecture. Needs and requirements analysis has become an ongoing effort rather than a one-time activity. Feedback from the NADEP has been valuable in pointing out possible improvements in the allocation modeling and overall structured analysis approach.

This paper begins with an overview of the NADEP, reviews the baseline RAMP architecture, and provides an overview of the development of the CPRMS conceptual architecture. Detailed insights on implementation strategy, enterprise integration, and shop floor integration, are then discussed. This sets the stage for explanation of the allocation modeling technique. Finally, the lessons learned by the RAMP systems engineers during the conceptual architecture adaptation process are summarized.

2. Overview of the Cherry Point Naval Aviation Depot

The NADEP is located on the Marine Corps Air Station at Cherry Point, North Carolina. It is one of eastern North Carolina's largest employers, and has earned the Chief of Naval Material Productivity Excellence Award several times in the past for its performance. The Cherry Point NADEP is unique in that it is the only NADEP, of six nationwide, managed by Marines. The mission of the NADEP is to rework several aircraft, including the F-4 Phantom, the CH-46 Sea Knight, C-130,A4F, AV-8B Harrier, and OV-10 [FAR86].

In order to support the rework of aircraft, as well as the Navy's Aviation Supply Office demand for spare parts, the NADEP has an established Numerical Control (NC) Machine Shop and NC Programming capability. These shops support both the manufacture of SMPs from raw material, as well as the rework of aircraft parts.

During reviews of the RTIF SMP program in 1988, NADEP Manufacturing Engineering management recognized the relevance of the RAMP SMP to the NADEP NC Machine Shop, and understood the quality and throughput improvements that could be achieved by introducing the RAMP technology there. The NC Machine Shop was therefore selected as the pilot segment for the CPRMS implementation.

A second segment selected for the Cherry Point RMS was an Engine Blade/Vane Facility (EB/VF). The mission of the EB/VF is to rework large

quantities of retrograde blades and vanes from overhauled aircraft engines. An existing Interim Blade/Vane Shop, used for extensive prototype of Blade/Vane repairs, will be expanded to provide for full production capability in a 65,000 square foot facility.

As illustrated in Figure 2.1, the technology transfer approach derived for the SMP and EB/VF segments of the CPRMS was to implement each segment in two phases. The first phase of each segment will introduce RAMP technology by providing man-machine interfaces for controlling the generation and distribution of workstation instructions to shop floor artisans, and provide the capability to collect shop floor data for management.

The second phase of each segment will provide the capability to generate a complete set of technical data and control the release of orders to the shop. Automated material handling, in conjunction with the existing government furnished shop floor production equipment, will be configured to establish an automated Flexible Manufacturing System capability. Enterprise integration will provide for the linking of the segment with the site.

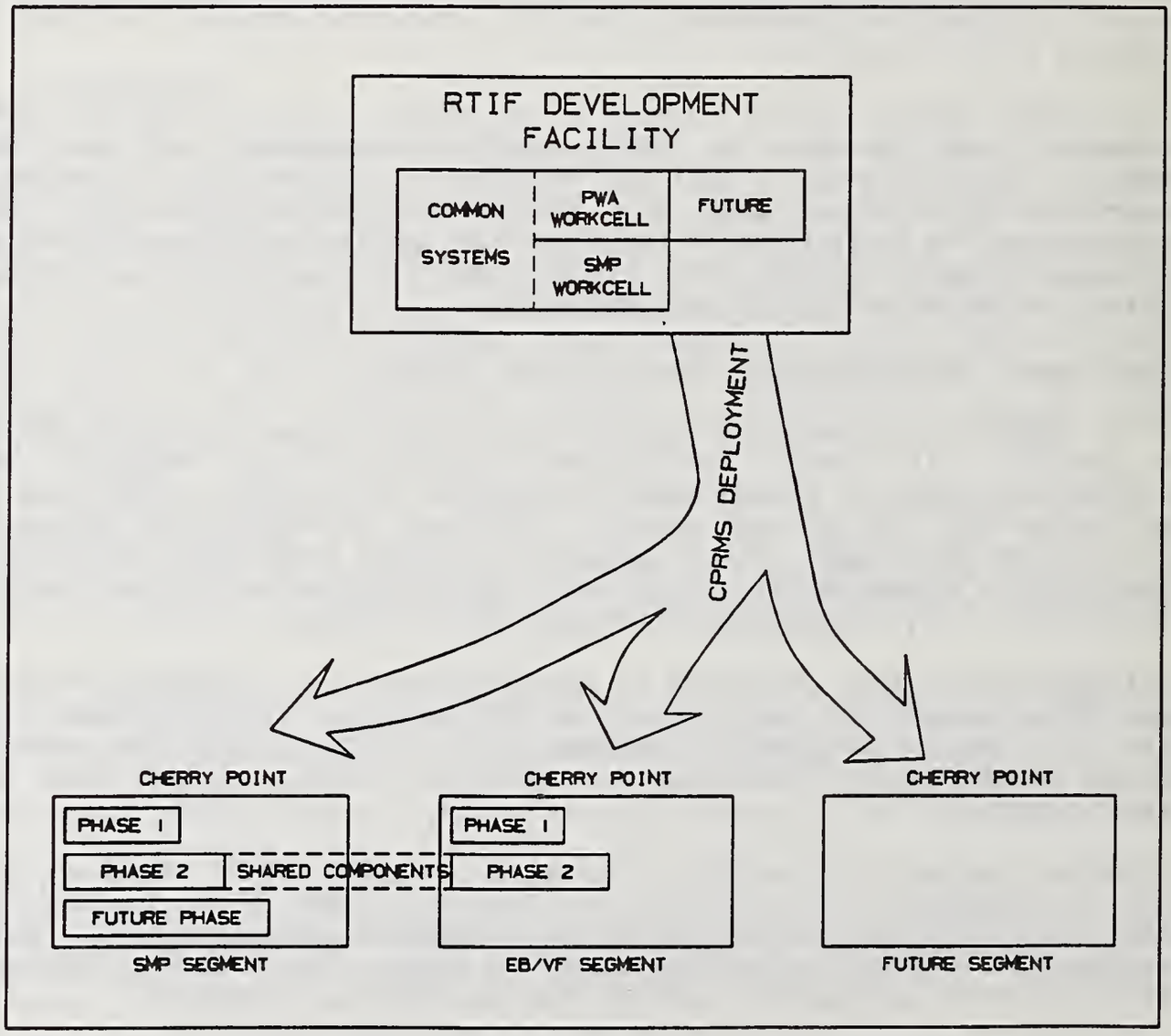


Figure 2.1. Cherry Point RMS technology transfer strategy.

As shown in Figure 2.1, a key consideration of the CPRMS architecture development was to provide for sharing of system hardware/software components during the second phase implementation.

3. Leverage of the RAMP Architecture

3.1 Retention of the RAMP Architecture baseline

During the process of allocating the generic RAMP architecture to produce the top-level CPRMS architecture, it was determined that the baseline of RAMP top-level system functional components would be retained. As described in [LIT90], the CPRMS definition of top-level system functions was baselined upon the generic RAMP concept of four functional Top Level Components (TLCs) - 1) Production & Inventory Control (P&IC), 2) Manufacturing, 3) Manufacturing Engineering (ME), and 4) Quality, integrated through three additional TLCs- 5) Control via RAMP Order Manager (ROM), 6) Information Management System (IMS), and 7) Communications.

In addition, it was determined that the baseline of enterprise functions, critical to supporting the RAMP system at an established site, would also be retained. The CPRMS definition of enterprise support functions was baselined upon the generic RAMP concept of interfaces between the CPRMS and a Site Order Receipt Activity, Quality Assurance (QA), Maintenance, Accounting, CPRMS Management, Material Management, Packing & Shipping, Receiving Inspection, Item Make or Buy Activity, Cognizant Engineering, and a Product Definition Exchange Specification (PDES) Quality Assurance and Generation Activity.

Another aspect of the RAMP architecture that was kept intact was the five (5) level Control architecture - enterprise, system, cell, workstation, and equipment.

The physical hardware/software configuration, as reflected in the physical implementation model, was largely adopted as a baseline for the CPRMS architecture. Since the generic RAMP architecture is highly modular, COTS substitution was a feasible option. However, a fundamental justification for maintaining the RAMP configuration of hardware platforms and COTS software packages for the CPRMS was to avoid potential time delays and cost impacts associated with technology assessment and interface software development.

This retention of the baseline of RAMP system capabilities was important, because any addition of system functions to the baseline would have made the adaptation process more difficult. For example, if the customer had requested that Accounting be added to the baseline of system functions, rather than remaining as an enterprise function, the CPRMS conceptual architecture development would have taken longer to complete.

3.2 AS-IS IDEF0 Modeling

The benefits of applying the existing RAMP architecture began with the first CPRMS task, the site survey. The baseline of enterprise functions was used as a basis for scheduling the NADEP enterprise support function interviews.

The next CPRMS task was documenting the site survey and AS-IS NADEP

infrastructure. Since the AS-IS modeling was unique to the NADEP, the RAMP architecture was not applied to this task. Of the modeling approaches available for documenting the AS-IS functions, the customer stated a preference for the IDEF0 function modeling methodology.

This modeling effort encompassed definition of interfaces between enterprise functions, as well as adapting interfaces between the enterprise functions and the CPRMS segments. This extension of the modeling boundaries, to include in-depth analysis of the site support functions, was performed to provide a clear understanding of the site needs.

Figure 3.2.1 shows the AS-IS IDEF0 A0 diagram for the SMP segment. The complete functional flow was modeled from Order Induction (order receipt), to Production Control, and then to the NC Programming and NC Machining shops, through completion of an order. The Order Induction process and Support

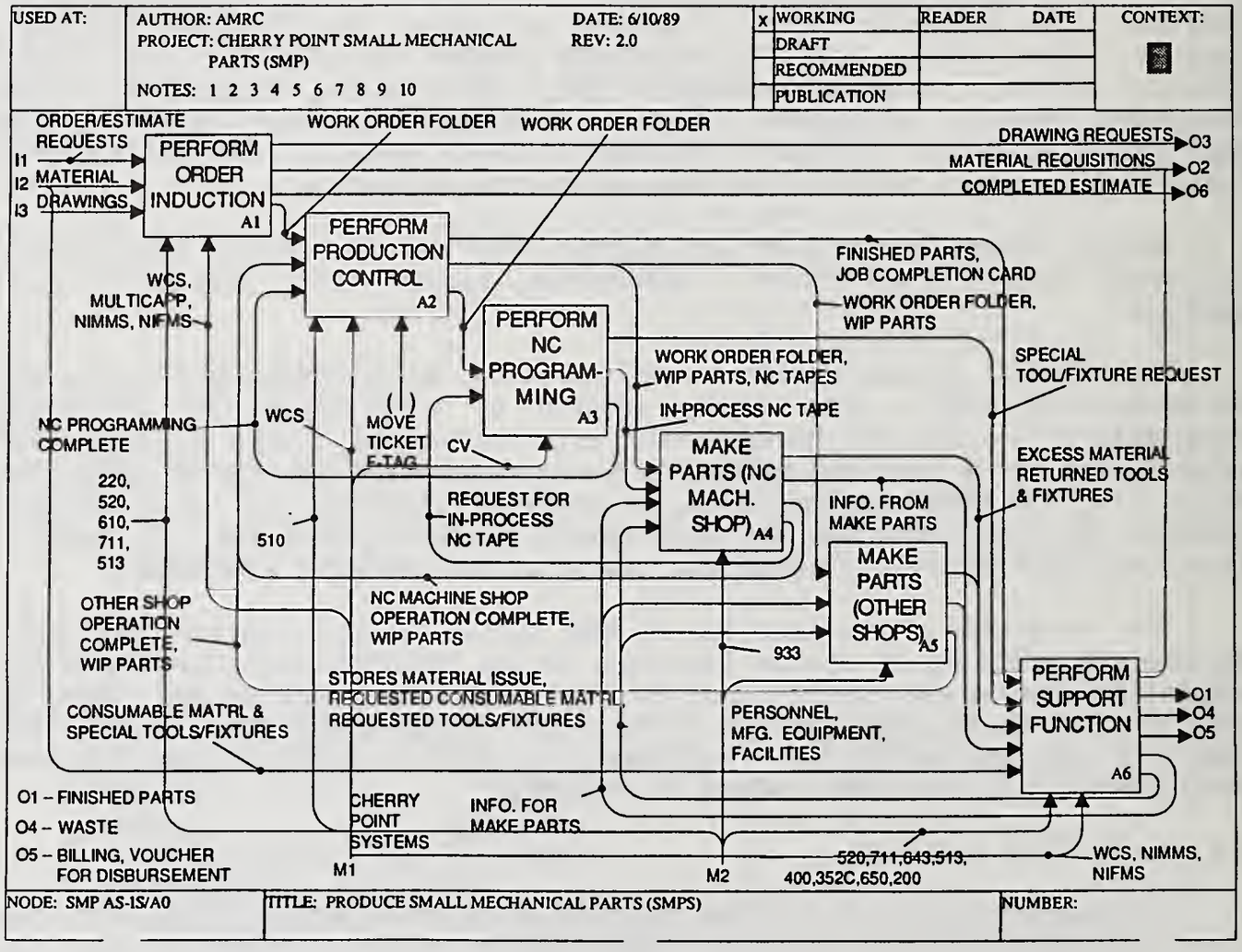


Figure 3.2.1. AS-IS IDEF0 A0 model for the SMP segment.

functions were modeled in detail in order to understand the key relationships between the NADEP Production Planning & Estimating, Accounting, and Material Management functions. A key aspect of the AS-IS IDEF0 diagrams was the use of controls to model the established NADEP business objectives and procedures, and the use of mechanisms to model the computer systems (e.g., WCS) and the responsible department codes (e.g, 220, 520) responsible for manually, interactively, or automatically performing functions.

Since the focus of phase 1 of the SMP segment was on the NC Programming and NC Machine Shop, in-depth modeling of these activities was performed. Figure 3.2.2 shows an example of the controls and mechanisms necessary for the SMP "Make Parts" decomposition. Since the ultimate goal of SMP phase 1 is to improve machine utilization by reducing setup and NC program prove-out time, IDEF0 functional flows were developed to illustrate the existing information flow problems.

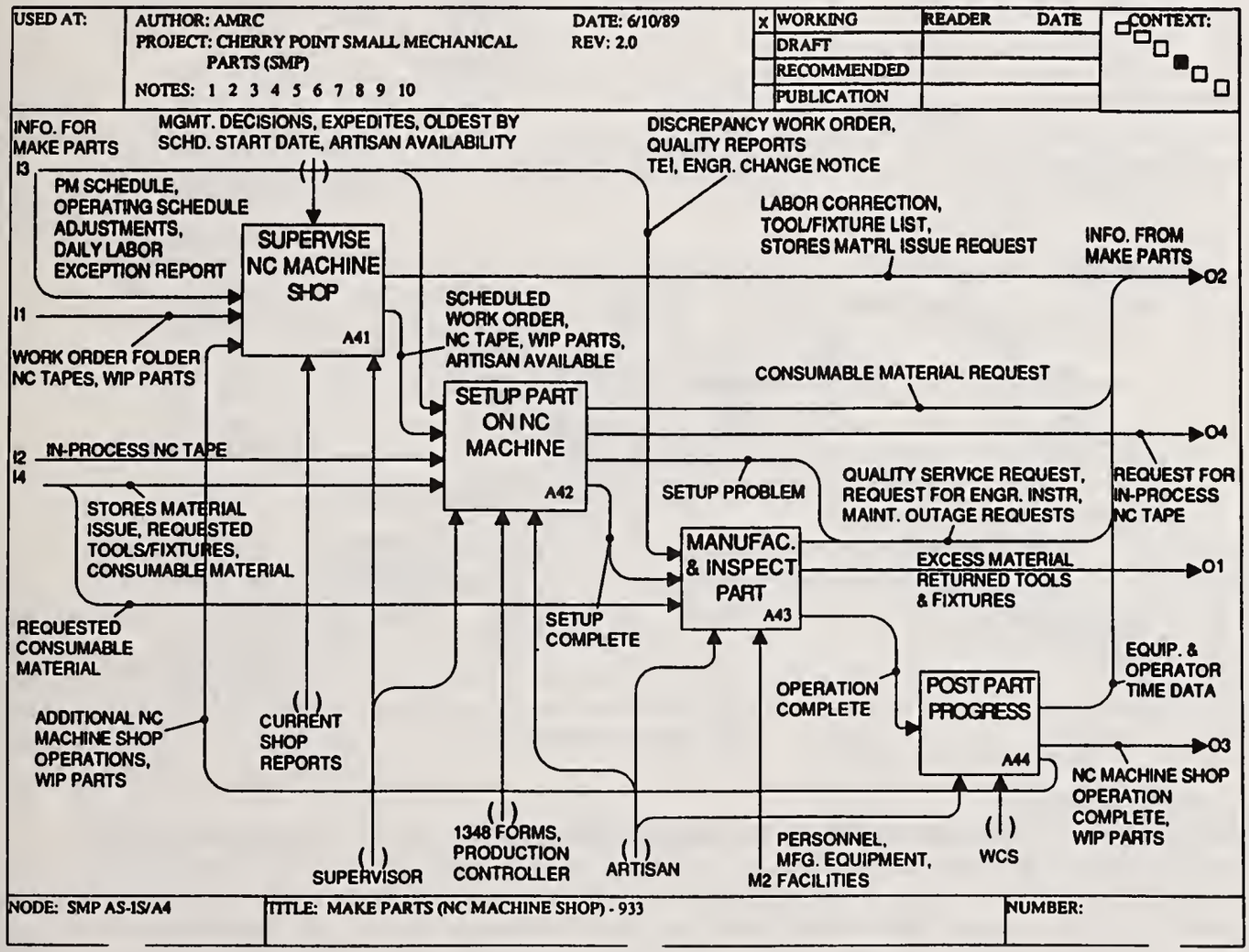


Figure 3.2.2. AS-IS IDEF0 A4 model for the SMP segment.

The basis of the EB/VF AS-IS IDEF0 modeling was the existing Interim Shop. As shown in Figure 3.2.3, the functions and the relationships between functions for performing the repair process are somewhat different than those shown in Figure 3.2.1. Specifically, the procedure of requisitioning blades/vanes to be repaired (retrograde material), and then determining the technical process of repair is exactly opposite the SMP procedure of determining a technical plan, and then requisitioning material.

The AS-IS IDEF0 modeling for EB/VF focused on repair methods development capability and the repair process. The latter consists of screening the blades and vanes, defining the repair process, performing repair operations, and performing post-repair inspection. Modeling of the repair process was important, since the impact on control and dispatching of repair orders through the EB/VF needed to be evaluated against the RAMP architecture to note any variances.

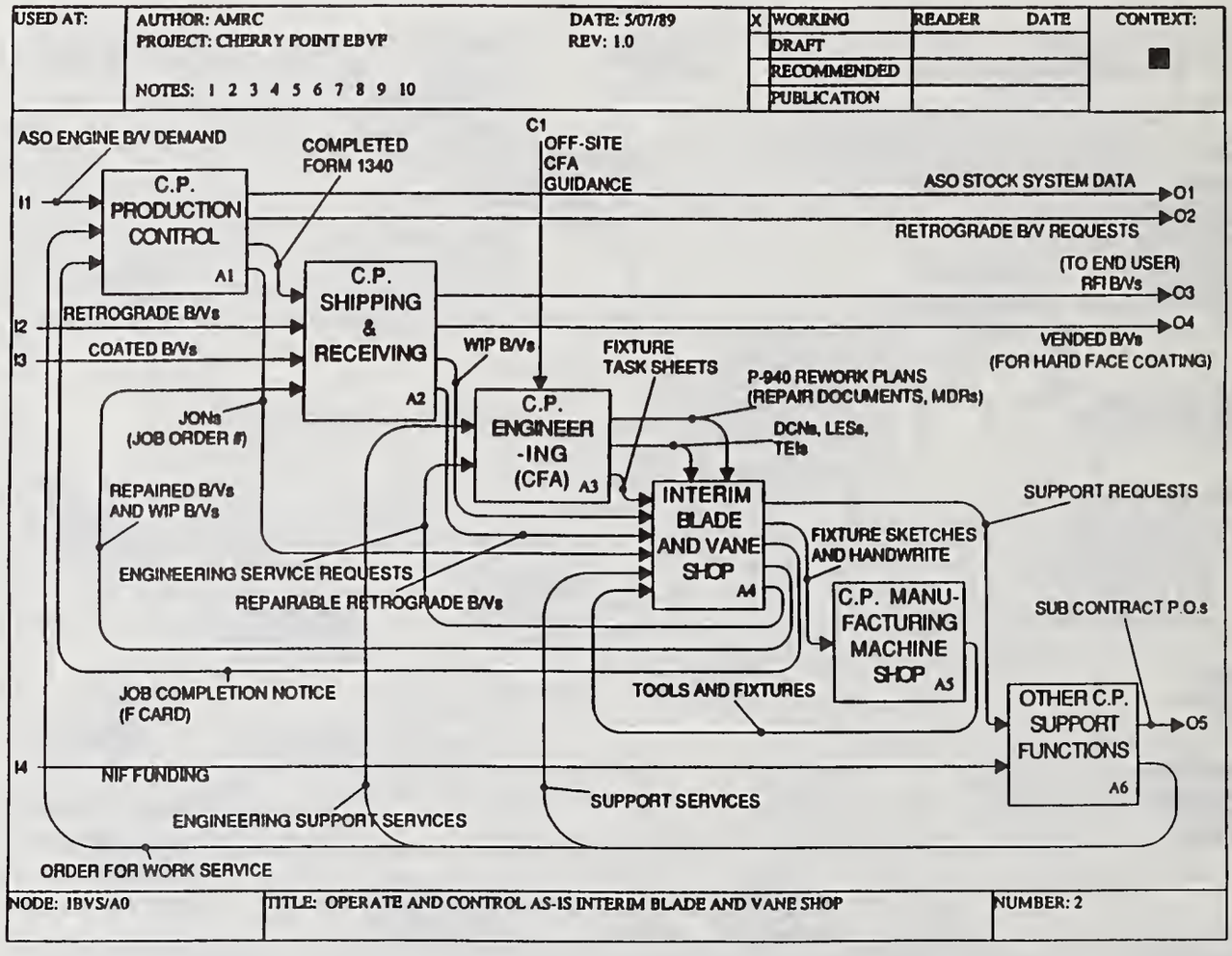


Figure 3.2.3. AS-IS IDEF0 A0 model for the EB/VF segment.

3.3 Adaptation of the RAMP Architecture to develop the TO-BE models

The generic RAMP architecture has the fundamental purpose of modeling the TO-BE system functions and enterprise functions and interfaces. The RAMP architecture is an aggregate of several perspectives of the RAMP system, including an IDEF0 functional model, a Yourdon-Demarco data flow model, a simulation model, a process control flow model, and physical models of the shop floor and hardware/software configuration. The RAMP architecture was developed over time by transforming each perspective of the reference model into other perspectives, starting with the IDEF0 functional model and finishing with the physical hardware/software configuration.

In developing the conceptual TO-BE CPRMS architecture, AMRC systems engineers were able to avoid much of this time-consuming transformation process, by applying the latest versions of the RAMP models. For each CPRMS adaptation issue, it was possible to reflect the solution in all of the RAMP modeling perspectives in a parallel mode. This reduced the initial conceptual CPRMS architecture development time and manpower significantly.

Following the completion of the AS-IS modeling, TO-BE IDEF0 functional models were created. The RAMP Architecture IDEF0 functional models were not applied, since they had already been outdated by transformation into the Yourdon-Demarco data flow model and process control flow diagrams. However, creation of the CPRMS IDEF0 models was made considerably easier by knowledge of the inputs, outputs, controls, and mechanisms present in the RAMP Architecture. Figure 3.3.1 shows the A0 level model for the SMP segment, which reflects the RAMP baseline functions:

- Functional TLCs P&IC, Manufacturing, ME, and Quality are shown as A1 through A4,
- Control over the processing of messages through the functional components is provided by the ROM TLC,
- The mechanisms for the transmission of messages and data between functional components is provided by the IMS and Communications TLCs; the physical hardware/software, personnel, and equipment mechanisms are also shown.

In developing the CPRMS architecture, one of the features found lacking in the IDEF0 modeling methodology by the systems engineers was the ability to easily follow the end-to-end sequencing of processes and branching necessary to process customer orders. This problem can be seen in Figure 3.3.1, by the numerous flows of the diagram. Therefore, the process control flow diagrams developed in the RAMP architecture were adapted to document the end-to-end perspective of the system segments. Process control flow diagrams were developed for both segments.

Adaptation of the RAMP architecture Yourdon-Demarco data flow models was also considered for use in the CPRMS. It was determined that these data flow models would not be useful because they represented a top level design with generic functions and data stores, whereas the RTIF SMP system was already under construction with application specific COTS packages and databases. To make these models useful, it would have been necessary to create an as-built Yourdon-Demarco to reflect the COTS modules used.

Figure 3.3.1. TO-BE IDEF0 A-0 model for the SMP segment.

In the end, only the top level Yourdon-Demarco data flow model was used in the adaptation, because of its illustration of interfaces between the RAMP functional components and the site functions. Top level Yourdan-Demarco diagrams were developed for both phases of the SMP segment, and for phase 2 of the EB/VF segment.

Perhaps the most effective modeling method used for documenting the implementation of the system and communicating the system requirements was the physical hardware/software configuration model. This model provides a basis for understanding the control architecture, the logical flow of information through the system, as well as understanding the bill of assembly of modules that comprise the system. The effectiveness of the physical model was demonstrated by the overwhelming demand for copies of the model by AMRC designers. Physical hardware/software configuration models were developed for both phases of the SMP segment, and for phase 2 of the EB/VF segment.

The following section provides more technical detail on the factors influencing the adaptation for each phase of each segment. Examples of the CPRMS top level Yourdon-Demarco models and physical hardware/software configuration models are provided.

4. Site specific factors

As discussed in section 2, a multi-phase implementation strategy was desired by the NADEP for the CPRMS to accomodate cost, schedule, and military contruction transition plans. This bottom-up, phased strategy was also found to make the introduction of technology more acceptable to the workforce, and to reduce the implementation risk. Within each implementation phase, enterprise integration and shop floor integration issues were considered.

4.1 SMP phase 1 implementation strategy

The first phase of the SMP, scheduled for completion in August, 1990, will integrate the NADEP NC Programming and NC Machine Shop areas. As illustrated in the AS-IS physical model of Figure 4.1.1, the existing hardware/software configuration in these areas includes:

- an enterprise level communications network and personal computer (PC) workstations used by the site support functions,
- an existing Computer Aided Design/Computer Aided Manufacturing (CAD/CAM) system, and
- 12 existing machine tools.

The TO-BE physical model of Figure 4.1.2 illustrates the integration that will take place through replacement of the CAD/CAM system, addition of a file server and database management software for configuration management of NC programs, addition of PC workstation controllers for communicating with the machine tools, and establishment of additional network capability to link the servers with each other, the workstation controllers, and site PCs. This TO-BE physical model was adapted directly from the generic RAMP architecture model. The file server function, which also provides system level control, was moved from the RAMP system level to the CPRMS cell level for the phase 1 implementation.

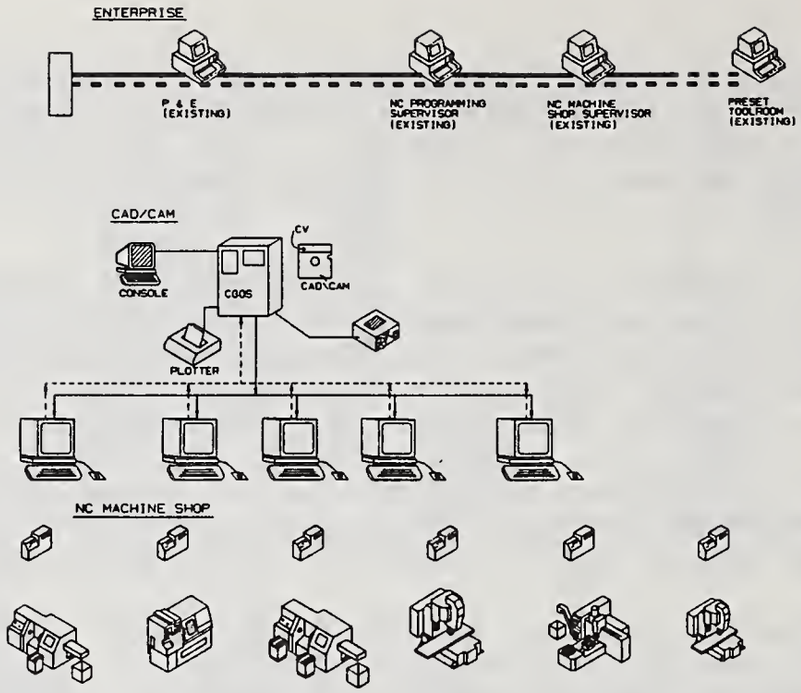


Figure 4.1.1. AS-IS NADEP NC Machine Shop, NC Programming, and enterprise functions.

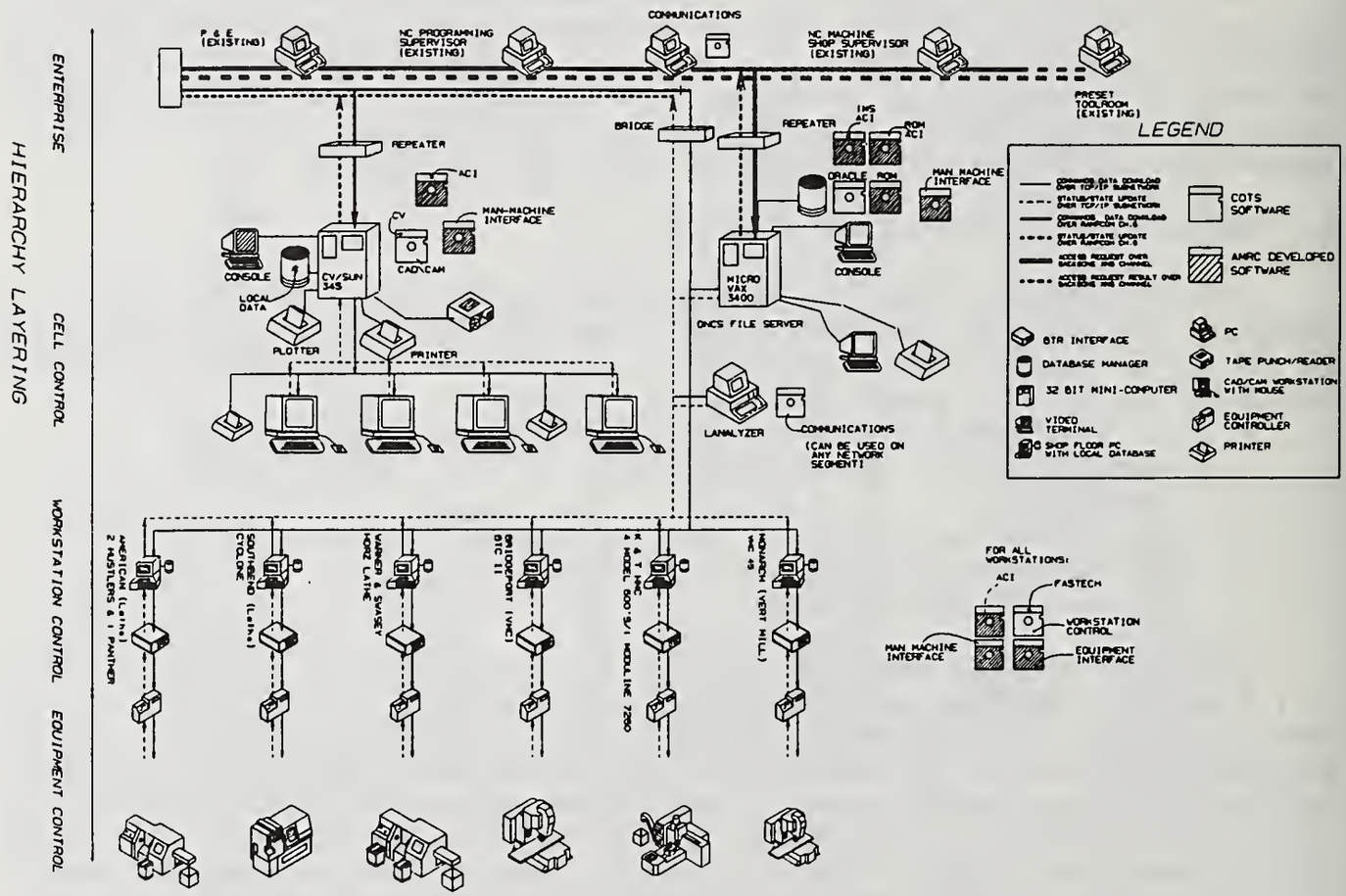


Figure 4.1.2. TO-BE SMP phase 1 physical hardware/software model.

Since the integration of existing site specific machine tools was not addressed by the RAMP architecture, SMP phase 1 presented a challenge to provide the appropriate integration technology. Ultimately, Behind-the-Tape-Reader (BTR) technology was selected to meet the requirements.

The goal of SMP phase 1 is to produce a packet of workstation instructions (NC Data, artisan text instructions, and tool lists) using the CAD/CAM system, and provide a man-machine interface to maintain configuration of these instructions in the IMS. A man-machine interface was also designed for NC programmers and artisans to request the distribution of these workstation instructions electronically to any of 12 workstation controllers used for download of NC data to the existing machine tool controllers.

Other requirements of SMP phase 1 are to collect machine status update data from the workstation controllers, and establish an engineering service communications mechanism between the shop floor artisans, NC Machine Shop supervisor, and NC programmers. The capability for NC Programming and Machine Shop supervisory personnel to formulate Structured Query Language (SQL) queries to access the relational database system of the IMS was also incorporated in the architecture.

In addition to the TO-BE physical model of Figure 4.1.2, a one-page process control flow model of SMP phase 1 was developed to effectively show the flow of information described above.

4.2 EB/VF phase 1 implementation strategy

The first phase of the EB/VF, scheduled for completion in March, 1991, will integrate the existing Interim Shop. The existing government furnished hardware/software configuration will include:

- the enterprise level communications network and PCs for the EB/VF management and support areas, and
- over 100 production equipment items, including an Automated Guided Vehicle system (AGVS) and Automated Storage and Retrieval System (AS/RS).

The first phase of the EB/VF will include functional requirements and a configuration very similar to the SMP phase 1. An additional feature provided will be the capability to send material handling requests to the production controller, for interactive entry of AGV controller commands.

The phase 1 systems will operate on a stand-alone basis, and will not be integrated with one another until phase 2.

4.3 SMP phase 2 implementation strategy

The second phase of the SMP, scheduled for completion in November, 1991, will overlay the phase 1 implementation to complete the CPRMS SMP segment. A key design decision was to retain the man-machine interfaces developed in SMP phase 1 as an alternate means of performing essential processes such as NC programming and workstation instruction requests.

In this phase, part technical data will be generated using the RAMP PDES Generation System (RPGS) technology, in accordance with the generic RAMP

architecture. PDES part technical data will be used to automatically create the shop routing, and will be semi-automatically converted to the CAD/CAM system native part geometry, in order to improve the consistency and throughput of workstation instruction preparation. In addition, however, the CPRMS architecture will provide for the establishment of part technical data using existing NC programs, if desired.

Per the RAMP Architecture, P&IC capabilities will be provided to enter, track, schedule, and control the flow of customer orders and materials through the NC Programming area and NC Machine Shop in a semi-automated manner. The ability to reserve and allocate the finite capacity of the shop floor will be provided, and control over the release of RAMP work orders will be introduced.

Implementation of P&IC for the phase 2 SMP will be impacted by the physical location of the production controller outside of the NC Machine Shop. The production controller has an existing workstation on the enterprise network, so the CPRMS architecture will accomodate the ability to access RAMP functions from this workstation. This is reflected in the physical hardware/software implementation model shown in Figure 4.3.1.

Implementation of P&IC will also be impacted by the AS-IS enterprise functionality. Currently, the Planning & Estimating (P&E) branch generates inter-shop routings for parts that travel through all of the NADEP shops, including heat treating and plating, to complete an SMP. It was determined that it would be best for P&E to continue to use their existing group technology system for generating the complete inter-shop work order, and that the generation of the inter-shop work order would also generate a CPRMS SMP order.

Another feature unique to the CPRMS will be the capability to respond to requests for estimates (probes). Generation of estimates for potential RAMP order candidates is a requirement not in the RAMP baseline.

The generic RAMP architecture provides for inventory control capabilities to request and track the receipt of direct and consumable materials, tools, and fixtures. The greatest impact on inventory control for the CPRMS SMP will be the physical mechanisms provided to receive materials from supply storerooms, work-in-process parts from other NADEP shops, and tools and fixtures from NADEP support departments. The data flow required for these interfaces is illustrated in the top level Yourdon-Demarco diagram shown in Figure 4.3.2. An additional physical model was developed to clarify the physical interfaces.

The capability to perform shop floor resource scheduling, and to automatically control the flow of RAMP work orders on the shop floor between workstations will be introduced in phase 2. The CPRMS SMP will be a manned, flexible manufacturing cell configuration, with transportation provided by artisans, and fixture/unfixture performed at the workstation, rather than at a remote area. This will impact the concept of automated transport capability inherent in the RAMP architecture. However, addition of a Coordinate Measuring Machine (CMM), and upgrade of the existing preset toolroom area, will improve the capability to make quality parts in a flexible manner.

Figure 4.3.2 illustrates other modifications that were made to the generic RAMP model to accomodate site specific needs. The interface to Quality was de-

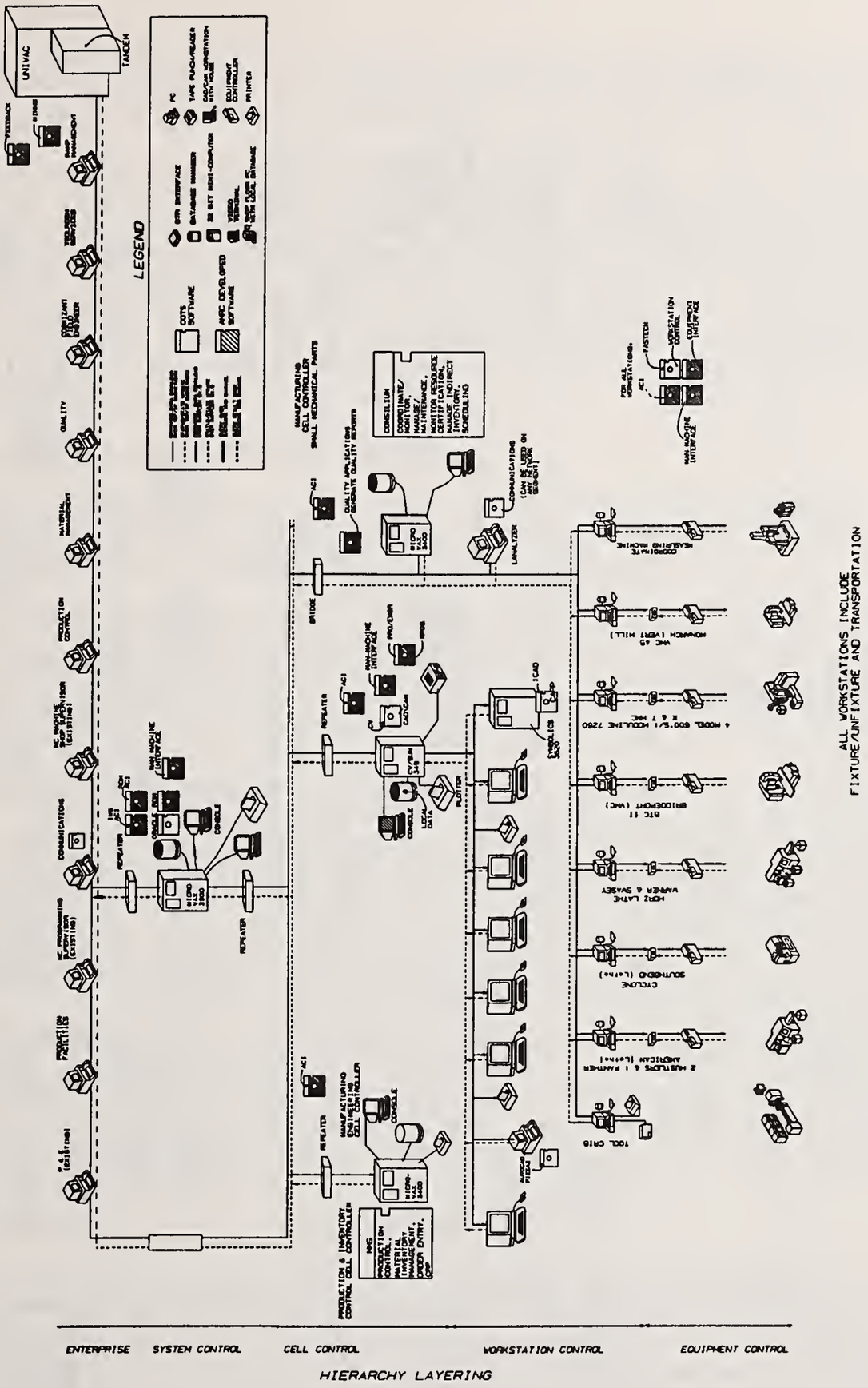


Figure 4.3.1 SMP Phase 2 Physical Model.

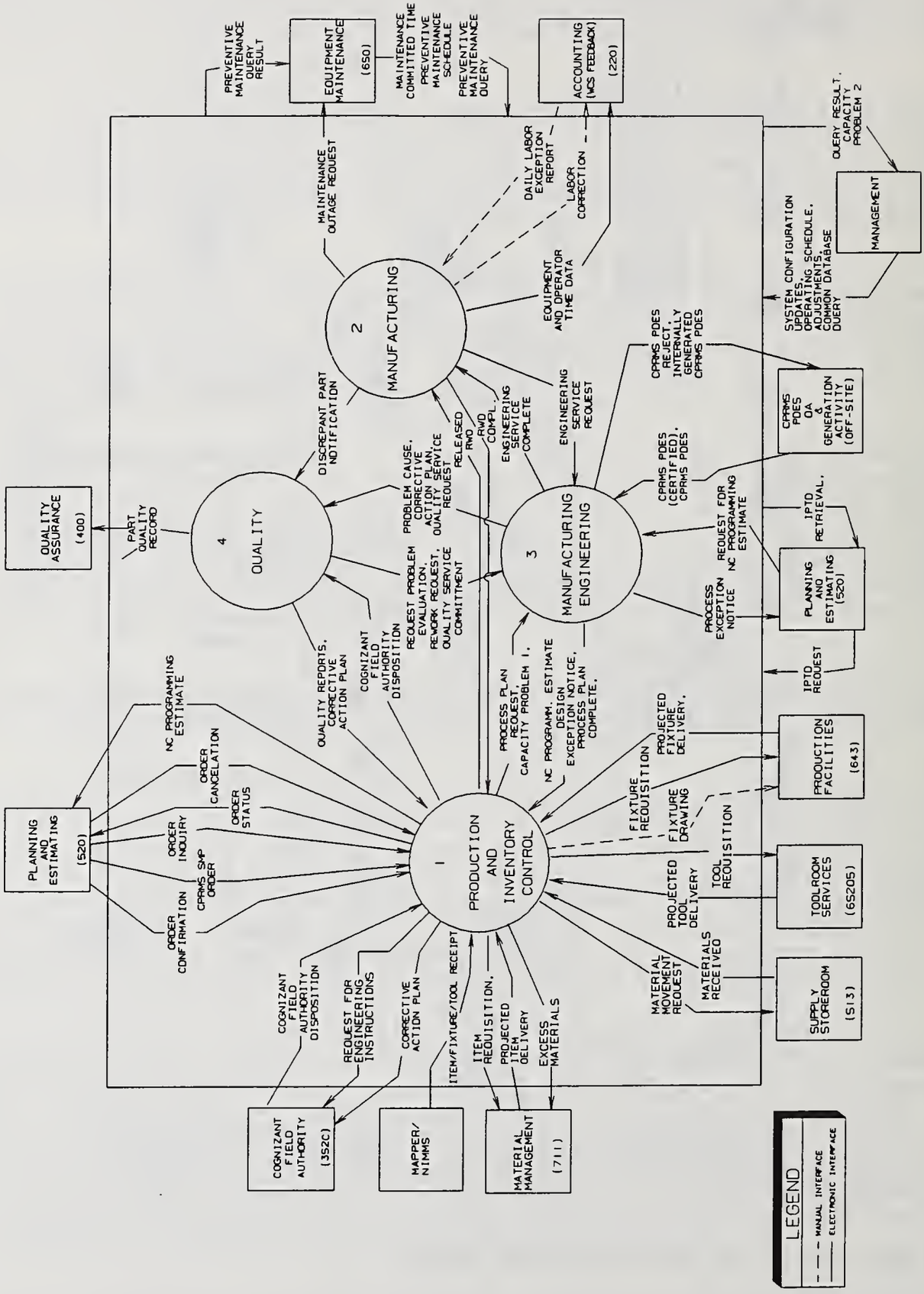


Figure 4.3.2. SMP phase 2 Yourdon-Demarco model.

emphasized, when the decision was made that parts would be reworked if easily resolved problems occured, but parts would be scrapped rather than analyzed through a Material Review Board (MRB) process, if the rework is complex.

It was also determined to simplify the interfaces to Maintenance to take advantage of the existing NADEP computer systems and procedures already in place, and accepting a degree of manual and semi-automated interfaces. This option was selected rather than pursuing a costly alternative of automating interfaces for which there was limited programming support. It was determined that the current method of phoning a central Call Desk for reporting machine outages to Maintenance would be a more prudent alternative than developing an automated interface from the CPRMS to the NADEP computer system which performs the dispatching of Maintenance personnel.

4.4 EB/VF phase 2 implementation strategy

The second phase of the EB/VF, scheduled for completion in February, 1992, will overlay the phase 1 implementation, and integrate with the SMP phase 2 implementation through shared hardware/software components.

The EB/VF implementation will not use PDES files to generate technical data. Instead, the system will provide for the establishment of part technical data repair routings, to be called out based on a repair category. Additional TO-BE IDEF0 and process control flow modeling was performed to document the requirements for the unique technical data generation capability of the EB/VF.

Integration with enterprise functions will be very similar to the SMP phase 2. The EB/VF implementation will also use the P&E department for generation of CPRMS orders. Determination of the methodology for accounting of order costs was a significant effort, because of the proprietary nature of the NADEP cost accounting policies. The EB/VF phase 2 segment will be more of a stand-alone facility than the SMP phase 2, and thus will need the full RAMP-like interface with QA and Packing & Shipping departments. Special interfaces will be needed to requisition retrograde blades and vanes.

A key feature of the implementation strategy will be to use the shared architectural elements developed in SMP phase 2. The P&IC, Manufacturing Engineering, ROM, IMS, and Communications TLCs will be shared with the SMP, as reflected in the physical hardware/software implementation model shown in Figure 4.4.1. A dedicated Manufacturing Cell Controller will be provided to ensure the reliability of shop floor control.

5. Development of the allocation model

All of the RAMP program software development plans and documents follow the intent of the MIL-STD-2167A methodology. One of the conventions of this methodology is to define and decompose a Computer System Configuration Item (e.g., the CPRMS) in terms of its TLCs, Lower Level Components (LLCs), and units. The seven (7) top level components of the CPRMS are those listed in section 3.1. As Figure 4.3.2 indicates, the numbering system embedded in the RAMP architecture Yourdon-Demarco models provided a structure for identifying the TLCs, LLCs, and units. This structure proved to be essential for allocating the RAMP architecture to the CPRMS.

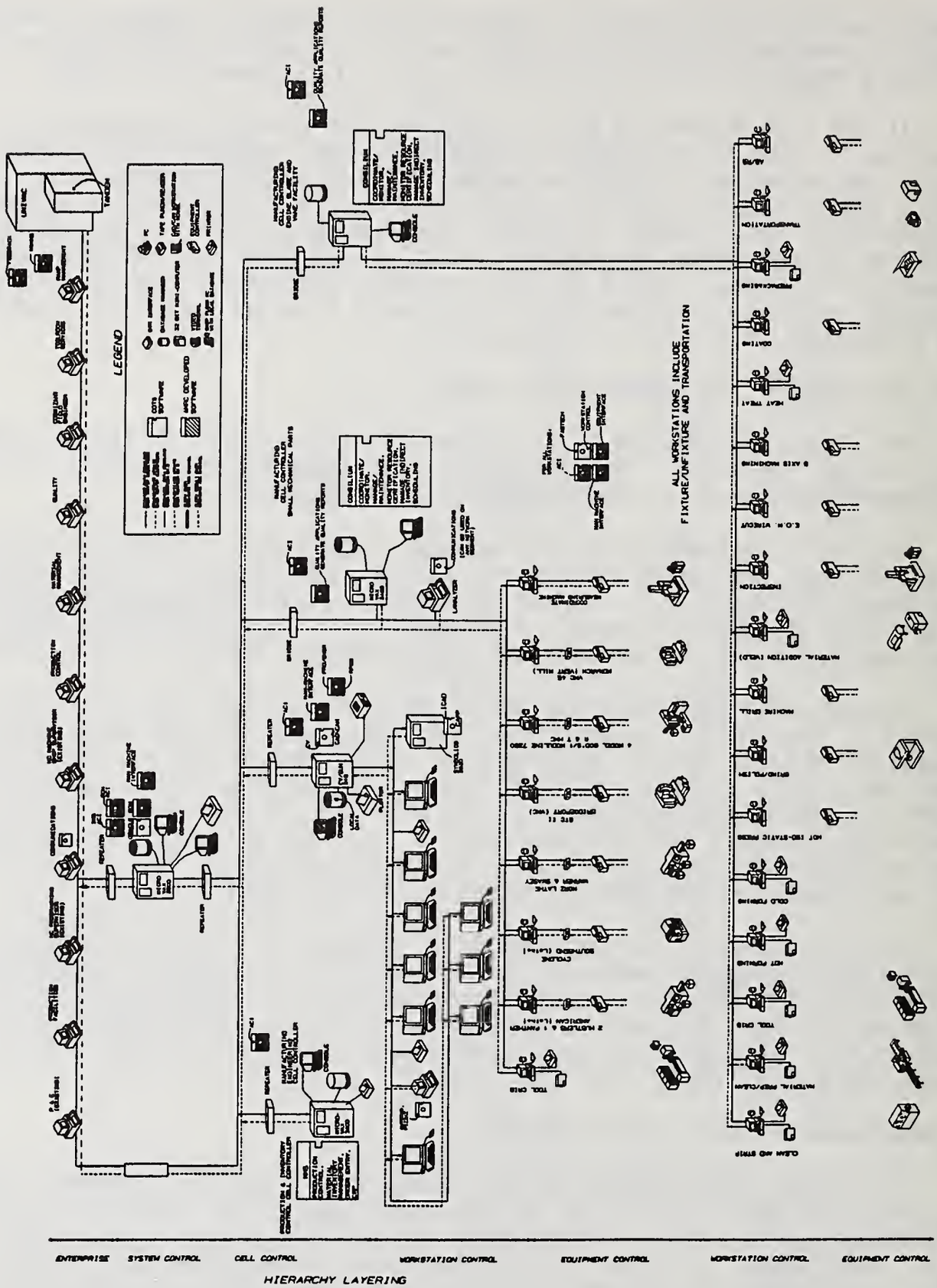


Figure 4.4.1 EB/VF Phase 2 Physical Model.

Following the completion of the needs analysis amd modeling effort, it was possible to determine the allocation of the generic RMS TLCs and LLCs to the CPRMS SMP and EB/VF phases. A subset of the allocation is shown in Table 5.1. Allocations were designated as full(F), partial(P), variant(V), or not applicable (N/A). A full allocation indicates that all the capability of the generic RMS functionality will be provided. A partial allocation indicates that a subset of the generic RMS functionality will be applied to the CPRMS segment. A variant allocation was used to designate additional functionality needed which was not present in the generic RMS baseline, and which required COTS procurement and/or AMRC-developed software to satisfy. A not applicable allocation indicates that the functionality was not needed.

The allocation for the SMP phase 1 will be used as an example to show how the technique works. The baseline functions that were allocated for use in SMP phase 1 included Manufacturing, ME, Control (ROM), IMS, and Communications, as indicated in Table 5.1. The Control and Communications components of the generic RMS were allocated virtually intact. The Control functions were not changed; only the tables which provide task sequencing were a partial implementation of the generic set of components. Along with the full implementation of the relational Database Management System (DBMS), these functions provided the basis for establishing a core of the generic RMS, which would be fully configured in the SMP Phase 2 segment.

A glance at the SMP phase 1 allocation indicates partial and variant implementations of the Manufacturing, Manufacturing Engineering, and Information Management components. The partial allocations were due to the limited phase 1 implementation. Additional functionality, as indicated by the variant codes, were due to the development of man-machine interface capability, where none existed in the RAMP architecture for the NC programmers and shop floor artisans.

6. Summary of lessons learned

By applying the RAMP architecture to the CPRMS architecture development, an initial conceptual CPRMS architecture was developed in just six (6) months. The conceptual architecture included the AS-IS IDEF0 models, TO-BE IDEF0 models, top level Yourdon-Demarco models, process control flow models, and physical hardware/software configuration models.

By comparison, approximately two years were required to transform the basic functional requirements and technology assessment into a physical hardware/software configuration model for the initial conceptual RAMP architecture.

The primary factor that made this improvement possible was the acceptance by the NADEP of the baseline generic RAMP architecture.

6.1 Lessons learned from SMP phase 1 Implementation

Since the June, 1989 timeframe, the primary focus of the CPRMS development has been the implementation of the SMP phase 1. SMP phase 1 has gone through the additional steps of top level design, detailed design, and unit coding, and thus provides a basis for evaluating the effectiveness of the conceptual architecture. It was found that the one-page process control flow diagram,

Table 5.1. Allocation of RAMP TLCs and LLCs to the CPRMS.

LEGEND:
F-FULL ALLOCATION OF RAMP/RTIF, P-PARTIAL ALLOCATION,
V-VARIANT ALLOCATION, N/A-NOT APPLICABLE

SEGMENT		SMP SEGMENT		E B / V F	
	PHASE	1	2	1	2
TLC/LLC					
1.0	Production and Inventory Control	N/A		N/A	
1.1	Capacity Requirements Planning (CRP)		F		V
1.2	Production Control		P		F
1.3	Order Entry		V		V
1.4	Material Inventory Management		V		V
2.0	Manufacturing				
2.1	Schedule Manufacturing Cell	N/A	F	N/A	V
2.2	Manage Maintenance	N/A	F	N/A	F
2.3	Coordinate/Monitor Mfg Cell	P,V	F	P,V	F
2.4	Manage Indirect Inventory	N/A	F	N/A	F
2.5	Workstation Control	P,V	F	P,V	F
2.5.1	Application Control	F	F	F	F
2.5.2	Man-Machine Interface	P,V	V	P,V	V
2.5.3	Equipment Control	P,V	V	P,V	V
2.6	Transportation Control	N/A	F	P,V	F
3.0	Manufacturing Engineering				
3.1	Create Process Plan	P,V	P	P,V	P
3.2	Evaluate Problem Cause	P,V	F	P,V	F
3.2.1	Application Control	F	F	F	F
3.2.2	Man-Machine Interface	P,V	V	P,V	V
3.3	Generate PDES	N/A	F	N/A	N/A
4.0	Quality	N/A		N/A	V
4.1	Generate Quality Reports		F		V
4.2	Coord. Disp'tion of Quarantined Part		F		V
4.3	Assemble Part Pedigree		F		V
4.4	Generate Part Quality Record		F		V
4.5	Monitor Resource Certification		F		F
5.0	Control				
5.1	Control Functions	F	F	F	F
5.2	Control Tables	P	F	P	F
6.0	Information Management				
6.1	File and Table Management	P	F	P	F
6.1.1	Application Control	F	F	F	F
6.1.2	Man-Machine Interface	P,V	F	P,V	F
6.2	DBMS	F	F	F	F
7.0	Communications	F	F	F	F
7.1	RAMPCOM (Developed)	F	F	F	F
7.2	RAMPCOM (COTS)	F	F	F	F
7.3	Command/Status Services	F	F	F	F

top-level Yourdon-Demarco data flow diagram, and physical model were sufficient in guiding the top level design effort. Each functional block of the process control flow diagram was decomposed in greater detail, culminating in identification of software units. The allocation model was later expanded and used for illustrating the reusability of AMRC-developed software at the unit level and application of Commercial-off-the-Shelf (COTS) software modules to functional requirements. This resulted in determination that 104 units of code from the generic RAMP Manufacturing System configuration were reusable, and that 42 units of code had to be developed to satisfy the variant functions.

Because of the variations in implementing the man-machine interfaces and interfaces to existing NADEP systems, on-going time and manpower savings from applying the generic RAMP architecture have not been as significant. For example, detailed design of the CAD/CAM system and shop floor machine tool upgrades to provide the appropriate system integration were more difficult than first thought.

The lessons learned in SMP phase 1 are expected to result in significant manpower and time savings for EB/VF phase 1.

6.2 Lessons learned during phase 2 preliminary design

Phase 2 of the SMP and EB/VF are both in the preliminary design stage. Since the June, 1989 timeframe, the effort on the phase 2 conceptual architecture has been minimal. However, a few conceptual architecture modifications were made to reflect additional information gained from the site.

For example, the need for the production controller to access the CPRMS SMP segment from an existing site workstation was determined during this period. In evaluating the impact of this function, the AMRC system engineers were able to analyze the effect on the functional model, data flows, process flows, and physical hardware/software configuration. It was possible to perform the analysis of all modeling perspectives in a parallel mode, rather than developing different modeling perspectives sequentially. This continued to provide time and manpower savings.

One of the lessons learned, therefore, is that needs and requirements analysis becomes an ongoing effort rather than a one-time activity. Feedback from the NADEP has been valuable in pointing out possible improvements in the conceptual architecture.

Another lesson that was reinforced during the conceptual architecture development is that several modeling approaches, as opposed to a single modeling approach, is a preferred method of describing an Architecture. No modeling method available provides a total perspective of a system.

In terms of the modeling effort, AS-IS IDEF0 models have stood the test of time in documenting the existing site operations. One lesson learned during this task was to develop a standard method for conducting site survey interviews.

Further documentation of the TO-BE conceptual architecture is under study. At the June, 1989 timeframe, the NADEP was satisfied with the level of conceptual architectural modeling that had been performed. Later, as the NADEP

became more proficient in understanding the RAMP architecture and capabilities, additional detail on the allocation process was desired. As the NADEP retrospectively examined the conceptual architecture, the aggregate of the models used - IDEF0, data flow, process control flow, physical, and allocation - was found to be insufficient in communicating the allocation from the generic RAMP architecture to the CPRMS conceptual architecture. The AMRC systems engineers are continuing to explore alternate structured analysis techniques that will succinctly illustrate the threads from the conceptual CPRMS architecture phase 2 requirements to the top level design, detailed design, and ultimately to unit coding.

6.3 Future architecture development efforts

As mentioned in the previous section, a key lesson learned from this adaptation exercise was to expect continuous challenges from the end-user site to keep the conceptual architecture updated and be able to efficiently demonstrate the effects of modifications throughout the entire system design.

The AMRC systems engineers have identified some of the most difficult aspects of the implementation effort, which have not yet been totally resolved in the conceptual architecture. These will require additional time and manpower to address. Foremost among these unresolved issues will be to address the cultural barriers to integration, particularly with regard to the CPRMS staff job descriptions and the NADEP accounting practices.

The testing phase of the RTIF system was recently completed, and resulted in the recommendation of several improvements to the system. As the RTIF architecture is adapted to reflect these improvements, the CPRMS architecture will be updated, as well.

7. References

[FAR86]
Fargher, John S. W. Jr., "Using the Modular Approach for Development of Computer Integrated Manufacturing Systems," International Industrial Engineering Conference Proceedings, 1986.

[LIT90]
Litt, Eric, "The Development of a CIM Architecture for the RAMP Program," CIMCON '90 Proceedings, 1990.

AN APPROACH TO DEVELOP AND MAINTAIN DATA QUALITY.

MAJOR KNUTE E. HANKINS, U.S. ARMY.

Abstract: The Watervliet Arsenal (WVA), Watervliet, New York is currently installing a Shop Floor Control (SFC) system in its factory. This system consists of two industry standard scheduling software packages, ASK MANMAN (MRP) and FACTOR (Finite Scheduling). In addition, a customized Tool & Gage Inventory and Accounting System is in development to complete the architecture of the overall SFC system within a custom system shell. A major effort in this project has been the correction and verification of multiple independent arsenal resident databases utilized for existing systems. The integration of these databases into an integrated system such as SFC has resulted in a much greater accuracy requirement in order to make SFC work. This paper presents the approach used by WVA to establish identifiable parameters for data, establish quantifiable measures to ensure data "accuracy", and methodologies to verify data. Finally, this paper reviews the impact of this work on the overall SFC project.

1. Introduction.

The Watervliet Arsenal (WVA), Watervliet, New York is a component of the United States Army Armament, Munitions, and Chemical Command. The Arsenal's mission is the engineering, procurement, fabrication, assembly, and product assurance of all thick wall cannon for the U.S. Department of Defense. As part of its long range automation plans, WVA has purchased and is in the process of implementing and integrating a Shop Floor Control System (SFCS) into current arsenal information systems. This system consists of two industry standard software packages, ASK MANMAN (MRP) and FACTOR (Finite Scheduling). In addition, a customized Tool & Gage Inventory and Accounting System is in development to complete the architecture of the overall SFCS within a custom shell. The SFCS will meet the arsenal's objectives to enhance management control of the activity on the shop floor, provide 100 percent accountability of tooling, and produce daily manufacturing schedules through the use of Material Requirements Planning (MRP) and Finite Capacity Forward Scheduling.

The implementation of the SFCS is managed by a staff of eleven personnel. Most members were assigned from future using organizations. This philosophy is used to ensure user requirements are tied to the SFCS as it evolves. The implementation strategy for the system is divided in two phases. The arsenal divides its production areas into three branches. Minor Components Branch manufactures small components, resulting in a high volume, high product mix environment. The rest of the production areas are divided into Major and Tubes Branches. These areas deal in much larger components resulting in a lower component mix and less overall component quantities. Although the strategy has deviated somewhat, the primary focus and long term development work for the SFCS and its associated databases has been in the Minor Components Branch. Once the system has proved out in this Branch, the system will be expanded to encompass the entire arsenal. Currently, personnel in Minor Components Branch are being trained and prepared to test the system. Due to the volume of information required to operate the system for the arsenal, the data describing Majors and Tubes Branches' components continues to receive extensive review.

One of the greatest challenges in the implementation of the SFC core system is to interface, integrate, or replace at least 11 existing databases (Figure 1).

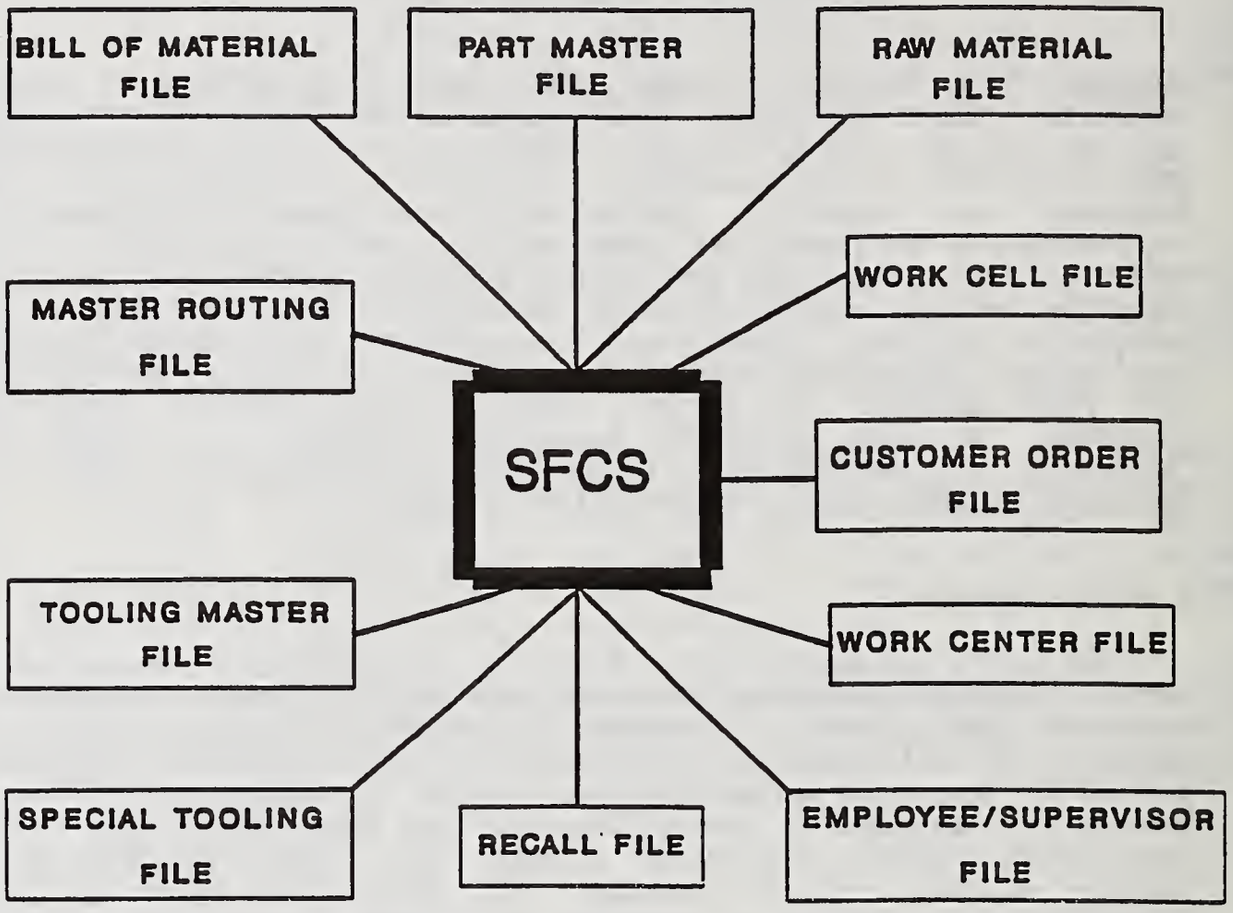


Figure 1. SFCS Database Relationship.

The accuracy and consistency between these databases must be extremely high for the effective operation of SFCS as an integrated database. In order for the scheduling and resource functionality of SFCS to work correctly, any data field describing a component, the required materials, tools, and process to manufacture it, and the schedule and quantity requirements dictating its production have to be consistent throughout all the databases. Oliver Wight, a major author and integrator of MRP systems, was quoted as saying that the most critical databases are the Bill of Materials (BOM) which identify the components making up an end item, Routings, and Inventory Records. He went on to say that inventory records should be 95% accurate, the BOM should be 98 to 99% accurate, and the routings should be 95 to 98% accurate [WIGHT 84].

Prior to the SFCS, each of the WVA databases identified by Wight has been an independent database utilized by unique organizations for their own requirements. Indeed, it could be argued that the data met the data accuracy requirements for the environment they were used in. For example, the Customer Order File describes the orders from the customers and is a tool for Production Control personnel to coordinate customer requirements with shop capacity and progress. Currently, information required by other organizations

is exchanged by using a written document or through phone conversations. Another example, which will be the primary focus of this paper's discussion involves the Master Routing File (MRF). The MRF is an automated record of the component routings (divided by each operation step) used to describe the manufacturing process to produce a part (Table 1). As demonstrated in Table 1, the MRF is strictly a identifier of the tools, machines, costing information, and process times to manufacture a component. The complete narrative process plan is maintained in the Computer Aided Process Planning (CAPP) system. A computer program transfers the MRF specific information to the MRF (IBM resident) from CAPP (DEC resident) each evening by an electronic batch process. In the current environment, production planners are assigned specific parts to monitor and are required to maintain the accurate process plans and MRF unique information required to manufacture these items.

Table 1. Sample Master Routing File.

ROUTING OPERATIONS

```
TASK: VIEW DWG:  15647813  MOD: 00 OPER# 0010  RTG DATE: 02/25/90
OPER DESC:  SHEAR           PLN: DMR  WK CTR: 22 WK CELL:  02 LOT/COMP:  C
PROC HR: 0000  1ST PC/PATL  HR: 0025 IN PROC INSP HR: 0000 FINL INSP HR:00
     MISS/SUPP:                          ACTIVE/INACTIVE        A
     COST AREA:         1074             MACHINE GROUP:         0
     N/C CODE:                           UNIT STD HOURS:    0.255
     TIME STUDIED:         T             SET UP HOURS:         12
     IF NEC % FACTOR:    .00             TEAM OPER NO:         01
     BENCH CODE:                         LABOR CATEGORY:
     WV-1:      0    WV-2: 0             WV-3:    0   WV-4:      0
     WV-5:      0    WV-6: 0             WV-7     0   WV-8:      0

ALT/IF NEC OPER:   BASIC:      0            PERCENT:      0
                   BASIC:      0            PERCENT:      0
                   BASIC:      0            PERCENT:      0
                   BASIC:      0            PERCENT:      0

ALT AUTH  QTY:           ALT EXPIRE DATE:        CHG OVER SETUP HR:  000
IF NEC AUTH QTY:         IF NEC EXP DATE:        CHANGE OVER %:      040
EXT ALLOW QTY:           EXT ALLOW EXPD:         RTG REVISION:
MOVE TO:          0      QUEUE TIME:      0      TEARDOWN TIME HRS:  000

     MOVE QTY:    0      ORIG STD HOURS:         0.000
     ACTIVITY DATE: 03/03/90   EXTRA ALLOW HRS:  0.000

TOOL-INFO-SFC-UPDATE

DRAWING NUMBER:  15647813     MOD:  00   OPERATION NUMBER:  0010
TOOL-ID   GAGE #     TOOL DWG  #     T#     FED STOCK  #  WASTC KIT QTYPE
01 00000    00000000  15647813     T004                          000   01
DESC: GAGE, LENGTH (366.6mm)                    REQTY:  0.000  CRITICL:
02.0000     00000000  15647813     T003                          000   01
DESC: GAGE, LENGTH  (1015.0MM)                  REQTY:  0.000  CRITICL:
```

If there is a mistake or immediate change identified involving the manufacturing process, it may be resolved by a phone call without timely update to CAPP, the MRF, or a hard copy routing on the shop floor. In reality, the most accurate routing is the hard copy in the production planner's desk. Since the telephone is eliminated as a primary planning tool in the SFC scheduling environment, the routing must be accurately reflected within the MRF. The coordination described is indicative of many of the established databases at WVA. In the SFC environment, these systems are no longer manually interfaced, but are integrated into a single system database.

As can be seen, highly accurate databases easily integrated into a single system was not our starting environment. The remainder of this paper will discuss how WVA has reacted to meet the quality requirements needed for an effective Shop Floor Control system.

2. Data Analysis.

The focus of the discussion will center on the Master Routing File (MRF) database. As was previously described, the MRF is extremely complex. Initially, it included information such as drawing number, parts description, steps required to create a part, the time to machine a part, set up hours, the machines to be utilized, in which area the part was to be manufactured, etc. In addition to this information, the implementation of the SFCS required many additional fields of information to be added such as tooling/gaging identification, estimated process times, Quality Control inspection times, work cell/ work centers information, consumption rates, etc. With the exception of the tooling/gaging information, all other additional data had to be developed and manually inputted into the MRF database. Because the entire routing was also kept on file on a CPT word processor, the tooling/gaging information was transferred from this file to the newly created fields in the MRF.

The database, as it initially existed, was estimated to be 90% accurate for the information it maintained. It was not until SFC required the same data fields in the multiple databases that the accuracy problems began. Approximately 70% of the tooling data transferred to the MRF had to be redefined and standardized. This was attempted through discussions, monitoring end-users creation of data, and the development of a data-dictionary for the respective databases utilized by Shop Floor Control.

To ensure data quality, two approaches were utilized. First, the accuracy of the data was reviewed to compare the information within the MRF routing to the actual operations on the shop floor. This quality was verified by personnel conducting walk-throughs on the floor to validate if what was reflected on the routing was actually done by the machinist on the factory floor. Although the most complete processing document available to shop personnel, it tended to be inaccurate and incomplete with processes being done differently from what was described on the hard copy routing. This included the utilization of different machines, time standards changing but not reflected on the routing, and tooling being inaccurate or incomplete. Additionally, there were some instances of entire operations being deleted or added to the process, but not yet reflected on the routing. Note however, what was being done on the shop floor was in most instances the actual process to be utilized as discussed with the planner of record and as generally described in the MRF. However, tooling and gaging information was very inconsistent and eventually became one of our main focal points for data review.

The second approach utilized by WVA to develop consistent information

was to concentrate on the quality of data as it applies to data field formats. Again, the same fields in multiple databases were defined in different ways as individuals used personal preferences when developing data. For example, consistent tooling information is essential to SFC for resource checking. The descriptions in the MRF must be the same as those in the Tool & Gage system so tooling requirements and availability can be checked. If the tooling is not described consistently, numerous artificial shortfalls could be identified because the tools were not numbered identically between the two systems. A tool drawing number and "T" number combination uniquely identify a unique tool for a specific operation. Only common tooling has generic numbers which could apply to multiple operations. Some planners would identify the same tool within an operation by a different function code (the function code being a brief/consolidated description of the item - one planner's "washer" with a function code 227 may have been another planner's "spacer" with a function code of 200) because of the different function codes assigned, the item was given a different "T" number which identified the item as being two different tools. Also, in the MRF the same tool was identified two or more times with the only difference being the drawing size identified at the end of the "T" number. The drawing size meant nothing to the planner of record or the machinist on the floor other than to alert him/her that a drawing actually existed. However, within Supply Division, the data entry clerk was not technically qualified to make this determination. Therefore, the same tool was manually input into the supply inventories two or more times with different on hand inventory quantities for each item. I.E. 12528311 T001A versus 12528311 T001B. Another data problem was the numerous occurrences when the planner of record assigned a "T" number designation to a tool when in reality the tool was a standard off-the-shelf purchased item. This was done so he/she could control the on-hand quantities for his/her particular job. The outcome was unnecessary duplicate data maintenance and inaccurate on-hand balances. Finally, one of the most time consuming corrections to be made was actual data input. In one system, a tool was identified as 12528311 T2 and in another it was identified as 12528311 T02 or T002 or T002A. Obviously, as this information was to be utilized in integration with the rest of the SFCS, this practice had to be discontinued. This meant developing consistent data definitions and formats across individual lines formats across individual lines within the organizations. The development of a lines within the organizations. The development of a data dictionary became the baseline for all changes and modifications to any database within the SFCS. Through meetings and reviews with arsenal computer programmers and associated system end-users, approximately 11 databases were defined and documented within a one week period.

Once the databases and their associated data fields were documented for format and consistency, the next effort was to verify the format quality of the arsenal data. All reviews have been and are limited to active operations for work currently scheduled in the arsenal. An example of this process was the requirement to identify work cells describing machines capable of accomplishing a certain machine process. This is critical in assigning work to the correct machines utilizing the Shop Floor Control scheduling functionality (Table 2). During the first manual review of over 12,000 active operations, the MRF routing operations were found to only have about 10% of the work cell/ work centers properly identified. The results of this review were sent to the end-user for correction. This manual review continued within the office, not only for work cells, but for approximately twenty other data fields. This effort required continuous extensive reviews utilizing at least two staff members full-time.

Table 2. Work Cell/ Work Centers/ Machine Relationship.

COST AREA: 1050

MINOR COMPONENTS

WORK CENTER: 12

SHAPERS

WORK CELL:	01	02	03
	P&W VERT MECH W/ DIVIDING HEAD	P&W VERT MECH	
WORK STATION:			
	WV11295 25-2-M-0	WV02885 25-2-M-0 WV02826 25-2-M-1 WV11879 25-2-M-1 WV03474 25-2-G-6	

Due to the need to continue the reviews, the next challenge was to continue this analysis while minimizing the manual review time. This led to the development of programs which allow continuous routine checks for consistency of data. First, for the end-user, data has been developed indicating the overall correct data and the amount of data that is in the database. This is also utilized to discuss data quality with management as numerous data quality parameters can be described in a synopsis format (Table 3). Most important was the development of a data exception report listing the lines of data which are incorrect (Table 4). As can be seen, the end-user has a consolidated list of work centers/cells which have not been identified. This document then provides an immediate feedback to the end-user to identify and correct specific data fields.

Table 3. Data Quality Management Synopsis.

WORK CELL/WORK CENTERS NOT ENTERED
(SFC0000111

WEEK ENDING	TOT OPNS CHECKED	TOT EXCEPTIONS	% COMPLETE	WEEK ENDING	TOT OPNS CHECKED	TOT EXCEPTIONS	% COMPLETE
27 OCT 89	2464	2178	12	12 JAN 90	2528	1952	23
03 NOV 89	2469	2178	12	19 JAN 90	2571	1995	22
10 NOV 89	2474	1958	21	26 JAN, 2&9 FEB 90 NOT RUN DUE TO MRF ERROR			
17 NOV 89	2475	1958	21	16 FEB 90	2794	2210	21
01 DEC 89	2493	1956	22	23 FEB 90	2819	2238	21
08 DEC 89	2499	1956	22	02 MAR 90	2833	1451	49
15 DEC 89	2508	1955	22	09 MAR 90	2886	1029	64
22 & 29 DEC 89	NOT RUN DUE TO HOLIDAY						
05 JAN 90	2515	1928	23				

Table 4. Data Quality Exception Report.

SFC00001 13:41 ACTIVE MASTER ROUTING OPERATIONS REQUIRING IDENTIFICATION COST AREA = 1050

WK-CTR	WK-CELL	DRAWING NUMBER	MOD	OPER	DESCRIPTION	WV#	COST AREA	STATUS CODE
00	00	WTV-C33853	00	0020	MILL		1050	A
00	00	WTV-C33853	00	0050	GRIND		1050	A
00	00	WTV-C34808	00	0030	LATHE		1050	A
00	00	WTV-C34808	00	0040	MILL		1050	A
00	00	M119NM34974	00	0085	GRIND TO ALIGN		1050	A
00	00	M119NM34990	00	0108	BENCH		1050	A
00	00	M119NM35002	00	0044	MILL		1050	A
00	00	M119NM35002	00	0048	MILL		1050	A
00	00	M119NM35002	00	0048	STRAIGHTEN		1050	A
00	00	M119NM35014	00	0004	GRIND		1050	A
00	00	M119NM35038	A1	0010	MILL		1050	A
00	00	M119NM35038	A1	0020	MILL		1050	A
00	00	M119NM35038	A1	0030	MILL/NC		1050	A
00	00	M119NM35038	A1	0040	MILL/NC		1050	A
00	00	M119NM35038	A1	0080	GRIND		1050	A
00	00	M119NM35038	A1	0090	DRILL		1050	A
00	00	M119NM35038	A1	0100	MILL/NC		1050	A
00	00	M119NM35038	A1	0110	MILL		1050	A
00	00	M119NM35038	A1	0120	MILL/NC		1050	A
00	00	M119NM35038	A1	0130	GRIND		1050	A
00	00	M119NM35038	A1	0140	GRIND		1050	A
00	00	M119NM35038	A1	0150	MILL/NC		1050	A

These programs are proving to be the key to providing continuous data quality feed-back to the end-users and management alike. The programs and reports have reduced the overall review cycle time to less than one hour a week by one person on the SFC staff. The only required task is to annotate the quality percentages in the management overview document and send the packet to the respective end-user capable of making the corrections.

3. Discussion of Results.

What has been described is the efforts of the arsenal over the last 17 months. Currently, the accuracy of many data elements has improved. Audits on the floor show minor discrepancies of some common tools, minor time changes, and little deviation of appropriate machine requirements. This is a significant improvement over the earlier audits. As was discussed, the initial audits reflected major discrepancies such as incorrect operations, machines, and tooling.

3.1. Minor Components Branch Data.

The following figures describe numerous areas that are being monitored and how the accuracy has improved in the MRF routings for Minor Components Branch (Figures 2a-2c). The legend on the left side of these charts quantify the total number of work cell/ work centers, tooling, and inspection times reviewed for accuracy. The right legend describes the percentage of data fields which are correctly formatted. As Minors was the initial area for implementation, the data correction and the development of monitoring programs were created and changed concurrently over the initial phases of the program. The start point of this data in the figures was well into the correction process. As such, the corrections and accuracy show minor changes from the start to end of the time line. It can be seen in the latter dates, that some of the data started to degrade. This was due to the extensive activation of old routings which had not been corrected earlier. It is important to note that these trends were quickly and routinely detected using the exception reports. As such, the tools were in place to continue the corrections and make the required changes.

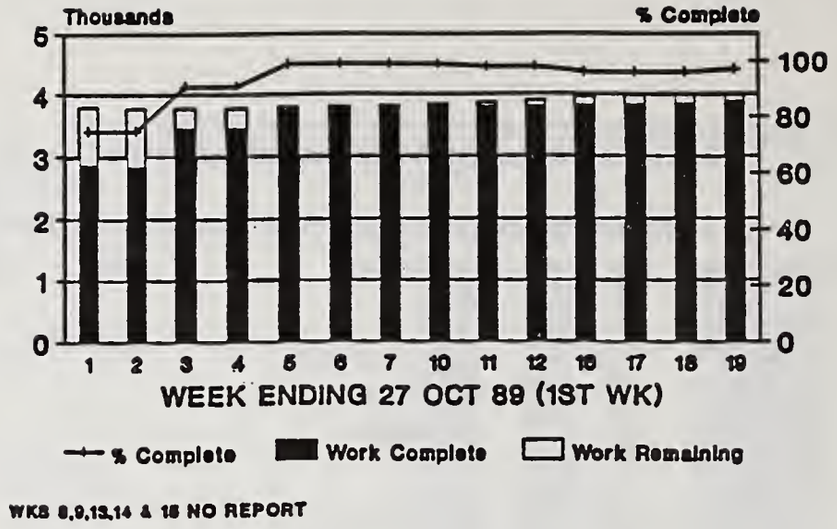


Figure 2a. Minor Component Branch Work Cell/ Work Centers.

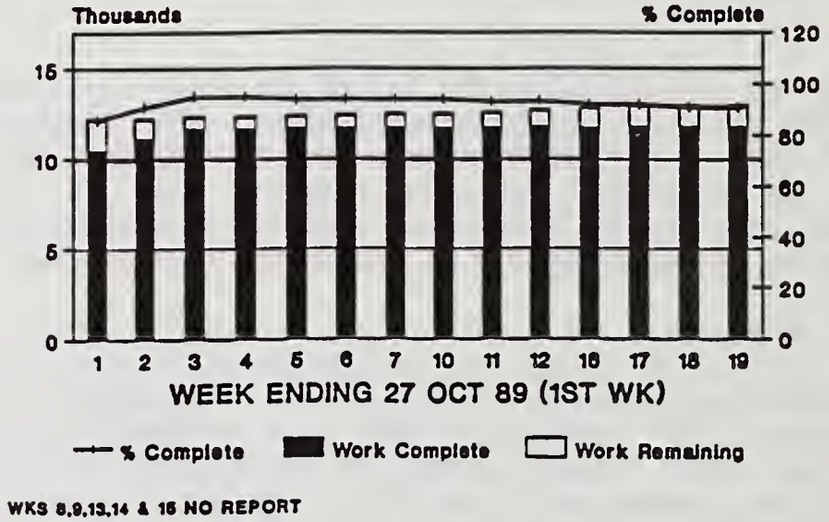


Figure 2b. Minor Component Branch Tooling.

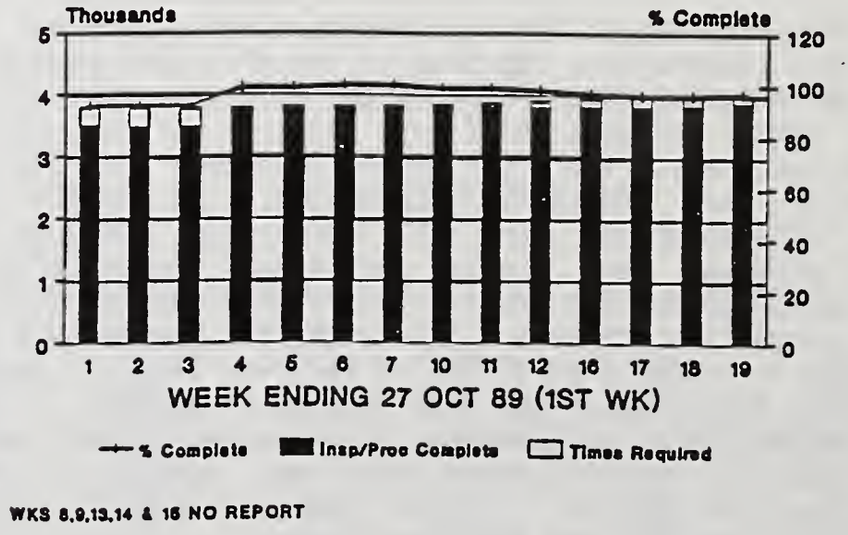


Figure 2c. Minor Component Branch Inspection Times.

3.2. All Other Operation Areas.

The comparison of the other production areas compared to Minor Components Branch enforces the importance of developing a system to monitor data quality. As previously mentioned, most of the data in the other arsenal areas was not worked on until most of the minor components were complete. The exception was in the tool/gage and inspection data. This was due to the amount of data that needed to be updated or created. However, rather than the 10 - 17 months required for the correction and updating of the processes describing the manufacturing of minor components, it was possible to make the corrections in weeks. This is most obvious in the work cell/ work centers update (Figure 3a) demonstrating an increase from 20% to 65% of the total data corrected and completed in two weeks. As can be seen, the tooling identification and inspection times have not made the same degree of improvements (Figures 3b-3c). With the same emphasis used to correct the work cell/ work centers data, these areas have the same systems in place to allow rapid improvement.

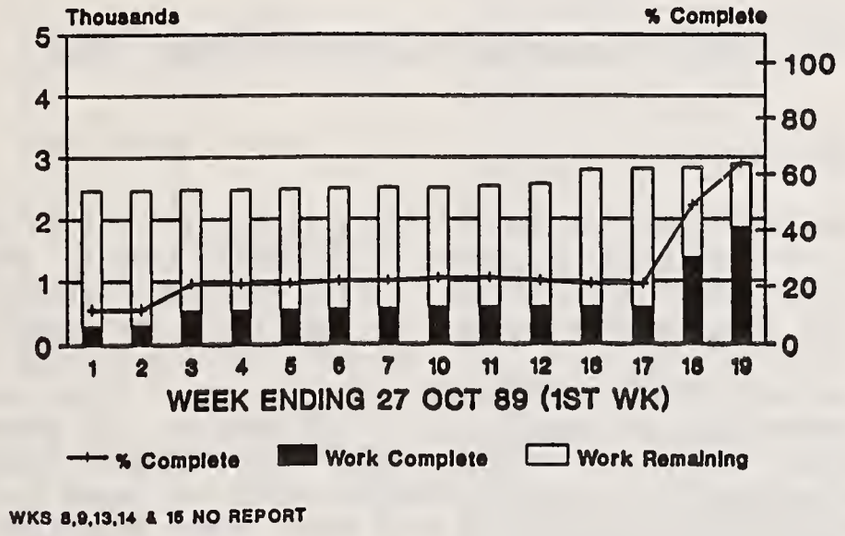


Figure 3a. Work Cell/ Work Centers in All Other Areas.

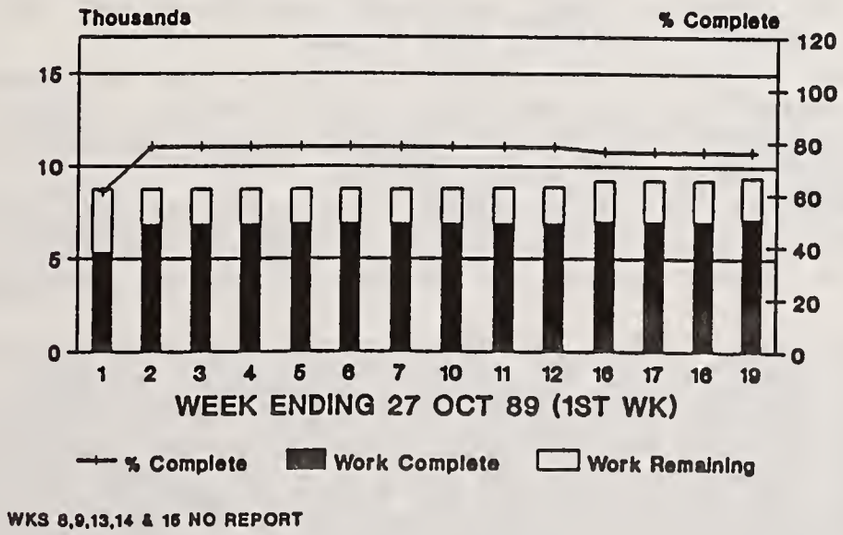


Figure 3b. Tooling in All Other Areas.

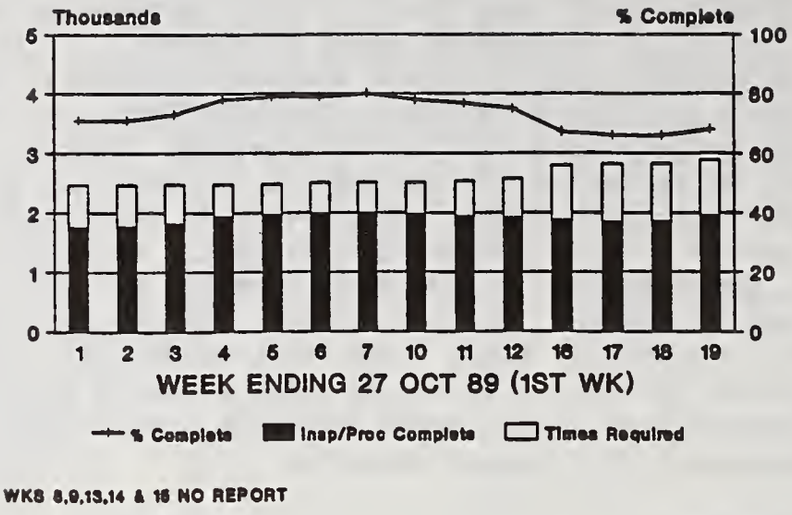


Figure 3c. Inspection Times in All Other Areas.

3.4 Ongoing Challenges.

Other areas continue to be difficult to correct. Tool serial numbers and their match over multiple databases is extremely hard to rectify. This is due to more "owners and creators" owning different aspects of the data, combined with the great numbers of tooling involved. For example, Planning, Supply, and Product Assurance all describe tooling or gages. Each has a different "slice" of the pie and may describe the same item with a slight variance. Compounding the complexity of the problem, the arsenal maintains a tooling inventory of over 51,500 line items. These lines equate to maintaining accountability of over 600,000 tools and gages in the shops and warehouses. Methodologies, similar to what have been described, are being utilized to correct these deficiencies.

The key is to continue the emphasis in developing and maintaining "good" data. This includes letting all personnel know it is a management concern through the use of organizational and individual "performance" standards and discussions. Additionally, the continued education and assistance in establishing parameters to evaluate data is key.

All arsenal procedures are currently under review to ensure data entry procedures and parameters are described consistently. Additionally, the procedures are accounting for the new computer systems and their influence on the function within the arsenal organizations.

Finally, emphasis is being placed in reviewing the many multiple databases to determine whether many could be combined and others eliminated. Machine data is a classic example and demonstrated by the following diagram (Figure 4).

As can be seen, there are multiple possible data entry points using either the SFCS, CAPP, or the Maintenance Management system. During the initial phases of the development of the SFCS, while establishing a data dictionary, initial plans had been made to determine the hierarchy of data entry and flow. Current reviews are being done to confirm the "entry" point for new data. Once verified, programming will be finalized to ensure data is transferred electronically to the other databases. This technique has already been used to develop data entry points for the Customer Order File, the Bill of Materials, and the MRF.

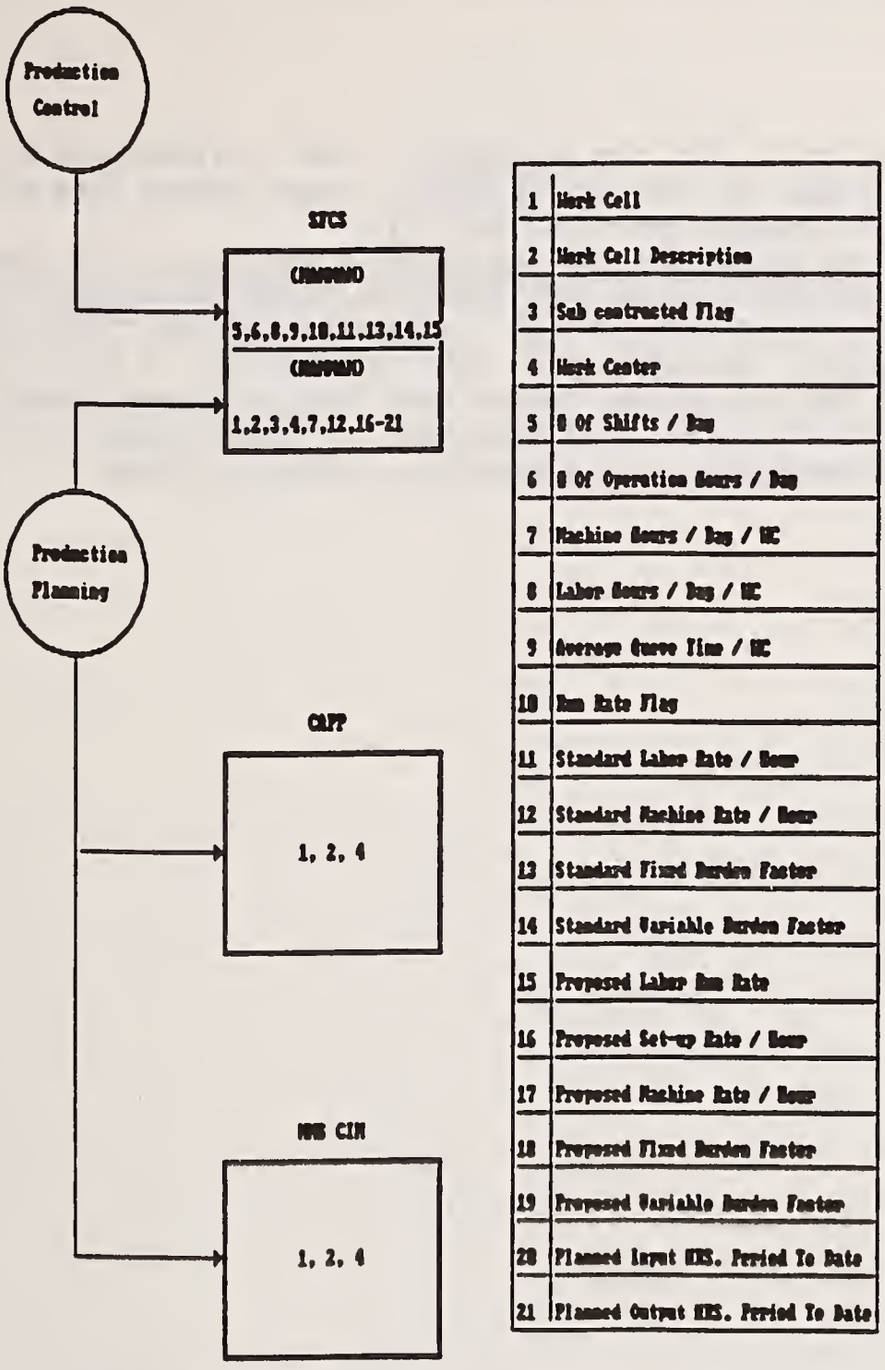


Figure 4. Machine Information Data Flow.

The primary disadvantage of establishing a single data entry point has been a one day delay in the batch transfer of some information to the other databases. However, this is a minor compromise in order to ensure all redundant databases have the same information.

4. Conclusion.

This process of correcting data and improving data quality has been on-going at WVA for over a year and half. The attention to detail and the emphasis on data quality, since the development of the data quality reports, have increased dramatically. This has also resulted in a great deal of improvement in our present systems as evidenced by the data improvement described in this paper. Additionally, every organization has been forced to review its present procedures and the procedures to be used in the future to conduct business. The end result of using the steps the arsenal has taken to

improve data quality is that the majority of the implementation staff has been able to focus on the SFC technical issues rather than being totally absorbed by the dilemma of poor data quality.

Finally, these techniques are important to create in the beginning of the project to promote routine continuous data quality and consistency. For any organization transitioning to an integrated system, this methodology should prove helpful. By doing it as soon as possible, it allows the project team to focus on the new system, rather than devoting large amounts of time to reviewing existing data. Most important, it greatly improves the chances of successfully developing and implementing a working system.

REFERENCE

Wight, Oliver, Manufacturing Resource Planning: MRP II, The Book Press, Brattleboro, Vermont, 1984.

John E. Ettlie
University of Michigan

ORGANIZING FOR INTEGRATED MANUFACTURING

Manufacturing is in transition. New technologies and new managerial styles are storming assumptions thought scared ten years ago. This paper is a summary of the findings of a four-year study of over three dozen cases of plant modernization. All but one of these plants were in the U.S. or Canada; and all of these cases represented significant attempts by these discrete part manufacturing firms to upgrade and integrate flexible automation into workflows. The purpose of the project was to determine what, if any, implementation strategies were most successful in making the transition from old to new manufacturing technology in automotive, equipment, appliance, aerospace, and general manufacturing industries.

We found a consistent pattern in not only the chronic problems that modernizing plants encounter, but also in the solution approaches these plants and firms have tried. Successful firms try to match their requisite philosophical shift with the type of technology adopted. The more radical the technology change, the more radical the new strategies and structures needed to be successful. When one realizes this, one begins to see the necessity of concentrating on the *administrative* innovations as well as the new technologies embodied in these manufacturing systems. The technology management perspective has much to offer in this vein.

CHRONIC PROBLEMS IN MANUFACTURING MODERNIZATION

Year in and year out, when we visited plants modernizing their facilities with integrated, flexible manufacturing systems of forming and assembly, the number one problem was always the same: *software and programming*. Both the development and maintenance of manufacturing software became such an issue that in our last visit to these plants in 1987 we included an entire section of or interview schedule to this problem to attempt to get insights into the nature of these issues. The results of the analyses were published recently in Ettlie and Getner (1989).

We found that in these modernizing plants and offices, people reported being most satisfied with manufacturing software maintenance when they delayed significant development tasks into the maintenance cycle of the project. That is, these installation sites were typically learning their

requirements as they went along. We believe that the 200 odd number of CASE tools now available to address this problem have not addressed the fundamental issue of factory modernization: people do not know what their general requirements are because they have not translated business plans into manufacturing strategies. Tools to help customers learn their requirements faster so they can effectively use new hardware and software systems in manufacturing are desperately needed.

ORGANIZATIONAL INTEGRATION

In order to be both more innovative and flexible but still meet orders, manufacturing units have to take new structural approaches to integrating functions of the firm. Lawrence and Dyer discuss this issue at length in their book Renewing American Industry (1983) before it became fashionable to think about radical shift in managing domestic firms. What we did was to take the approach that most manufacturing firms are at a distinct disadvantage when they modernize their facilities to become more flexible and integrated. They buy most of the technology from suppliers, they simplify most of their products so they can be more easily assembled--but can be copied, and they often are playing a catch-up game which requires down-sizing and organizational turmoil in the resulting structure. How does one capture value from innovating under these circumstances?

We proposed three major hypotheses to capture this process of new technology utilization in manufacturing:

1. Manufacturing firms change technology and structures simultaneously in order to be successful. They become more integrated in their hierarchy, in the design-manufacturing relationships, with customers and with suppliers. (See Appendix for summary of the measures of these structuring mechanisms).

2. Performance outcomes tend to cluster in two-tiers in manufacturing firms when process innovation is the focus: a system-level or department level of measurement (e.g., utilization), and an adopting unit performance level (e.g., return-on investment).

3. Integrating mechanisms that flatten or allow power sharing in hierarchies, and structures that help coordinate design-manufacturing transactions, tend to impact system-or department-level performance outcomes, while extraorganizational integration with suppliers and customers tends to impact the second-tier of performance measures for the adopting unit.

These three hypotheses are generally supported by results with some notable additions. For example, the details of restructuring for design-manufacturing integration can be used to predict ROI, although our valid data on this measure was limited to 14 cases. That is, knowing some structures for internal coordination between functions can be used to predict adopting unit performance on these significant modernization programs.

Other notable and statistically significant results are as follows:

1. Hierarchical integration (e.g., technology agreement with union) tends to result in greater throughput time reductions after modernization.

2. Design-Manufacturing integration (e.g., rotation of engineers between design and manufacturing engineering) tends to increase system utilization (2-shift basis).[1]

3. Supplier integration (e.g., JIT purchasing) generally promoted significant reductions in scrap and rework percentages as a percent of total production costs (average = 3% for entire sample).

4. Customer integration (e.g., JIT delivery) promoted faster changeover times.

A pattern that emerges here which reinforces much of the earlier thinking on manufacturing management is that the best plants and business units leverage their suppliers to satisfy customers. *What is more, internal integrating mechanisms tend to promote improvement on capacity or work flow related parameters like throughput, cycle time and utilization, whereas extraorganizational integrating mechanisms tend to promote quality and flexibility improvement. These results support*

---

[1]We now have historical data points from three epocs (1969, 1973 and 1987) which strongly suggests a linear improvement in 2-shift utilization of 1% per year over this approximate 20 year period, averaging about 72% in the most recent epoc.

the emergent general axiom of the late 1980's in manufacturing: you have to be good at many things to survive and prosper in global competition.[2]

WHY ARE SOME MANUFACTURING FIRMS MORE INNOVATIVE?

When one studies the innovation process in manufacturing, there is the tendency to equate novelty with success. This is not always the case. We are aware of firms that are well known for their significant new product winners but they have a poor track record for deployment of new processing technologies. What is more, as one adopts the technology management perspective, one begins to see the facilies of "wisdom" often advanced in the popular press in the area. For example, if "flexible" or "reprogrammable" technology is not used in a flexible way or is not reprogrammed, it is assumed to be a danger signal. Not necessarily so. If technology becomes obsolete before it can be reprogrammed, or if it pays for itself before it must be replaced, it may have bee vital to the unit's survival. Some technologies waste materials when used most flexibly. What is more, new technologies such as net shape may have much greater impact on a firm's profitability and survival and flexibility. There are many new and important technologies in manufacturing in addition to flexibility and integration.

What accounts for a firm's willingness to experiment with both new technology and new management approaches--regardless of their outcome? We have found there are two consistent factors that are the well-spring of innovativeness in manufacturing: the *general management experience profile* and the *manufacturing technology policy* of the unit.

General Management Experience Profile

We have followed up the proposals of Hayes and Abernathy (1980) who said our demise in domestic firms had resulted from domination among CEOs and general managers by the legal

---

[2]Overall, modernization cases averaged about a 30% improvement in the cost of quality on the scrap and rework measure over existing plant production.

and financial professions. Innovative manufacturing firms, it was thought, resulted from general managers that came up through the technical and manufacturing ranks.

We tested this notion and found the theory supported with some twists. It does seem to be true that when the CEO has manufacturing experience, the firm is much more likely to have an aggressive manufacturing technology policy (see below), as well as adopt new equipment that incorporates radial technology. However, these same firms are also more likely to emphasize labor savings from modernization--which may not be the most important outcome of value capture of this generation of flexible, integrated manufacturing. Lower level managers with manufacturing experience tend to deemphasize labor savings. Commitment to training for modernization extends well into the top management suite, much higher than we expected, to the senior V.P. level. When divisional managers have manufacturing experience, the unit is much more likely to use an administrative experiment for modernization and is also more likely to achieve higher utilization with the new hardware and software. Clearly, managerial experience profiles are essential to understand if one wants to understand of why some manufacturing firms are more innovative.

Manufacturing Technology Policy

Over the past ten years we devoted considerable effort to document and evaluate the causes and impacts of innovative technology policies. Starting with our studies of mixed samples of service and nonservice cases (Ettlie and Bridges, 1982) and then moving on to work on the food industry and their suppliers (Ettlie, 1983; Ettlie, et al., 1984), and finally, continuing on to this project on integrated flexible manufacturing deployment (Ettlie, 1988), we have studied more than 300 firms and business units. We have found that the manufacturing technology policy of a firm to be a good predictor of new product and process deployment as well as a reliable way of implementing business strategies driven by changing environments.

An aggressive manufacturing technology policy is typified by a reputation in an industry for being the first to consistently try out new methods, equipment and ideas. It is also typical in companies that have an ongoing program to recruit the best qualified technical and manufacturing

talent available. These firms are strongly committed to technological forecasting and advertise their new processing technology to their customers. (Although we observed a recent tendency towards more secrecy).

An aggressive technology policy (in addition to being the result of a CEO with manufacturing experience, see above) tends to be significantly associated with the following in our study of plant modernization:

1. These manufacturing units are generally more likely to use some type of administrative experiment for modernization;

2. These manufacturing plants that are being modernized are more likely to be strategically focused than their less aggressive counterparts; and

3. These plants are more likely to have more flexible manufacturing systems (in either assembly or forming) based on the variety of part families they schedule on these new systems (as opposed to their change-over time) and to a lesser extent in the number of part numbers scheduled.

4. We have found a moderate but consistent trend over three years of data from modernizing plants for an aggressive manufacturing technology policy to be related to innovative changes in organizational structure and changes in policies for dealing with customers--precisely where policies would be expected to impact. We found no relationship in these three-year data with design-manufacturing integration or supplier integration.

It appears that manufacturing technology policy, and perhaps other measures of innovation strategy in general, will continue to be an important determinant of innovativeness and other outcomes in modernization.

THE RECONCILIATION OF MIS AND MANUFACTURING FOR INTEGRATED MANUFACTURING

BARBARA M. FOSSUM
JOHN E. ETTLIE

Computer integrated manufacturing (CIM) systems are complex. They require the simultaneous support of technical staffs in management information systems (MIS) and manufacturing. Just when these functions should be working together very closely, they are frequently in conflict. Using cases of manufacturers who have implemented computer systems for integration and control of factory operations, or "paperless environments," the authors examine this relationship of MIS and manufacturing. These cases suggest propositions for testing in empirical research and actions for breaking down chronic barriers between MIS and manufacturing.

1. Introduction to Paper

Moving toward an integrated enterprise, and simultaneously toward a computer integrated manufacturing (CIM) system which is compatible with this integrated enterprise, has become a major category of strategic response to competition and to customer needs in many industries. Progress toward CIM is slow, however, and many reasons are cited. Among these, Fossum [1986] suggests that the complexity of CIM technology is one of several significant barriers to CIM progress [1]. Related to the complexity of the technology is the need for in-house technical expertise. Manufacturing managers and executives who responded to an Industry Week survey ranked lack of in-house technological expertise as the number one obstacle to CIM progress [SHE, 1989] [2]. The manufacturers involved in the Fossum study [1986] also cited a strong technical staff as a significant factor to integration progress [3].

Although substantial technical skills often can be found in the computer aided design (CAD) and computer aided manufacturing (CAM) application areas of organizations, the concentration of information systems technology expertise still resides primarily in the MIS function, whether this function is centralized or decentralized. There is a critical need for a strong technical staff to support implementations of complex CIM technology. However, MIS and manufacturing, functions which should be working together very closely, are in frequent conflict with one another in many organizations. Worse, as manufacturers make progress toward CIM, the conflict can be expected to increase, as shown in Figure 1, due to additional transitional conflict that results from change.

The MIS-Manufacturing relationship varies greatly across organizations. At one extreme, MIS is perceived to have a bias toward particular mainframe vendors, often disguised as pursuit of standardization, and a bias toward particular methods of system implementation. At the other extreme, MIS is perceived not to care, as they abandon users with computer-aided software engineering (CASE) tools or they steer away from decisions that concern microprocessor-based factory equipment. Battlelines often form around the timely delivery of effective manufacturing software and subsequent maintenance of new systems.

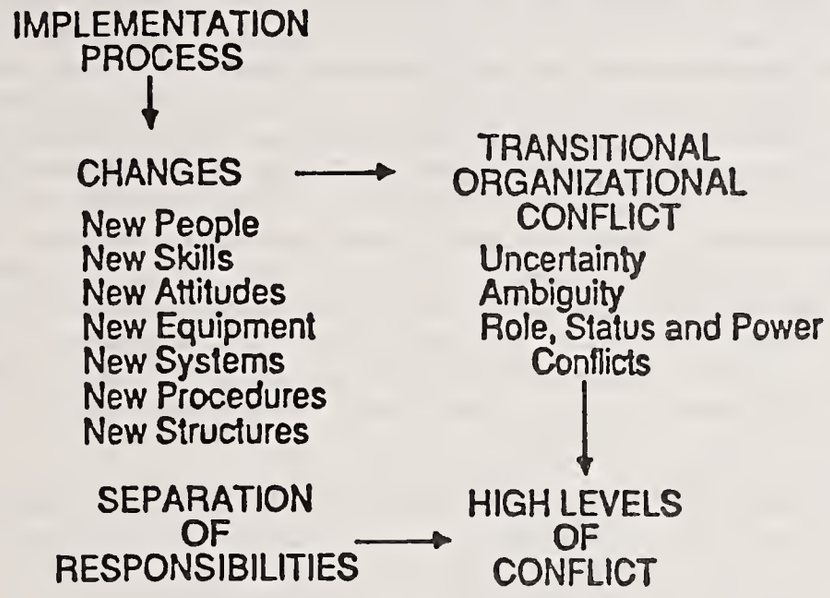


Figure 1. Intraorganizational Conflict [FOS, 1986].

Even when an outside supplier is involved, some function usually has to take ultimate responsibility for the new technology. Disputes arise over this responsibility. Resolution of conflicts between the functions often end up in the hands of general managers who are not adequately prepared to deal with the issues. In this paper, we suggest that there is an alternative to the conflict scenario which is repeated frequently throughout modernizing manufacturing enterprises. We examine the MIS-Manufacturing relationship in the context of the experiences of manufacturers who have implemented, or are in the process of implementing, factory management and control systems. With these systems, the organizations are addressing problems such as limited control of the process, lack of visibility of work status and location, no access to yield and cause of defect information, limited flexibility to respond quickly to change, no traceability of products, large work queues, excess and stalled inventory, rework, excessive indirect labor, and too much paper.

Some of the manufacturers described in this paper developed their own factory management and control systems. Others acquired packages or "starter solutions" and modified or enhanced these systems. Because control systems must be integrated with almost all other information systems, many of which are maintained by MIS staffs, and with systems of automation in an enterprise, the implementation of these control systems provides a good touch stone for the study and subsequent suggestion of methods of improvement of the MIS-Manufacturing relationship. We draw on these cases, the literature and our experience in modernizing plants to suggest propositions for empirical testing and actions to help reconcile MIS-Manufacturing differences.

2. Factory Management and Control Systems

As shown in Figure 2, a factory management and control system bridges the gap between systems in a CIM framework [4]. In one direction, control systems communicate with manufacturing resource planning (MRP II) systems, general business systems like accounting and payroll, and engineering systems like CAD and computer aided process planning (CAPP). From these systems a control system receives such data as bills of resources, process flow, workloads, computer numerical control (CNC) programs, and operator work instructions, including part graphics. To these systems, the control system reports production status and resource utilization. In another direction, a control system communicates with devices and human operators on the factory floor and, in so doing, integrates these islands of execution. To these entities a control system passes instructions and other types of information. From these entities, a control system collects data, for example, labor data from a human operator or test measurements from a device. All processing and communication is in real time or near real time ("real soon") mode of data processing.

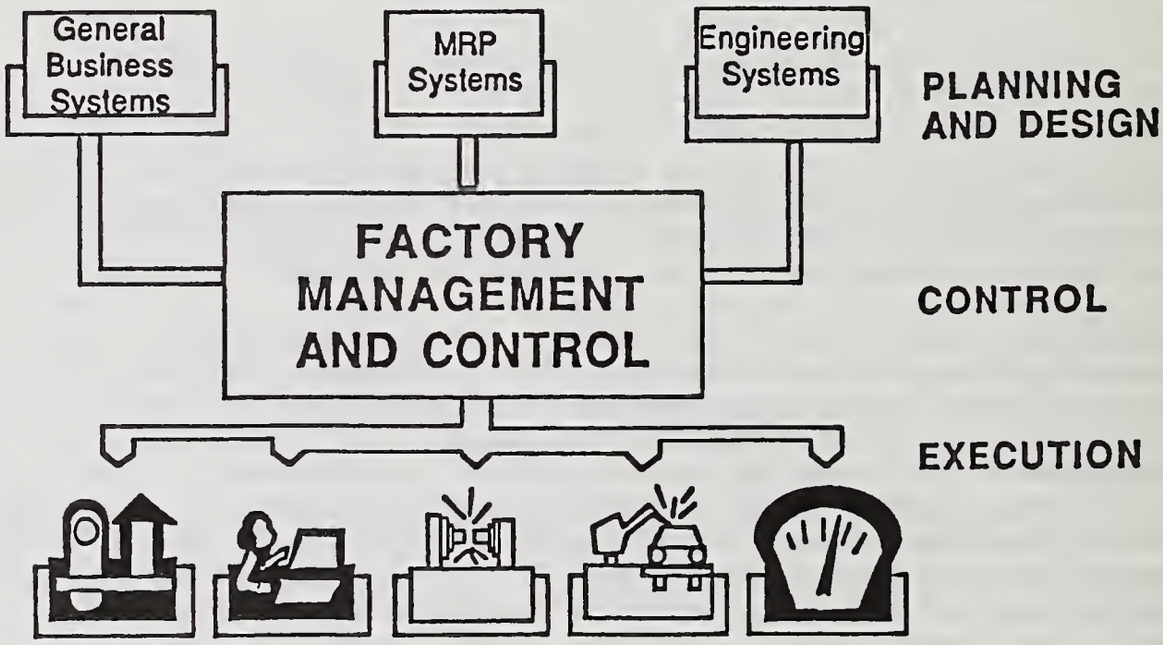


Figure 2. Bridging the Gap.

A factory management and control system is sometimes called a plant operations system, a plant control system, a factory system, or simply a "control" system. A primary function of a control system is to direct, manage and control all resources (people, processes, material, tools, machines, other equipment, work instructions, etc.) in a factory to ensure that all the necessary resources are brought to bear at the right time and at the right place to produce a product. Control systems are event driven and are responsive to contingencies. Although the many functions of a control system can be distributed over multiple heterogenous computer systems in a CIM

architecture, in each of the case studies the control system is operational on a single homogenous system in a hierarchical structure.

Implementation of a control system is an ongoing process. Products and processes change. Interfaces to new process and material handling equipment or new methods of collecting and reporting data are required on a continual basis. Because a control system automates the information flow, from receiving through shipping in a facility, the control system represents or models the process as events occur. Therefore, implementation of a control system requires involvement by those who have intimate knowledge of current methods of operation and clear vision of intended new methods.

3. Cases Descriptions

3.1. PGAS Inc.

PGAS Inc. [5] is a single plant division of a multi-plant, large [6] corporation. This is a case in which the MIS-Manufacturing relationship historically has been, and continues to be, good. MIS traditionally has not attempted to control the selection, implementation and use of computers by PGAS Inc. but has provided technical guidance during these activities. The manufacturing process at PGAS Inc. is mixed: the beginning of the process consists of many batch, job shop oriented metal fabrication activites; the remainder of the process consists of repetitive, unit assembly activities. Pay systems are both day rate and piece-rate incentive based.

PGAS Inc. acquired a factory management and control system and worked with the supplier of this system to modify and enhance the solution. A prototype (modeling) process was used to develop functional specifications for modification and enhancements to the software. Using PGAS' part types and process flows, the software supplier developed with PGAS Inc. a model of the material flow and the associated data flow using the standard or base software. Simultaneously, the software supplier trained PGAS Inc. on the use of the system and in the development of functional specifications for changes and enhancements to the system. Through an iterative modeling process, functional specifications were completed and the supplier of the software used these specifications to develop the modifications and enhancements to the system. During the process of functional specification development, corporate MIS individuals were involved to relate information requirements of the host mainframe based systems.

All general business functions (accounting, payroll, etc.) as well as some manufacturing planning functions are operational on a centralized mainframe shared by all divisions and are maintained by a corporate MIS staff. Engineering and plant operation functions, as well as most manufacturing planning functions, are decentralized and supported by computer systems in each of the factories. There is bi-directional, electronic communication between the corporate mainframe and the factory computer system in PGAS Inc. PGAS Inc. relied on the supplier of the factory management and control system and corporate MIS to create this interface. The factory system also communicates electronically with material storage carousels in the factory. PGAS Inc. relied on the supplier of the factory management and control system to create this interface. Future add-on projects include communication between

the factory control system and automated test equipment to be acquired by PGAS Inc.

The factory management and control system is part of an overall, continuous factory renovation program which focuses on competitive manufacturing. The control system projects, including integration with or interface to other systems, are led by an individual who is responsible for all factory automation projects which affect material control. These include automation of the processes or material handling as well as automation of information flow. This individual is a "user project manager" and reports to the vice president of operations for the division. (CAD/CAM projects also report to this executive.) Communication between the user project manager and the industrial engineering staff responsible for the selection of process equipment is very good.

Corporate MIS were involved in the selection of the control system only to the extent of ensuring that the control system could communicate with the corporate business computer systems. Corporate MIS personnel, not employees of the division, were assigned formally to the project and reported to the user project manager. In addition to corporate MIS personnel, the plant had its own analysts and programmers assigned to the project. Prior dissatisfaction with the level of knowledge of corporate MIS staff on manufacturing operations, and the high level of turnover of corporate MIS staff, led the division to build its own, albeit small, computer support staff. Although PGAS Inc. hired computer programmers, they also trained manufacturing operations personnel in computer systems analysis. They found this process easier than training computer personnel in operations. These individuals, the systems analysts and the computer programmers, continue to maintain the system with little or no involvement by corporate MIS. The control system has been operational for almost three years and the plant is in its fifth stage of implementation with add-on projects.

In summary, the factory management and control system implementation at PGAS Inc. is considered a success: the system is meeting, and in some cases exceeding, documented objectives related to factors of competitive manufacturing. Among many of the objectives met is inventory accuracy which currently stands in excess of 99.9 percent. End users at all levels are very happy with the system. Payback occurred in less than one year. The program is user-driven with a strong user manager, and continues to involve users at all levels in ongoing projects associated with the overall system. MIS personnel were involved directly and provided necessary integration and interface expertise. Application-oriented systems analysts and programmers who understand the manufacturing process maintain the system.

3.2 BAA Inc.

BAA Inc. [5] is a single plant subsidiary of a large corporation and represents a case in which the MIS-Manufacturing relationship historically has not been particularly good but currently is improving. MIS traditionally has controlled the selection, implementation and use of computers by BAA Inc. MIS personnel consistently have been criticized by manufacturing users for their lack of understanding of the production process. The MIS function is located physically some distance from the factory. MIS personnel had very

little visibility in the factory. Almost no attention had been devoted to meeting specific information requirements of the production process.

The manufacturing process at BAA Inc. is mixed: the beginning of the process consists of very large batch and process oriented activities; the remainder of the process consists of discrete, repetitive, unit filling and packaging activities. Although the manufacturing process incorporates a significant amount of capital equipment, the overall production cycle is labor intensive. Equipment does not exist for some of the processes; others can be performed more effectively by people.

BAA Inc. acquired a factory management and control system and is working with the supplier of this system to modify and enhance the solution. The factory management and control system is part of an overall operations improvement program focused on competitive factors such as lead time, flexibility and cost. The general business functions (accounting, payroll, etc.) for BAA Inc. are operational on a mainframe located in the corporate office where the MIS function is staffed. Engineering and manufacturing planning functions are supported by a factory computer system located at the plant. The factory management and control system is operational on yet another (and different vendor) computer system located at the plant. Bi-directional, electronic communication between each of the three computer systems, as well as between the factory management and control system and factory devices and equipment, is in the implementation process.

A pilot implementation is in progress and involves a team of manufacturing users, MIS, and the software supplier. The manufacturing process was prototyped using the software provided by the software supplier. The supplier worked with the implementation team to model the flow of work and the flow of information with the existing software. A moderate amount of required changes were identified and made, and the software was implemented in production mode in a major product line of the plant. After using the software for several weeks, the implementation team and the software supplier began developing specifications for modification and enhancement of the base software system. The process appears to be working well in that the (1) the pilot was migrated to several areas of the facility where it is in production use (depended on by workers and supervisors), (2) the users were able to use and understand the system prior to defining changes and additions, (3) the supplier had time to understand the customer's process and requirements, and (4) the requirements specification process, of an overall three-month duration, is complete. The requirement definition process used by BAA Inc. supports the finding of Ettlie and Getner [1989] where users prolonged the software development cycle into maintenance in order to compensate for lack of understanding of requirements. In the case of BAA Inc., however, plant personnel used the system, prior to addition and changes, to increase understanding of requirements.

For the first six months of the implementation of the factory management and control system, two individuals shared the role of implementation project manager: a user (the director of engineering, reporting to the plant manager) and the corporate MIS manager of factory systems (reporting to the corporate director of MIS). MIS actually drove the project; the role of the director of engineering was more one of managing a user liaison. However, the perceived shared responsibility caused numerous interpersonal communication

problems and slowed the project considerably. Subsequently, the project management structure changed such that a new MIS program manager (reporting to the corporate director of MIS) has overall responsibility for the delivery of working hardware and software but the director of manufacturing (reporting to the plant manager) has direct responsibility for implementation success. The new program manager resides at the factory most of the time. Beyond the functional specification process, most interpersonl communication occurs between the software and hardware suppliers and the MIS program manager rather than the manufacturing users. MIS personnel do not report directly to the project manager but rather to a technical manager. The technical manager has responsibility for supporting more than the factory management and control system project and as such can not devote full attention to MIS staff who are supporting the project. The reporting structure of the MIS personnel requires an additional level of interpersonal communication for project assignments and coordination and continues to cause some problems.

During the functional specification process, corporate MIS individuals were involved to relate information requirements of the host mainframe-based systems. For interfaces between the systems and between the factory management and control system and devices, corporate MIS is relying on third parties due to lack of the necessary in-house communications expertise. In general, MIS individuals have limited knowledge of the manufacturing process and are still low on the learning curve for the new computer system. Both of these factors have had an adverse effect on schedule commitments. Manufacturing itself has no computer systems analysts or programmers and no plans to staff such individuals. Although the supplier of the software currently has maintenance responsibility, plans call for the ongoing maintenance of the system by the MIS functional area rather than by the manufacturing functional area. Corporate MIS is considering locating the MIS support staff at the plant. This positive change is expected to occur.

In summary, the factory management and control system implementation at BAA Inc. is expected to be a success in terms of meeting the objectives established, although these objectives may have been met with less expense with a different approach to program or project management. Production employees are using and benefiting from the standard version of the software system provided by the software supplier. With the exception of the technical interfaces between hardware and software systems, users at all levels and in all functions have been involved in all aspects of the implementation, particularly in the functional specification and training processes.

3.3 Northrop Corp.

Northrop Corp. [7] implemented a $14 million system called the Integrated Management, Planning and Control for Assembly (IMPCA) system for final assembly processes of its Navy F/A-18. The IMPCA system uses 135 workstations in the final assembly area where 1000 employees are involved in the process. Also connected to the system are users in manufacturing production, manufacturing engineering, industrial engineering, materials review, liaison engineering, quality assurance and quality planning. With the exception of the printing of some engineering changes, the print for which is discarded immediately after use, the system has eliminated paper -- about 400,000 pieces per day.

The real problem attacked by the IMPCA system solution is a function both of the product itself and the way in which the final assembly process traditionally has been controlled in the aircraft industry. The long (one mile) and narrow (20 yard) final assembly area creates unique communication and coordination problems. Additionally, both customer and engineering changes result in production lot sizes of one, and extremely precise assembly is required to insure safe flight over the forty-year life of an aircraft. The IMPCA system was implemented to solve the control problem, to eliminate the problems associated with the massive amounts of paperwork, and to coordinate not only the activities of assemblers but also the activities of the many other functions involved in the assembly of the aircraft.

The strategy behind the conversion from an old to the new IMPCA system was that everyone involved in the design and use of the system would be considered as equals. This strategy meant that all assembly personnel, quality assurance groups and others had the same briefing on the new system. Two years prior to the completion of the system, an attempt was made to provide all employees with knowledge of the intended new system. The philosophy presented to the employees was that no one fails and no one gets fired. Parallel systems were operated for nearly six months to minimize the risk associated with system downtime. (Lost information costs were estimated at $150,000 per hour.)

Prior to the successful development of the IMPCA system, Northrop pursued a traditional approach which involved the design of the system by experts. The approach failed. With the IMPCA project, an approach of "users are designers" was used. Assembly workers, for example, were treated as project team members. Committees were composed of representatives from each of the functional areas and the users defined the system requirements. The lead role of the MIS group was straightforward: find out and apply a method of team-based transitioning that would work at Northrop. The program manager was familiar with manufacturing. Project cost growth occurred, however, because the project takes time and requirements change over time.

The investment in the IMPCA system is part of the Department of Defense (DOD) Industrial Modernization Incentives Program (IMIP) in which cost savings resulting from defense programs are shared between the contractor and the contracting agency. Projected savings totaled $21 million. In May, 1989, after full production use for four months, cost reductions became obvious. The estimated $21 million in savings over the life of the aircraft did not include a savings estimated at three times the $21 million amount in intangible benefits resulting from such factors as error free manufacturing and no rework or schedule changes. By February, 1990, benefits such as less scrap and smaller, less costly mistakes, as well as fewer and less dramatic engineering changes were visible. The amount of savings in intangible benefits was re-estimated at four times the $21 million figure. The program appears to have convinced management that very large, integrated projects are both feasible and desirable. One manager commented that using the IMPCA system is like having a meeting of 1000 people.

In summary, the IMPCA system implemented at Northrop appears to be successful in terms of achieving estimated cost savings and many other difficult to quantify benefits. However, the emergent issues of the MIS-Manufacturing relationship in this case revolve around learning that the

"expert" approach to system development for manufacturing did not work. In the case of Northrop, the alternative taken after initial failure was the team approach, where teams or committees consisted of representatives from the various functional areas. Apparently when a cross functional group works well, it helps to reduce the stress at the MIS-Manufacturing interface. Stress was further reduced through the operation of parallel systems and through the elimination of the fear of the employees relative to failure and lay offs.

3.4 Integrated Paper Co.

The Integrated Paper Co. [5] case description has two parts. First is the experience this paper company had with a Greenfield mill and the second is the experience which was transferred to an existing facility which was being modernized.

The essential feature of the Greenfield mill project is that it was the first experiment by the company with a three-level information system hierarchy, as depicted in Figure 3, as compared to a two-level information system hierarchy consisting of the top and bottom levels shown in Figure 3. Integrated Paper Co. acquired "off-the-shelf" systems for the top and bottom levels of the hierarchy. The MIS function was assigned responsibility for the development of the middle level, the integrating level and, thus, played a significant role in the project -- management and initial responsibility for the success of the project. The resulting approach taken to the development of the middle level system was very traditional.

Starting with a blank piece of paper, MIS solicited descriptions of information requirements from a subset of functional area plant employees who "wanted it all," designed very specific screen functions to meet these requirements, and expected the system to work. Neither a general system overview nor a functional specification was developed. A new computer supplier was selected. Little or no time was planned for learning and the installed computer system lacked sufficient capacity to support the application software developed. As a result of these problems, the middle level system in the hierarchy is used only partially.

When Integrated Paper Co. migrated the experience to an existing mill, a completely different approach was used. First, although corporate MIS individuals were involved in the project, plant personnel (process engineers) managed and assumed total responsibility for the project. The process engineers were trained in computer analysis and programming. System development followed a structured approach in which detailed functional specifications were defined first, and detailed systems specifications were developed prior to coding the system. MIS and manufacturing assumed joint ownership of the system and a plan showing separate as well as joint responsibilities was developed. Implementation of the system with its resulting change in information structure precipitated a change in organizational structure, and jobs were reallocated. For example, after implementation of the system, it was no longer clear who had responsibility for batch process raw material shortages.

The Integrated Paper Co. case is interesting from the perspective of viewing how learning and experience obtained during one development effort

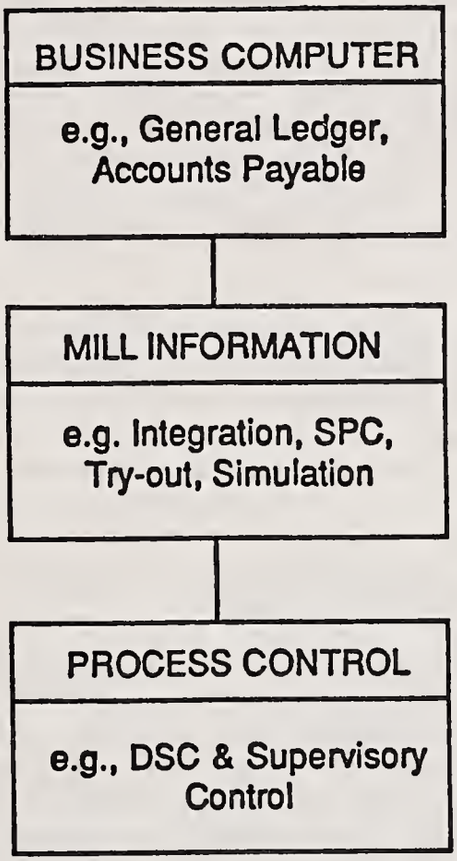


Figure 3. Architecture at Integrated Paper Co.

can be applied to a second effort. The interaction between the centralized MIS system development personnel and manufacturing was limited to line personnel almost exclusively during the Greenfield case; in the second mill, all plant personnel were involved. The implications resulting from this case are that the "correct" type of systems development person is different for new manufacturing systems and that decentralizing the "new breed" of development personnel is a highly likely result. Apparently, structuring changes for new decentralized systems development positions offers real hope of delivering systems that satisfy, reasonably, user requirements.

3.5 CCAS Inc.

CCAS Inc. [5] is a single plant but large subsidiary of a large corporation. This case represents another in which the MIS-Manufacturing relationship historically has not been good. A corporate and centralized MIS function, located geographically some distance from all plants, traditionally controlled the selection, implementation and use of computers by all plants. MIS personnel were seen rarely in the plant and were criticized for lack of understanding of manufacturing and for lack of interest.

The manufacturing process at CCAS Inc. is very high volume and repeti-

tive assembly. CCAS Inc. acquired a factory management and control system and worked with the supplier of the system to modify and enhance the software. As in several of the other cases, the factory management and control system is part of an overall operation improvement program focused on competitive factors such as lead time, quality and cost. The general business functions (accounting, payroll, etc.) for the subsidiary are operational on a mainframe located in the corporate office and are maintained by the corporate MIS group. Manufacturing planning functions also are supported on the corporate office mainframe. Bi-directional, electronic communication exists between the corporate mainframe and the plant-based factory management and control system.

In an effort to improve the relationship of corporate MIS with the plants, CCAS Inc. changed the direction of corporate MIS to distribute the control of manufacturing systems to the plants and to assume more of a guiding rather than a directing or controlling role in the project. Manufacturing personnel selected the system, with corporate MIS guidance. After the selection process, however, the implementation process became less user driven and more MIS driven. The plant staffed its own small computer systems group, and this group, which reported to the financial officer of the plant, was given responsibility for the project.

A prototype process was used to develop functional specifications for modification and enhancements to the software. Using the part types and process flows of CCAS Inc., the software supplier and the implementation team developed a model of the material flow and the associated data flow using the standard or base software. Simultaneously, the software supplier trained the implementation team on the use of the system and in the development of functional requirement specifications. Through an iterative modeling process, functional requirement specifications were completed. The supplier of the software used these specifications to develop the modifications and enhancements to the system.

The implementation team consisted primarily of the plant's small computer systems group. Although users were involved throughout the process, involvement was very limited. Very little opportunity was afforded the users to understand the capabilities of the system prior to the definition of desired changes or additions to the functions of the system. Also, many of these changes and additions were defined by the MIS oriented individuals. As a result, the functional specification process was not completed on time, many more changes and enhancements were made than were initially projected, and ultimately many of the modifications and enhancements made to the system subsequently were not used as users became familiar with the technology. As a result of the significant additional implementation time exceeding the scheduled time, not all enhancements or modifications were made. The development process was iterative: produce a modification or enhancement, let the users try it, change it as requested, let the users try it, etc. Thus, even after installation and production use of the system, many changes to the previously made changes were requested. The iterative process, which was not short cut by allowing plant personnel to use the system to increase understanding of requirements, also supports the finding of Ettlie and Getner [1989] where users prolonged the software development cycle into maintenance in order to compensate for lack of understanding of requirements. To worsen matters, the involvement of corporate MIS individuals was sporadic and unpre-

dictable relative to the definition of the required electronic communication to the host mainframe systems. This outcome may have resulted from a corporate MIS function in transition.

Unlike some of the other cases, there is not a concensus relative to the success or failure of the control system implementation at CCAS Inc. The system is used and meets some objectives: control and visibility of the process, lower costs resulting from decreased inventory, faster throughput and increased quality. However, not all of the expectations of senior managers were met. Also, some factory floor operators do not like the system, probably due to the fact that some necessary changes never were implemented. Additionally, not all of the functions of the sysem are being used. Thus, the overall system will not meet its full potential. An iterative design methodology was employed and the development process was stopped short of completion.

3.6 PWAS Inc.

PWAS Inc. [5] is a multiple plant division of a large corporation. This case presents a situation in which a standoff was created and still exists between centralized, corporate MIS and the advanced manufacturing technology group of the division which still is attempting to acquire a factory management and control system. The advanced manufacturing technology group of PWAS Inc., which has no internal MIS function specific to the division, is responsible for the introduction of new technology to the plants of the division. This case represents another in which the centralized, corporate MIS function traditionally has controlled the selection, implementation and use of computers by all plants. As in some of the other cases, corporate MIS personnel also are criticized by those in manufacturing functions for their lack of understanding of the process and the lack of understanding of the information system requirements of the manufacturing users.

Users in the various manufacturing functions of PWAS Inc. understood the requirements for integrated resource management and control by an information system, particularly as these requirements affect short interval scheduling. Personnel were adjusting continually to ever changing conditions based on contingencies as they occurred in the factories. These manufacturing personnel were well aware of the effect of one set of constrained resources on another. In most cases, due to lack of information system support, manufacturing personnel were managing the contingencies manually through a combination of shrewd native ability, constant vigilance and steel nerves. A major difficulty relative to an integrating system implementation was the inability of the manufacturing personnel to articulate these integration requirements to the MIS organization due to lack of technical expertise in the integrating capabilities of computer technology and information systems. Because of this difficulty, the default approach for each functional organization in PWAS Inc. manufacturing tended to be to focus on the information system requirments necessary for it to achieve its own objectives.

At the same time, the corporate MIS function continually was being asked to develop or deliver computer systems for multiple functional organizations within manufacturing, each with a seemingly unique set of requirements. The corporate MIS personnel understood the integrating capabilities of computer

technology and information systems, but they lacked the hands-on manufacturing experience necessary to apply these capabilities to a seemingly diverse set of manufacturing requirements. As a result, the MIS group had a very limited perspective as to what was required to integrate effectively the management and control of all resources of manufacturing. As a result, MIS personnel were unable to help manufacturing with the definition of complete systems requirements that included the interrelated elements necessary to achieve integration.

Corporate MIS individuals were, however, well aware of the efforts required to develop, deliver, and maintain any system. They also were concerned constantly about the proliferation of computer hardware, operating systems and data base management systems. Perhaps because they understood these issues much better than the overall requirements of the manufacturing users, they tended to focus on solutions to their own problems rather than to those of the manufacturing users. Requests for quotation developed by the corporate MIS group for PWAS Inc. characteristically were more concerned with defining hardware platform, operating system and data base management system restrictions and typically were focused exclusively on the objectives of a single manufacturing function, due to the inability of MIS to help manufacturing articulate cross-functional, interrelated requirements. They were less concerned with adequately addressing integrated functional requirements of a system which would meet, adequately, the information requirements of the manufacturing users.

The advanced manufacturing technology group of PWAS Inc. became increasingly concerned about the trend within the plants to propagate "islands of information," resulting from the implementation of stand-alone, non-integrated information and automation systems. The scope and objectives of these islands of information tended to be very self-serving and focused on the short-term requirements of specific functional areas within manufacturing. For example, stand-alone quality data collection systems, labor reporting systems, production systems, tool management systems and MRP-driven shop floor systems were growing in number throughout the plants in the division, yet the overall information needs were not being met. An integrated system was deemed necessary over these "point solutions" in the plants for many reasons, including the following: the availability of accurate, time consistent data to all users simultaneously; less data entry by the workers to create more meaningful information; support of a cohesive product and process history; simultaneous and coordinated management of resources; and control of the process. With respect to the latter, an integrated system was expected to enable manufacturing personnel to react to contingencies immediately and to support flexibility of the process. The advanced manufacturing technology group knew that, with point solutions, there simply are too much data to assimilate or interrelate fast enough to react to the changes which occur in a factory. At best, with point solutions, one can control one or more processes serially but can not control the interaction of the processes.

To address the situation, the advanced manufacturing technology group enhanced its staff with individuals who understood the manufacturing process but also understood the capabilities of computers. The group created and distributed a detailed specification as part of a request for quotation process, went through a formal evaluation of the responses to the request for quotation, selected a software product, and had an external consultant vali-

date the entire evaluation process. Acquisition of the selected system, however, was thwarted by corporate MIS because the acquisition would result in the introduction of a new hardware vendor into the division. For over two years, the manufacturing functions within PWAS Inc. and corporate MIS have battled over the initiation of the implementation of an integrating factory management and control system. Meanwhile, the manufacturing information requirements of the plants of PWAS Inc. remain unfulfilled.

4. Discussion and Summary

Table 1 presents a number of propositions which are based on the experiences of the case manufacturers and directed toward the improvement of the MIS-Manufacturing relationship.

These propositions have in common the application of cross functional manufacturing users and MIS expertise to the implementation process and the creation of a forum by which both the users and MIS can contribute their respective expertise to the implementation process. In at least one of the cases, as members were added to the implementation team, differences between MIS and manufacturing were reconciled more readily. One might ask who and how many should be involved in the process. The cases suggest that when the necessary perspectives on the entire manufacturing process and the required capabilities are represented, the functional representation for which will vary from one organization to another, the team or other forum of representation is complete. Particularly important is the application of the right resources during the development of information system requirements. Right resources means experienced, knowledgable individuals from the various functions of manufacturing and from MIS. It is suggested that the forum for joint and simultaneous contributions and the contributions themselves improve the MIS-Manufacturing relationship. It is thought that such a forum fosters rapid learning and communication of requirements, resulting in effective implementations.

Not all of the manufacturers described in this paper have thus far enjoyed the same degree of success [8] with the implementation of a factory management and control system. The many integrating functions of such a system and the experiences presented clearly make a case for the mandatory factors of user-driven implementations, MIS expertise and manufacturing knowledge. Other experience supports the importance of these factors. For example, users managing their own systems development "increases significantly the chances of systems being installed on time, within budget and satisfactory to the users [GRI, 1985/86]." Line managers should "take the lead" and "drive the applications" of information technology [EAR, 1989]. Other experiences [HAM, 1987] suggest that MIS ought not only to understand the manufacturing process, but show the manufacturing users that they understand and that they are applications oriented. Another factor which is proposed as critical to success relative to the cases described is well-defined information system requirements. All of the experiences suggest that the process of developing good system requirements is difficult in its own right and probably impossible without the involvement of knowledgable users and experienced MIS. These experiences support the findings of Ettlie and Getner [1989] that users (and technology vendors) have a great deal of difficulty learning requirements in plants. These researchers recommend that require-

Table 1. Summary of MIS-Manufacturing Case Experiences.

CASES	LESSONS LEARNED	SUCCESS/ FAILURE	DERIVED PROPOSITION
PGAS Inc.	Applying knowledge at the right time, and good people communication are key success factors.	Success. Meeting objectives, including 99.9 percent inventory accuracy. Less than one-year payback.	The MIS-Manufacturing relationship is improved when each function can contribute significantly to the implementation with the skills each have.
BAA Inc.	Have MIS and manufacturing users do what they are best at doing.	In process. User satisfaction thus far. Less than one-year payback.	Using a pilot system development methodology accelerates the development of well-defined requirements and improves the MIS-Manufacturing relationships.
Northrop Inc.	Designer "expert" does not work.	Success. Savings of over $20 million.	Adopting a group requirement development process reduces conflict at the MIS-Manufacturing interface.
Intergrated Paper Company	Everyone is a user.	Aspects of success and failure. Something fails when it is not used.	MIS (new breed) is decentralized in the best development, with all manufacturing personnel as users.
CCAS Inc.	System requirements definition cannot be delegated to MIS, even decentralized, "manufacturing" MIS.	Aspects of success and failure. Meeting some, but not all, objectives. System not used to full potential.	The system development process is extended when manufacturing users do not contribute equally (with MIS and the software supplier) to the requirements definition.
PWAS Inc.	(Not applicable)	Failure. MIS-manufacturing standoff on system acquisition.	The definition of system requirements in an integrated environment is a function which must be shared by those who understand computer technology and those who understand the many functions of manufacturing.

ments learning and development of tools to promote the development of requirement specifications should be the focus of both users and technology vendors for promoting successful modernization.

Three of the cases provide examples of the use of methods and tools which fostered the learning of requirements. These three manufacturers used existing software to prototype the process, and in one of these cases the manufacturer used the software in production mode, to accelerate the understanding of system capabilities prior to the requirement specification process. One of these implementations is very successful, another is in process but enjoying the first stages of success, but the third is only partially successful and the implementation process was terminated prematurely. The latter implementation differed from the first two in a very visible way: manufacturing users and MIS did not work together with the software supplier to define requirement specifications. This experience suggests that a design approach not only incorporate requirements learning and development of tools but also that the learning process and the use of tools be enjoyed by both MIS and the manufacturing users. This suggestion also is compatible with the findings of Ettlie and Getner [1989] who suggest that the problems of manufacturing systems are caused by lay designers who are not acquainted with manufacturing needs but are most responsible for decisions. These researchers further note that user satisfaction with systems is higher when users share responsibility for the design of the systems, including equal credit where credit is due and equal blame when problems occur.

The case experiences described in this paper also highlight what other researchers have found, namely that implementation of manufacturing systems cannot be approached in the same manner as are many traditional data processing projects relative to the definition of requirements and the management of the project. Ronen and Palley [1988] suggest, for example, that the application to manufacturing systems of the design tools and methodologies developed for financial applications is one of the reasons for the generally poor performance of manufacturing systems. Table 2 presents a summary of the differences identified by Ronen and Palley [1988] who note that the least complex manufacturing systems [9] resemble the most complex financial systems. Others agree that new systems in the areas of CAD and CAM have little resemablance to the MIS systems of the last twenty years [VAN, 1988]. A good look at the sharp contrasts presented in Table 2 leads to the recommendation that, to improve the MIS-Manufacturing relationship, any new design approach should take advantage of the knowledge of the differences between the environment of manufacturing and the environment to which the MIS function is accustomed -- finance.

The system development process which was used by BAA Inc. involved first using then modifying and enhancing an existing vendor-supplied system. Recall that the use of the system prior to the development of requirement specifications gave users the opportunity to understand the functionality and integrating capabilities of the system, and gave the software supplier and MIS participants the opportunity to understand the BAA Inc. manufacturing process and its requirements. Others [EAR, 1989] suggest steering away from past conventional information system development methods in favor of encouraging prototyping to discover detailed needs. This prototype approach, and perhaps the pilot, could be the necessary hybrid of the structured design approach and the iterative design approach to developing systems, neither of

which, singly, is said to work well for manufacturing systems [RON, 1988]. The approach certainly is compatible with the finding of Ettlie and Getner [1989] that users tend to prolong the development cycle into maintenance in order to compensate for lack of understanding of requirements. Perhaps computer-aided software engineering (CASE) tools [10] may become effective in manufacturing systems development were methods incorporated to facilitate the process of learning and communicating requirements in the complex environment of manufacturing systems.

Table 2. A Contrast of Manufacturing and Financial Systems.

CHARACTERISTICS	MANUFACTURING	FINANCIAL
Data	Many types, with more needed for automation. Low volume per data type. Highly dynamic. Greater changeability and timeliness. Shorter lifespan.	Few types. High volume per data type. Relatively static. Less changeability and timeliness. Longer lifespan.
Systems Hardware	Much greater variety of industrial hardware as input/output.	Much less variety of industrial hardware as input/output.
Application Software	More fluid. Developed for ill-defined, rapidly changing requirements.	More static. Better defined requirements. Fewer changes.
Systems Procedures	Informal.	Highly formalized.
People	Informal behavior. (Manager considered less disciplined by financial counterpart.) Heterogenous users.	Formal behavior. More homogenous users.
Information	More complex, and rapidly changing.	Less complex.
System Environment	Systems influenced greatly by rapidly changing environment. Adaptive/organic systems.	Systems more mechanistic.

NOTES

1. The lack of manufacturing and CIM knowledge by chief executives, the application of traditional financial methods to CIM justification, and the organizational conflict that occurs during implementation also are suggested by Fossum [1986, p. 183] as being among the most significant barriers to CIM implementation.

   Manufacturing is defined to span product concept through field maintenance and, therefore, includes product and process design engineering as well as plant operations.

2. Ranked two and three by the respondents to the Industry Week survey [Sheridan, 1989, p. 36] are executive ignorance and inadequate planning or lack of vision.

3. Factors ranked as important or a significant help to CIM implementation progress are: a formal CIM plan, with an architecture as its basis; integration as a primary criterion for the selection of individual CIM components and subsystems; an installation sequence of components and subsystems based on priorities; a user-driven CIM effort; a strong technical staff of individuals who understand manufacturing and who understand the CIM goals; a steering committee of high-level, functional area executives; a full-time project manager for individual CIM component or subsystem implementations; formal or adhoc implementaion teams of participants from all functional areas of the business unit, and stability of a core of these team members; user-participation in the formulation of CIM system specifications and the implementation of CIM components and subsystems; user participants who have knowledge of and authority and accountability for their functional area, understand how their work relates to the CIM goals, understand manufacturing, and have good interpersonal skills; direct involvement of functional area managers in the implementation process; consideration of human resource requirements; a substantial education and training investment; a formal plan for training/retraining personnel.

   The concept of implementation used in Fossum's research and in this paper is based on Bodenstab's definition [1970, p. 64]:

   > Implementation is the process of taking a technically sound computer system and making it operate effectively in the real environment of the business world. It involves getting people to interface or relate with the various facets of the system, to follow procedures to conform to the data discipline imposed by the system, and to act on information generated by the system.

   Based on the current status of CIM technology, however, the definition is extended to include making the computer system technically sound.

   Many companies have reinforced the traditional hierarchical organizations with devices that facilitate communication and problem-solving for CIM implementation [Fossum, 1986, pp. 297-305; MSB/NRC, 1984, p.32]. These devices take the shape of steering committees, project managers, and

implementation teams. When the support mechanism for CIM becomes a way of life in organizations, the need for these coordinating entities, but not the functions they perform, may disappear.

4. For a description of some of the functions of a factory management and control system, the reader is directed to the following source:

   Bedworth, David D. and James E. Bailey, Integrated Production Control Systems. New York: John Wiley & Sons, 1987, pp. 26-37, 177-179.

5. The name of the organization has been disguised to support the anonymity of the manufacturer. Any similarlity between the names used in this paper to disguise the manufacturer cases and a real name of an organization is purely coincidental.

6. The term "large" as applied to a business unit means that there are 500 or more employees in the enterprise (corporation, division, subsidiary, group or establishment).

7. The system initially was described in the article by Jones [1989]. Subsequent interviews were conducted by one of the authors in May and June, 1989 and in February, 1990.

8. Success is defined in all of the following ways [Fossum, 1986, pp. 314-315]: formalized objectives are met, the system works according to plan or in an acceptable manner, and anticipated benefits as well as windfalls are realized. Similarly, failure is defined in any of the following ways: formal objectives are not met, the system does not work according to plan or in an acceptable manner, the system is not used by all who should use it, and expected as well as unexpected negative outcomes occur.

9. Ronen and Palley [1988, p. 292] divide manufacturing systems into two types: technology oriented (CAD/CAM and robotics, for example) and management oriented. They further divide management oriented manufacturing systems into two categories: project management systems such as PERT and CPM and resource management systems such as MRP II, scheduling and capacity planning.

   A factory management and control system has elements of both a technology oriented and management oriented manufacturing system.

10. Ettlie and Getner [1989, p. 129] state that computer-aided software engineering (CASE) tools "have failed to solve the problem of timely, effective delivery of manufacturing systems." According to these authors, the view by CASE technology of the software development process is much too narrow for manufacturing systems. CASE tools were not intended to deal with the complexities posed by requirements generation for manufacturing systems -- many types of people, various disciplines and backgrounds, and multiple organizations.

REFERENCES

Bedworth, David D. and James E. Bailey, Integrated Production Control Systems. New York: John Wiley & Sons, 1987.

Bodenstab, Charles J., "10 Tips for Successful Implementation of Computer Systems," Financial Executive, Vol. 38, No. 11, November, 1970, pp. 64-70.

Earl, Michael J., Management Strategies for Information Technology. New York: Prentice Hall, 1989.

Ettlie, John E. and Christopher E. Getner, "Manufacturing Software Maintenance," Manufacturing Review, Vol. 2, No. 2, June, 1989, pp. 129-133.

Fossum, Barbara M., A Normative Model for CIM Implementation. Doctoral Dissertation, The Department of Management, The Graduate School of Business, The University of Texas at Austin, August, 1986.

Grindlay, Andrew, "Management of Computer Integrated Manufacturing," Business Quarterly (Canada), Winter, 1985/86, pp. 68-71.

Hamilton, Rosemary, "The Long and Winding Road to CIM," Computerworld, June 29, 1987, p. 8.

Jones, Sam L., "Northrop Unveils Navy Jet From 'Paperless' Assembly," Metalworking News, April 3, 1989, pp. 2, 31.

Manufacturing Studies Board of the National Research Council (MSB/NRC). Computer Integration of Engineering Design and Production: A National Opportunity. Washington D.C.: National Academy Press, 1984.

Ronen, Boaz and Michael A. Palley, "A Topology of Financial Versus Manufacturing Management Information Systems," Human Systems Management, Vol. 7, No. 4, 1988, pp. 291-298.

Sheridan, John H., "Toward the CIM Solution," Industry Week, October 16, 1989, pp. 35-82.

VanNostrand, R.C., "A User's Perspective on CAD/CAM and MIS Integration," CIM Review, Winter, 1988, pp. 38-45.

TOWARD A NEW CIM ARCHITECTURE FOR SANDIA LABORATORIES

JAMES R. YODER

ABSTRACT

Sandia National Laboratories has experienced several generations of design and engineering automation. Each successive iteration brought improvement to a specific function. The changes in technology were point optimized, i.e., the improvements were made to narrow functional areas and not necessarily to the entire organization. Further, implementation plans rarely included the transfer of information from one function to another. Consequently, the Laboratories began a significant effort to design an overall architecture under which integration of disparate activities could take place and which would serve to provide an information path between all functions. This paper describes the current version of the architecture and provides a description of the process that led to the architecture.

1. Introduction and motivation

Sandia National Laboratories, a subsidiary of AT&T, is a multi-program, multi-discipline Department of Energy laboratory which has responsibility for the design and development - but not the manufacture - of a broad spectrum of aerospace products. The products are built in a loosely integrated complex of factories managed by several aerospace corporations. Therefore, the Computer Integrated Manufacturing problem is two-fold: 1) computer support of the design, development, and qualification of hardware and 2) technology transfer to manufacturing (figure 1). Technology transfer from one industrial organization to another is an often difficult process which must transcend not only technical boundaries but inter-corporate cultural and political boundaries as well.

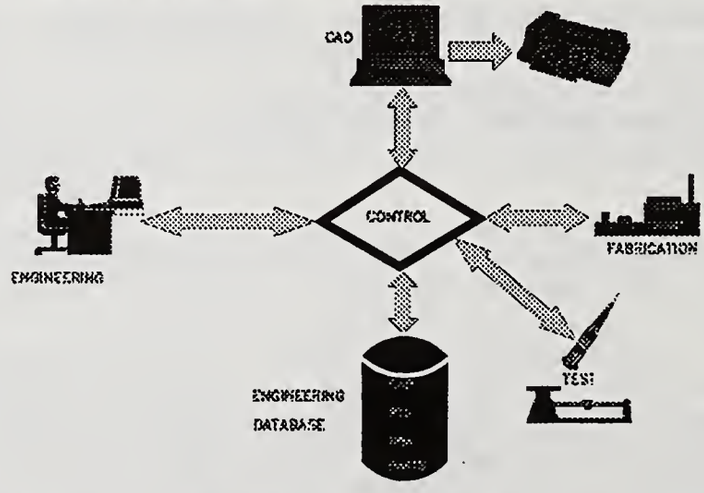


Figure 1. Computer Integrated Manufacturing at Sandia

At Sandia and elsewhere, the evolution from manual engineering and drafting technology to the electronic or computer aided world has been rapid.

However, all elements of the CIM structure have not evolved at a uniform pace. CAD technology preceded CAE by several years. Further, although advanced CAD/CAE workstations are commercially available, the supporting infrastructure (design databases, computer networks, procedures, etc.) is typically developed in-house and consequently lags the market-driven segment of CIM. Design support is, presently, a fragmented and highly specialized collection of services. Clearly, a new CIM architecture is needed.

Following nearly one year of study, we have completed the basic design of a new CIM network architecture and we have begun the initial phases of implementation. We began with an historical perspective and scope, developed assumptions and requirements, and established our business policies. Having thus defined the foundation for a new architecture, we then explored new technology, developed the architecture, and started implementation. This paper will describe the process of CIM architecture design, depiction, and implementation planning at Sandia Laboratories.

2.0 A useful definition of "architecture"

With the thought that a system architecture should be useful and lead to the development of a system that meets requirements, we first attacked the problem of deciding what we would mean by the term "architecture". There are several sources for descriptions of computer and network architectures. But, for those attempting to define an functional system architecture, especially one which must fit into an extant structure, we are offered only general guidance. We will try to utilize that guidance to formulate a concept of a "CIM Architecture" in the design laboratory context.

First, with respect to the need for an architecture, we note that in the absence of a theoretical or formal framework, architectural designs have conventionally been evaluated with respect to both logical correctness and performance only after they have been implemented as physical systems [Dasgupta, 84]. Architectural errors tend to be troublesome when they are hard wired or hard coded into the system. In fact, an architecture is necessary only because we are dealing with a level of complexity that is incomprehensible to a human in ordinary circumstances [Doran,79].

We want to store the design in some formal representation but, of course, storage is not the objective. We really need to be able to communicate the design to others in a precise, unambiguous way [Hayes, 78]. One should be able to understand the architecture both from a systems point of view and at the level of minute detail (n.b., this does not mean that implementation details are a part of the architecture - it does mean that the smallest detail can be evaluated in the context of the system architecture).

A "point of view" is important. Although the total image of what a system should be is compounded from the views seen by a range of users, there are cases where the view of one group of users is the dominant concept behind the systems's architecture. In these cases, the system is designed as if it were to be used preeminently for one particular purpose. Other applications may be recognized but these are accommodated by slight modifications or extensions rather than by a radical alteration of the architecture [Doran, 79].

It turns out that there are only two useful viewpoints in the logical representation of a system architecture: the User View and the System View - although the System View might be represented both by a Designer's View and an Implementor's View. The level of detail to be included in an architecture is important in another dimension. One may develop an architecture for use at different levels of comprehension or, equivalently, develop different levels of architecture as a function of the need for understanding specific issues or features [Dasgupta, 84].

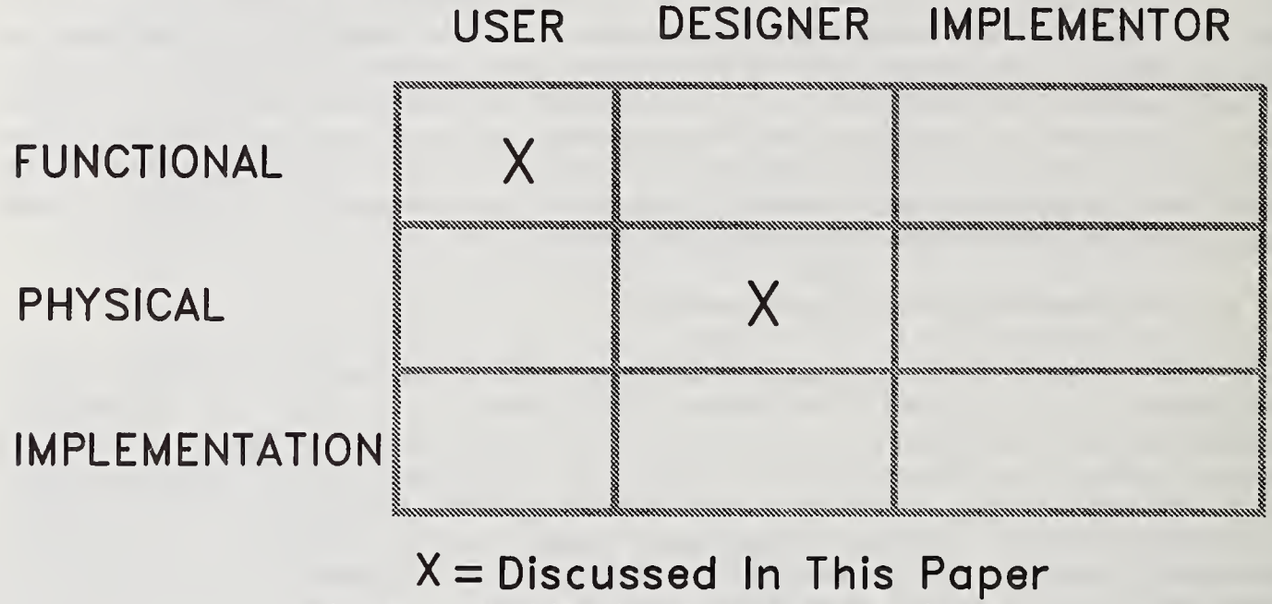


Figure 2. Architectural Layers and Views

Given these thoughts and some general concepts of design [Meijer,1983], we formulated a matrix structure to depict the architecture. The design depth is provided by depicting three levels: Functional, Physical, and Implementation layers. For each layer, we will provide a User, Designer, and Implementor view. All architectural layers are mutually independent. Although one layer may use the functions of a lower layer, it is not important how that function is performed. We will primarily illustrate the functional layer in this paper since lower layers are neither complete nor particularly interesting to those not responsible for implementation (figure 2).

3.0 Scope

One must decide the elements to include under the umbrella of Computer Integrated Manufacturing. Clearly, we include processes that lead directly to the design and manufacturing of a product. What about other supporting processes: field traceability, test and qualification data, preferred parts characterization data, graphics libraries, and so on? To support CIM, all related processes must be considered and, if possible, included in a seamless architecture (figure 3). It is convenient at this point to incorporate decisions related to which existing facilities to integrate into the system versus which facilities to develop or redevelop.

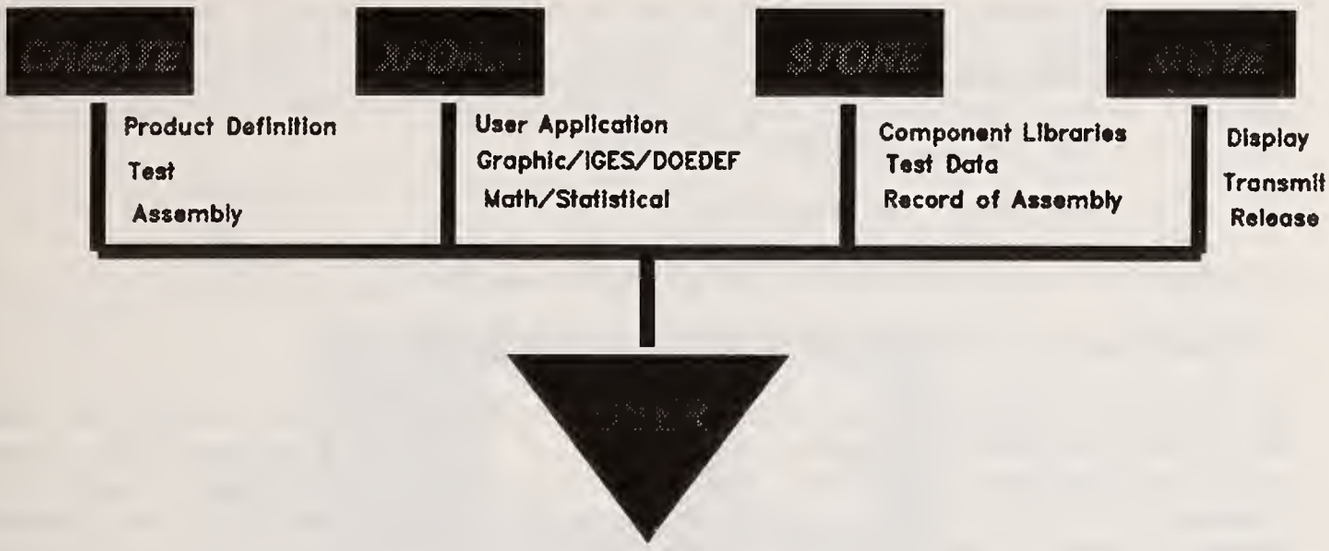


Figure 3. User View of Candidate Applications

After study of the present support structure and development directions, we realized that many elements of the design process have not really been automated but have merely been "electrified". For example, although we have captured configuration tracking data on computers for years, the design objects to which these data relate (drawings, specifications, change orders, etc.) have remained on microforms for manual storage and retrieval. With more recent technology, we will be able not only to index the design objects but to construct an Electronic Information Bank which delivers needed design information directly to a user's desk.

Further, although we have expended effort to develop supporting databases, an expert user function is apparent in every major application. The end user (a design engineer, for example) can rarely seek and obtain information directly from the databases. Were an engineer to try, the variety of operating environments, procedures, and data formats would be overwhelming.

4.0 Assumptions and requirements

We discovered that we must not only base a new CIM architecture upon functional requirements and specifications but these must, in turn, rest upon well considered business and technical policies. We examined business policies (e.g., "Engineering access to design databases is restricted to read-only") before any technical issues were considered. We then developed a-priori strategic directions (e.g., "Will we attempt to integrate office automation functions into the technical workstation environment?"). Next, we reviewed present and evolving standards in CAD/CAE, operating systems, network protocols, user interfaces, and office automation. From the investigations, we formulated and agreed upon a complete set of business and technical policies related to several important components of the architecture:

- Operating Systems (both servers and workstations)
- Network Protocols

- Database Management Systems
- Data Exchange Formats and Media
- Application Software (CAD/CAE)
- Retirement and Migration

Eventually, the architectural requirements emerged.

5.0 Technical directions

New computer networking and user interface technologies will answer the need for a seamless design support architecture. It should be possible to use standard workstation windows protocols, for example, to provide an apparently homogeneous access path to diverse computing environments. Bit mapped graphical user interfaces should provide the common "look and feel" necessary to reduce our dependence upon expert users. Clearly, the technical workstation market is trending toward a single operating system. However, much of the promising new technology and standards are embryonic and both the technology and application of the technology must be proven.

A typical design engineer can be confronted by more than a dozen distinct functions which, at present, are implemented on separate platforms and, as mentioned before, are supported by dedicated expert users. The architecture will take advantage of new user interface capabilities to not only provide a single direct source for all information but to provide a common "look and feel" for the entire system. We recognized that a user would best be served by a single system which delivers access to a "collection of services" (figure 4).

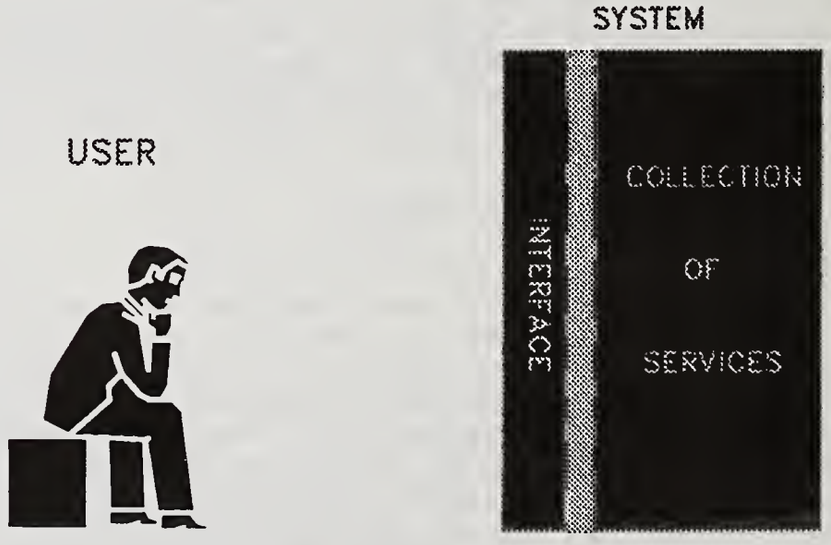


Figure 4. User View of System Services

The services provided include both information sources and functions. For example, an engineer may need to examine a design (drawing), analyze test data, and correspond with associates. The engineer may also need to initiate related functions such as release designs, authorize procurement, or obtain

tangible output. Under workstation control, the user can obtain the required data, communicate with others, and start processes by selecting the appropriate function as represented by a service icon. The related display is presented in a service session window on the workstation screen (figure 5).

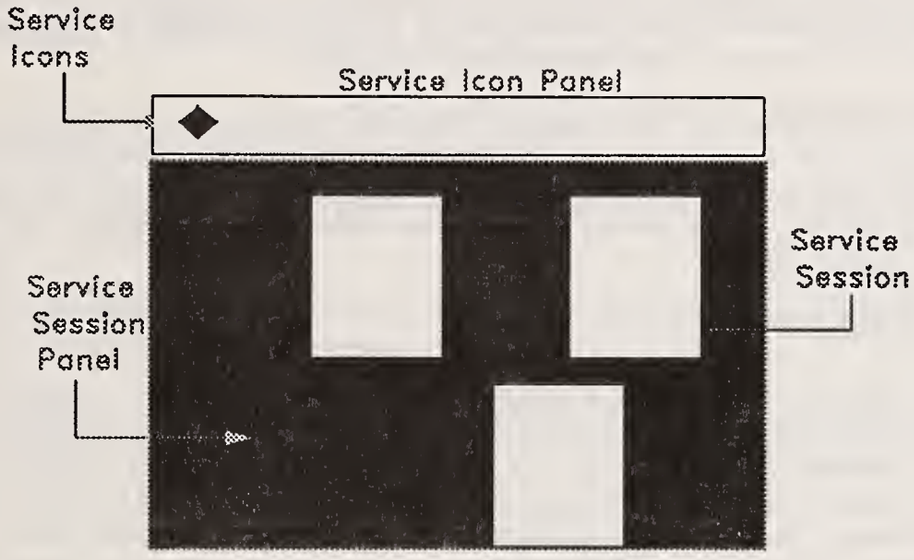


Figure 5. User Screen Layout

The service session windows will be connected to several diverse applications but the entire engineering support capability will be presented as if it were a single system. The user interfaces depicted in the above illustrations are not uncommon in the world of personal computer applications. On the other hand, the Sandia CIM architecture proposes to provide an integrated user interface over a very broad collection of application platforms. The platforms range from workstations and compute nodes to database servers - all from different vendors representing a variety of computing cultures (figure 6).

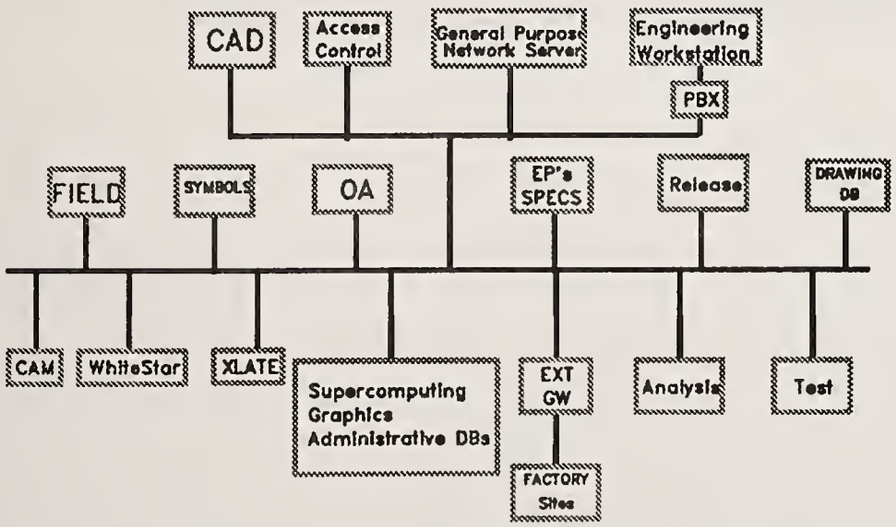


Figure 6. User View of the Functional Layer

Recent unpublished work at Sandia has resulted in the capability for two or more of the service session windows to simultaneously share (view and manipulate) the same object. Thus it will be possible to allow for an additional level of interaction, i.e., not only between engineers and computers but also between human collaborators supported by computer based tools. This feature of the system will provide important support for the introduction of concurrent engineering practices.

Although not addressed in this paper, computer security is an issue that must be considered in nearly every CIM application. Computer security requirements are often orthogonal to computer integration and must be addressed "up front." In fact, the whole network design may be governed by the security requirements.

6.0 Implementation

Although new technology and standards appear to offer solutions to CIM architecture requirements, implementation will be non-trivial. It will be necessary to reconfigure much of the present computing and network structure. Hundreds of computer programs must be developed, redesigned, or recoded. Several databases must be developed. It is fortunate that the concept, at least, of object oriented design is maturing as we begin our work.

A change in culture, however, may be the most difficult objective. "Recent events suggest that - information economy or no - people are not prepared to swallow information technology whole. Some attempts to weave computer and communications technology into the fabric of daily life have fallen on very hard times." [Wright, 1990]

The "seamless" integration of computers and computer-supported functions will result in reordering the structure of the organization. One may find the "engineering" and "drafting" functions, for example, somewhat redefined. The roles of secretaries and other support personnel may also be significantly altered. A change in culture, however, is the primary benefit of CIM and, to this end, we view the changes with positive anticipation.

REFERENCES

1. Dasgupta, Subrata, The Design and Description of Computer Architectures, John Wiley & Sons, New York, 1984.

2. Doran, R. W., Computer Architecture: A Structured Approach, Academic Press, London, 1979.

3. Eckhouse, Richard H., jr., and Morris, L. Robert, Minicomputer Systems, 2nd ed., Prentice Hall, NJ, 1979.

4. Hayes, J. P., Computer Architecture and Organization, McGraw-Hill, NY, 1978.

5. Meijer, Anton and Peeters, Paul, Computer Network Architectures, Computer Science Press, MD, 1983.

6. Wright, Karen, "The Road to the Global Village", Scientific American, March 1990, pp 83-94.

DISTRIBUTED KNOWLEDGE BASED SYSTEMS FOR COMPUTER INTEGRATED MANUFACTURING

SUDHA RAM
DAVE CARLSON
ALBERT JONES

Abstract

Efforts are being made in many organizations to use new technologies to automate and integrate the design, planning and manufacturing processes. The goal in developing these computer integrated manufacturing (CIM) systems is to increase productivity, improve product quality, and, minimize wastage of resources. This paper describes the information required to carry out major manufacturing functions. It also examines some of the special characteristics of these functions and their inputs/outputs. Based on this examination, the paper proposes an architecture for integrating distributed knowledge based systems (DKBS) to support CIM. An object oriented design for the various components of the DKBS is briefly described. Issues requiring further research are outlined.

1 Introduction

Most major manufacturing companies have made a strategic decision to make extensive use of computer technology in their factories. Initially, computers simply collected data and provided computational support to human decision makers. Currently, the focus is on shifting the responsibility of making these decisions to the computer. The short term effect of this move has been to improve the productivity of many individuals and the quality of their work. The long term goal is to have computers play a pivotal role in automating and integrating every phase of manufacturing. These computer integrated manufacturing (CIM) systems are expected to produce higher quality products at a reduced cost.

The process of automating all major manufacturing functions and integrating them into a successful CIM system is proving to be difficult and time consuming [DAV89]. Many researchers have concentrated primarily on the automation aspects of this process. Hierarchies similar to the organizational hierarchies that exist today, have been proposed by Jones & McLean [JON86]. Various functions are assigned to each level within the hierarchy, and interfaces between the levels are specified. Central to CIM is a database management system that facilitates sharing of data among the various components of the system. We believe that developing knowledge based systems to support these functions will be the key ingredient in CIM.

The objective of this paper is to show that in addition to database support, CIM requires the support of multiple knowledge based systems. An architecture for integrating multiple knowledge based systems is proposed to meet the requirements of a CIM environment. Each knowledge based system may interface with one or more databases. Section 2 describes several major manufacturing functions with details of their associated information flows. Section 3 examines

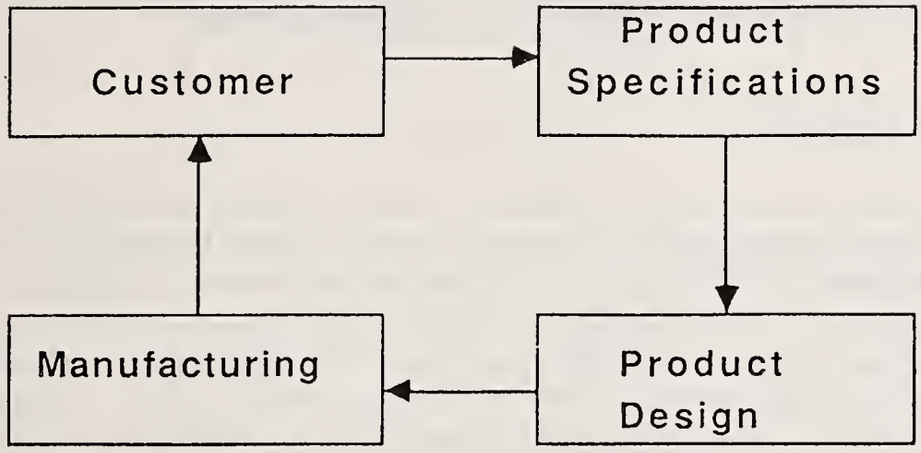


Figure 1: **Product Life Cycle**

the special requirements of the CIM environment which impact the design of a data and knowledge management system. Section 4 describes an architecture to integrate multiple knowledge based systems and proposes the use of the object oriented paradigm for an implementation based on this architecture. Several issues requiring further research are outlined in section 5.

2 CIM Functions

Figure 1 illustrates a high level view of the major steps involved in designing and manufacturing products. Requirements and orders from customers are used to generate product ideas. These ideas are refined to produce specifications for products. Design engineering results in detailed design for products which then proceed to the manufacturing phase. Product design, including research and development activities, must coordinate its functional responsibilities with marketing, manufacturing, quality assurance, and directly with the customers and suppliers. The finished product is delivered to the customer after quality control standards are met.

Ultimately, CIM systems will require the integration and, to the extent possible, automation of all major manufacturing functions including marketing and sales, product design, manufacturing engineering, manufacturing data preparation, production planning and inventory control, production scheduling, process supervision, and quality assurance. Each of these functions requires a different view on the data and knowledge used to manage its activities. Marketing deals with high level product *features*, product design requires data on the *performance* of components, and manufacturing engineers view the products according to the *physical attributes and configuration* of the components. However, a successful CIM system must integrate these various views and provide translations between the perspectives. To understand the impact this has on the design

of a data and knowledge management system for CIM, it is necessary to understand the details of these functions, along with their inputs and outputs (See Figure 2).

2.1 Marketing and Sales

Marketing and sales provide the primary interfaces between a manufacturing facility and its customers. They inform customers of available products, generate orders for selected products, price the products, negotiate delivery schedules, track shop floor performance in meeting these schedules, and ensure customer satisfaction after delivery. They also assist the customer in producing specifications for new or improved products. In addition, they often conduct a needs analysis to predict potentially profitable products. These activities can be supported by developing knowledge based systems that capture the expertise of marketing and sales analysts. To perform their assigned functions, the knowledge based system would need to retrieve and update information in numerous databases. These include product catalogs, customer orders, both the current and projected manufacturing capacities, finished products inventory, schedules, anticipated completion and delivery times, and orders for raw materials. It is important to note that these databases will contain textual and numeric data types, graphics, and 2 and 3-D images.

2.2 Product Design

Product development engineers receive a list of 'wants' and 'needs' from the product marketing and sales organizations. This list comprises the *features* of a new product. The development engineers may augment this information with further direct input from potential customers. A product design is then developed which satisfies these desired features, where feasible. These tentative design specifications are reviewed with marketing personnel and manufacturing personnel to determine whether the design can be produced at a cost which is acceptable to the marketing plan. Computer-aided design (CAD) tools are already commonly used to facilitate the creation and documentation of product designs, however, little support exists for the knowledge-based coordination which is essential for successful product development.

2.3 Manufacturing Engineering

The manufacturing engineering function coordinates the introduction and control of new production strategies. For example, Just in Time (JIT) strategies require effective communication among all functions shown in Figure 2. Since, in theory, JIT allows no time for inspecting incoming parts, a close working relationship must be maintained with suppliers. Poor quality parts or materials result in severe manufacturing problems, disrupting planning and scheduling functions which have a very low tolerance for defects under JIT control.

Future process engineering must accommodate information flows which operate *in parallel* to support the unit's business strategy. Future manufacturing organizations must support flexible

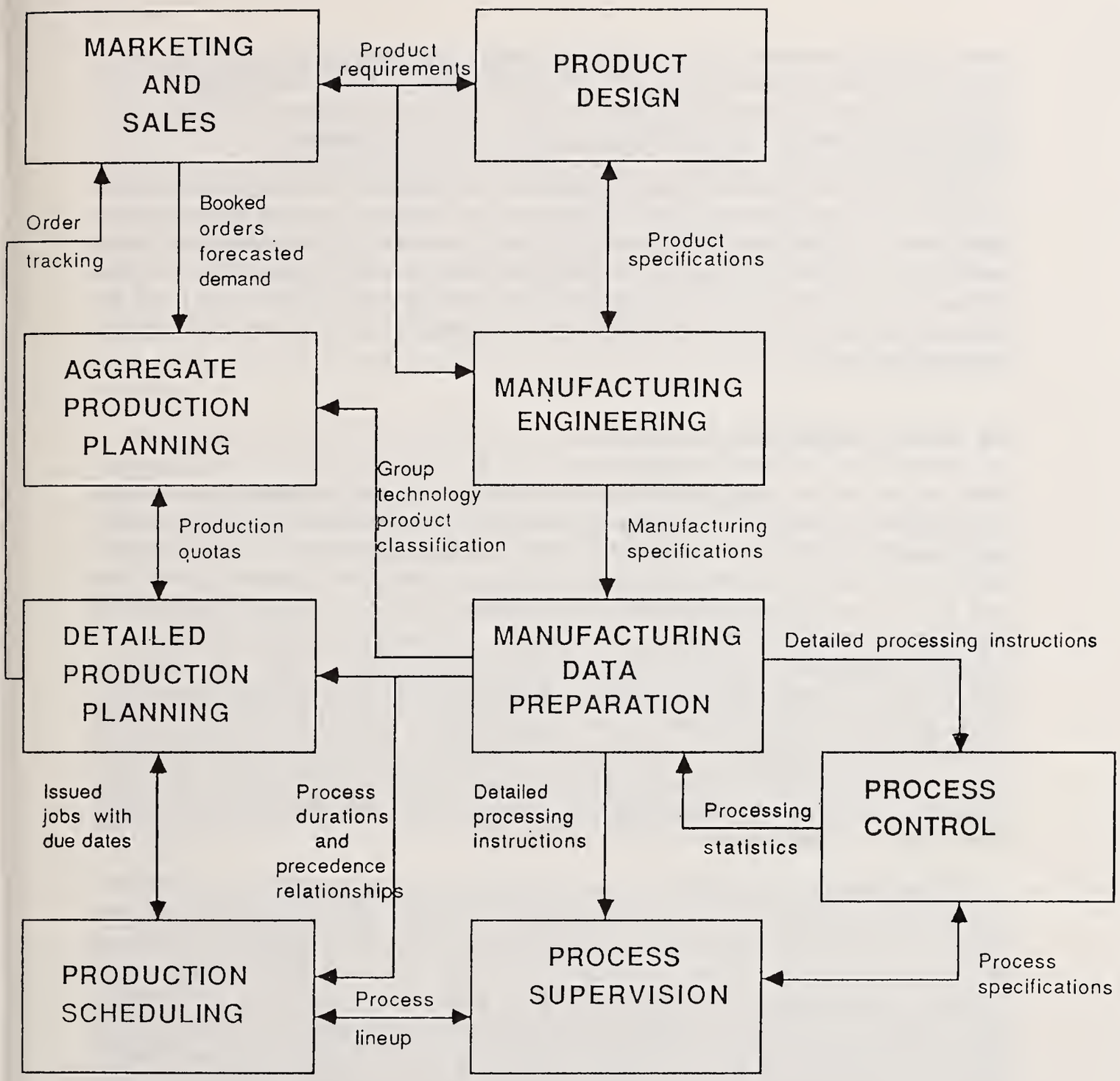


Figure 2: **Functions in a CIM Environment**

informational infrastructures where each function acts as a knowledge source capable of coordinating its activities with other agents. Current organizations pass information serially from one department to another within the hierarchical structure. Future organizations will require lateral communications between these knowledge centers in the CIM environment.

Manufacturing engineers are experts in designing products which satisfy the required quality standards at the lowest possible. This expertise must be available to product design engineers while product specifications are being developed. Several organizations have implemented expert systems which encapsulate this expertise in design for manufacturability [MAD87]. Once product specifications have been written in rough form, the manufacturing engineers will create the necessary manufacturing process specifications. The manufacturing data preparation captures these specifications.

2.4 Manufacturing Data Preparation

Manufacturing data preparation includes all of the functions required to generate the data needed to manufacture a product that meets a particular customer's requirements. The product design specifications (see above) include detailed 3-D drawings, solids, geometry data, tolerances, electronic circuit diagrams, and other required manufacturing specifications. These designs are then used by the manufacturing engineers to create a process plan which includes a complete list (including any possible alternatives) of raw materials, tools, machines, fixtures, and the precise machining instructions to be used during the entire fabrication process. Today, this is largely a manual task with some computer assistance. It requires a great deal of human expertise and significant interaction with the database. A three step procedure is used. First, the product is given a group technology classification code [CHA85]. This code is then used to retrieve existing plans for products with similar processing requirements. Finally, a process plan for the new product is created by revising, and possibly merging, one or more of these other plans. Knowledge-based system support can facilitate some parts of this activity.

The databases and knowledge bases required for this function are far different from traditional systems. There are storage problems and manipulation problems. Each design package and each process plan will contain large amounts of textual data, numeric data, graphics data, and image data all of which are interrelated in complex ways. This means that many records may be required to store a single piece of information. Each group users may require to have different *views* of the data as well as the capability to manipulate multiple data objects as a single unit. For example, the scheduler needs to know which machines the process planner has decided to use to make a part. A machinist will need detailed machining instructions in order to manufacture that same part. Expert systems can assist production personnel in diagnosing process and material problems. All of this information is part of the process plan. Finally, it may be necessary to keep several versions of design specifications and process plans for each product, and, quickly determine similarities and differences between versions. Product and process revisions must be carefully coordinated with production planning to minimize material obsolescence and rework required to upgrade finished goods inventory.

2.5 Production Planning and Inventory Control

Production planning is responsible for developing a list of "jobs" to be done on the shop floor during the next planning horizon (usually several months). In addition, it determines the hardware and materials necessary to do those jobs. This is accomplished in two steps. First, an aggregate production planner (APP) uses both the current and projected demands established by marketing to set production quotas and inventory requirements for each product type during each of several smaller time periods (usually one week) during the chosen planning horizon. These inventory requirements include all raw materials, tools, fixtures, castings, forgings, etc. needed to meet the demands. The APP continuously monitors and updates production quotas and inventory policies based on the feedback from the detailed production planner and updated demand forecasts from marketing.

The detailed production planner (DPP) uses these assigned quotas to generate production and inventory "jobs" for each time period. Before a production job is released to the shop floor for scheduling and processing, it is assigned a priority and a due date, and a check is made to verify that the required materials are on hand. Looking at future production quotas, the DPP may issue requests to external vendors to replenish inventories. The DPP monitors the differences between the assigned and anticipated job completion dates, and if needed, changes both the due date and the specification of the criteria to be considered in the scheduling function.

APP must access the following data: actual and forecasted demand from the marketing database; processing requirements for each of the products that make up that demand from the process planning database; current inventory status on tools, finished goods, work-in-process, and raw materials; and projected shop floor capacities. APP uses this data to update two additional databases. The first contains the number of each product type to be produced in the next planning period. The second includes the orders for new tools, raw materials, and any other items needed to produce those products. This latter database is also updated whenever orders are filled, canceled, or changed.

DPP must access the databases updated by APP together with those containing (1) process durations and precedence relations for each product to be produced and (2) detailed information on process utilization. The former is typically part of the process plan. The latter includes uptime, planned downtime, and any other restrictions on availability. The DPP uses this data to update release dates, priorities, and due dates for each job issued to the shop floor and requested availability times for any required but still outstanding inventory. The algorithms, and heuristics used in the planning function can be effectively replicated using one or more knowledge based systems.

2.6 Production Scheduling

Production scheduling develops detailed (usually daily) schedules of the operations required to complete the jobs issued by the DPP. These operations are then assigned to the various processes together with their anticipated start and finish times. Due date performance may be but one

of several criteria to be considered in establishing the sequence of activities. There may also be a several different algorithms which are use to generate new schedules. Once a production schedule has been generated, it is necessary to coordinate activities at each process to ensure that the schedule is met. This inter-process coordination requires continuous monitoring of the feedback from process supervisors.

Input data consists of the current schedule, the status and maintenance schedule for all machines, due dates and routing, and optimization criteria, and compromise strategies to be employed in making tradeoffs among the criteria. The production scheduler is responsible for maintaining an accurate schedule of activities at all machines on the shop floor. That schedule must be updated whenever (1) a new job list is received, (2) an existing job is canceled, finished, or given a priority update, and (3) a process experiences an unexpected delay.

2.7 Process Supervision

Each process has a supervisor who has two responsibilities. First, it implements the precise instructions from the process plan for every assigned operation. Second, it monitors the process during its execution of that operation to verify conformity to those instructions. Monitoring is typically sensor-based and allows the supervisor to detect changes in the processing environment. It can compensate for minor changes without substantial deviations from the original instructions. Major problems often force it to wait for a new set of instructions from process planning and from the Interprocess Coordinator (IPC) before completing the assigned task.

To do this, the supervisor must access information such as, the list of assigned jobs and scheduled start and finish times, their associated process plans, Numerical Control (NC) code or other equipment level programs, part description data, tool data, and fixture data. After each job has been completed, the supervisor must update the job database indicating the exact operations performed, the total processing time in the scheduling database, and the equipment and tool usage in their respective databases.

Production scheduling cannot rely completely on a priori information about machine operations because these operations are not predictable enough to determine entire factory schedules. To resolve this problem, an intelligent machine tool must close the factory wide feedback loop and start to feed information up through several levels of process supervisors. One example of knowledge-based support for process supervision is a Cell Management Language (CML) developed by Bourne [BOU86] to link complex systems of multivendor equipment together. CML was designed as a rule-based language which operates on a complete and ever-changing model of a flexible manufacturing cell. One of CML's primary objectives is to dynamically reschedule the machines in a cell; at each step in the process, it considers the current status of each machine in the cell and determines what to do next based on this complete knowledge of the cell's current situation.

Ideally, the cell supervisor would interact with intelligent controllers for each machine tool. Wright and Bourne [WRI88] provide the following definition:

The intelligent machine tool is defined by comparison with an intelligent human machinist. A higher level scheduler can rely on both of them in the same way. A given input leads to an expected output. Or, the intelligence reports back that the input is beyond the scope of the current system. We must therefore acknowledge that the degree of intelligence can be gauged by the complexity of the input and/or the difficulty of ad hoc in-process problems that get solved during a successful operation. Our unattended, fully matured intelligent machine tool will be able to manufacture accurate aerospace components and "get a good part right the first time."

2.8 Quality Assurance

Quality Assurance (QA) is divided into two major functions. First, it verifies that the output from each process meets the specifications prepared by engineering. These checks are the result of both on-line and off-line inspections. Whenever errors are detected, this information is used to correct problems in the designs, the process plans, and the processes themselves. Second, QA keeps historical records which can be used to improve the quality of all phases of the manufacturing system. In some cases, these records take the form of statistical studies which are used to track past and predict future equipment performance. These studies help guide decisions regarding machine maintenance and tool replacement. In other cases, information is archived on each product. This includes CAD designs, process plans, inspection and machining procedures, and other materials used in the fabrication of the that particular product. This helps guide decisions regarding that product the next time it is manufactured.

Quality Assurance (QA) tracks both the short term and long term quality of all manufacturing operations and the products they produce. Short-term QA requires access to inspection plans, usage charts, and planned maintenance schedules. Once the product inspection has been completed, the product history and scheduling databases must also be updated. Usage charts must be updated to indicate the total time every piece of equipment was used in the fabrication and inspection of each product. Long- term QA is achieved through updates to all historical and maintenance databases.

3 Need for distributed knowledge based systems to support CIM

In the preceding section we discussed several types of information required to carry out CIM functions. There is an ongoing effort to automate some of these functions such as design, process planning and scheduling. This has led to the development of expert systems that capture the heuristics, rules and algorithms used by humans to perform one or more CIM functions. These expert systems (also referred to as knowledge based systems) communicate with databases to assist in functions such as production planning, and scheduling. Several cooperating systems of this kind have emerged recently [KEL86]. Such systems consist of a database and a knowledge base each of which can function independently, and yet communicate with each other whenever necessary [KER88]. Structures such as semantic networks, production rules, frames, and logic are used to represent the *knowledge* needed by these systems. Data required to perform the functions is stored and accessed using database technology.

To support specific functions such as CAD/CAM several researchers have developed new data models based on traditional ones such as the relational model [BAT88; SU86]. Several researchers have also proposed the use of distributed technology to support CIM [THO89; DAV87]. While this is necessary, our contention is that it is equally important to integrate not only multiple databases but also knowledge bases to support the various CIM functions.

CIM data is used and generated by a variety of manufacturing functions running on a collection of heterogeneous computer systems. Individual systems will have a wide range of data access and data sharing capabilities. Some may have only file transfer mechanisms, while others may have sophisticated database management and/or knowledge management software. As indicated in the previous section, manufacturing data is of various different types. There are product catalogs, containing megabytes of information, which may be updated once or twice a year. This information contains text, numbers, graphics, and 3-D images. There are part models and process plans, which may contain several kilobytes of complex interrelated data, which are accessed and updated by many different users, several times each month [LOR83]. There is equipment status data, which may be only a few hundred bytes of data, but which must be updated several times a minute. In addition, sophisticated algorithms, simulation models, heuristics, and production rules are required. The latter types of information require the use of knowledge based systems. These systems allow for more complex ways of structuring, storing, and retrieving information. These include semantic networks, frames, and objects [WIN84]. These representations of information cannot be handled using the traditional data modeling techniques. Any system that supports CIM must provide facilities to [SU86]:

- capture complex data types
- capture temporal, procedural and positional relationships among objects
- perform operations on the objects
- model procedures, heuristics, algorithms to perform CIM functions

Since database technology alone is not sufficient to provide these facilities, we propose the integration of knowledge bases and databases in a distributed environment to coordinate CIM functions.

4 An approach for integrating data and knowledge for CIM

In this section we present an architecture that will facilitate the implementation of the CIM functions described earlier. This architecture provides support for integrating and sharing knowledge and data within a distributed environment [CAR89]. The architecture is a result of the synthesis of research in several different areas, such as, Distributed Database Systems [CER84], and, Distributed Problem Solving [DUR87; GAS87; SRI87]. It supports decision making in a distributed environment, using a network of knowledge and databases with coordinated communication among them. The idea of cooperative action is supported by this architecture. When

cooperative decisions are required, each knowledge based system accesses multiple sources of knowledge and/or data.

4.1 Distributed Knowledge Based Systems

There are several clear benefits to be derived from Distributed Knowledge-Based Systems (DKBS):

- Modeling real-world knowledge that has natural spatial or semantic separation
- Providing a modular architecture for large AI systems
- Integrating existing, heterogeneous knowledge-based systems
- Interconnecting multiple knowledge-based systems to solve a problem in which the individual systems have incomplete knowledge of the whole, but collectively, they can develop a solution

Huhns [HUH87] suggests that distributed artificial intelligence provides the next step beyond current expert systems by building a separate system for each problem domain, based on the ability of each expert, then making these systems cooperate. This modular approach would facilitate knowledge acquisition by finding experts in narrow domains and building separate systems around their individual expertise.

Figure 3 illustrates a distributed knowledge based system (DKBS) to support CIM functions.

The DKBS consists of several knowledge based systems (KBSs) that are logically and/or physically distinct from one another. A top-level Distributed Knowledge Based Management System (DKBMS), for the entire CIM system, integrates and manages all the knowledge based systems. Each individual KBS may support one or more functions outlined in section 2. These KBSs may be managed within a hierarchy of logically related DKBMSs. For instance, there may be three individual KBSs to decide on detailed production planning for three different products. These are all coordinated by the Planning DKBMS. A fourth KBS responsible for inter-process coordination in Production scheduling will need to coordinate its decision with the three KBSs for detailed planning. Any changes in the production schedules may affect future production planning decisions made by each of the three KBSs. Schedule-dependent goals from the planning KBSs would be routed by the Planning DKBMS to the Scheduling DKBMS, via the Planning & Materials DKBMS. The KBS for inter-process coordination may communicate with each of the manufacturing cell KBSs via the Scheduling DKBMS independent of these other goal resolutions. The global CIM DKBMS does not enter into these communications at all. A more detailed example is provided later in this section.

Each KBS, as shown in the exploded view in Figure 3, consists of an inference engine and one or more knowledge bases and databases. A knowledge base management system (KBMS) provides a mechanism and syntax to define and access the knowledge bases and databases for each KBS (see Brodie, et. al. [BRO86]). The DKBMS (described later) is capable of coordinating the activities of several KBSs, whenever cooperative action is required. A KBS is capable of independent activity, and does not need to be aware of the existence of other KBSs. Each KBS

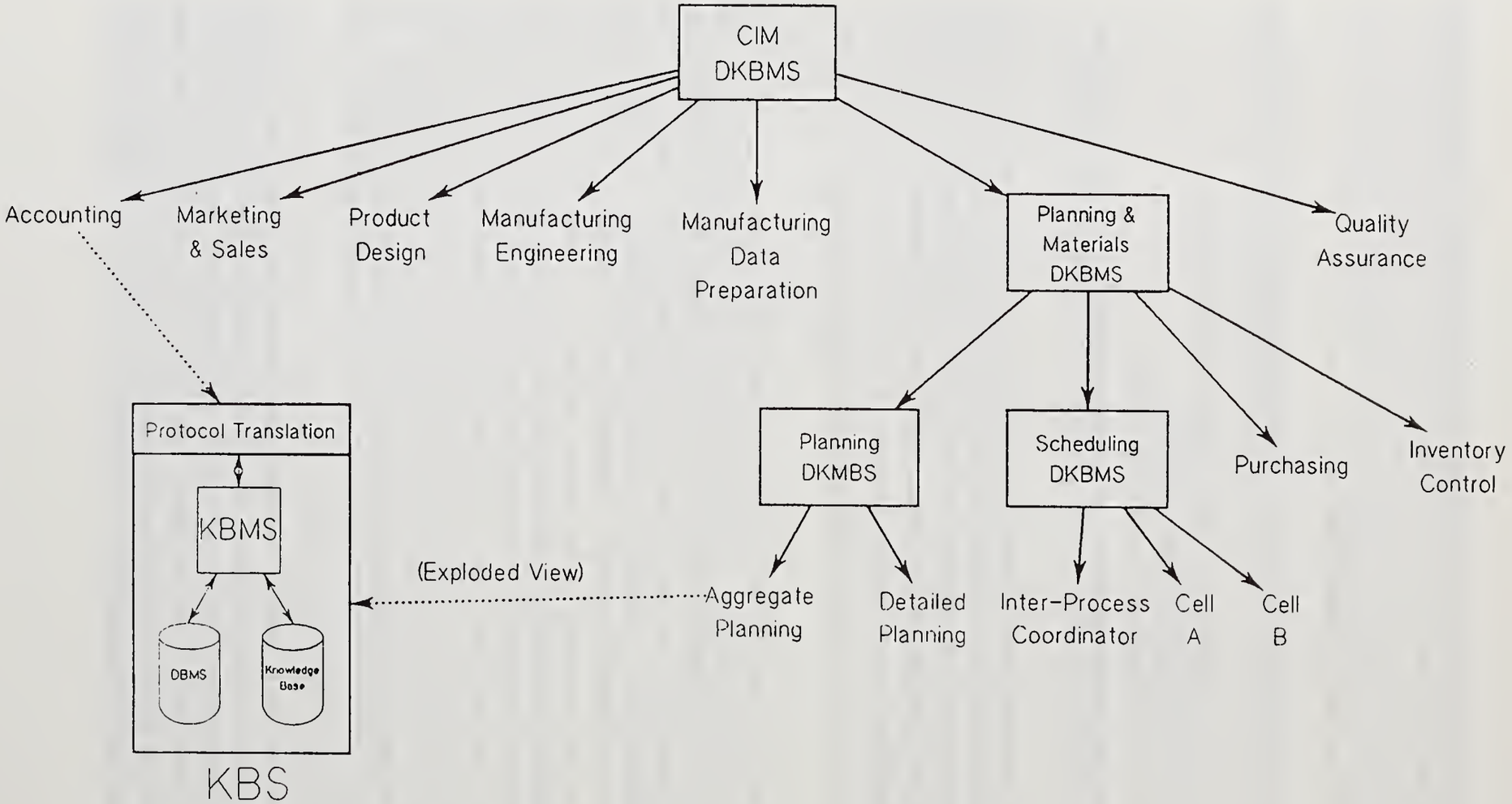


Figure 3: Distributed Knowledge Based System for CIM

provides a user interface for human users. Users can consult directly with the DKBMS or with a KBS which, as necessary, forwards general goals to its DKBMS. The DKBMS passes messages to relevant KBSs within its subhierarchy, or to its parent DKBMS when cooperative action is required. The DKBMS receives the results, consolidates the solutions and passes them on to the end-user. The objective of the DKBMS is to decompose global goals into one or more sub-goals, pass these on to one or more KBSs, to take advantage of parallelism and domain modularity, while minimizing communication between global and local levels of the architecture.

4.2 Distributed Knowledge Based Management System

Figure 4 illustrates components of a Distributed Knowledge Based Management System.

The DKBMS is composed of two components: a Global Knowledge Base (GKB) and a Communication/Mapping Module. The global knowledge base is further divided into meta-knowledge about the DKBMS's domain and a Knowledge Cluster Dictionary. The GKB contains information that defines the scope of each individual KBS, i.e. the types of inferences that each KBS is capable of making. For instance, the Planning DKBMS in Figure 4 should be aware of multiple KBSs that generate detailed production plans. It should be able to coordinate the communication and consensus among these KBSs.

The GKB contains knowledge useful for decomposing global goals into subgoals, for resolving conflicts among individual KBSs, for assembling results of individual responses from KBSs, and for recognizing when an inference has been completed. Subgoals are grouped into clusters which are in turn assigned to one or more local KBSs. This information that relates subgoals to cluster and KBSs is contained in the Knowledge Cluster Dictionary. By assigning a knowledge cluster to a local KBS the cluster subgoals are implicitly assigned to that KBS. Therefore the KBS is expected to contain the knowledge that will be used to resolve the subgoals of the cluster. It is possible to associate a knowledge cluster with more than one local KBS to achieve reliability through redundancy and to provide multiple knowledge sources in case a KBS is unable to resolve the subgoal. The assignment of KBSs to GKB subgoals (via the clusters) may be nondeterministic in some cases.

The Communication/Mapping module provides the communication link between the DKBMS and each individual local KBS. Local KBSs communicate with the KBMS using a standard protocol. This communication may either be for sending results of tasks that were passed down to them by the GKB or queries for resolving inferences that require the services of other Local KBSs. Sometimes it may be necessary to map the inferences from one representation to another. The latter step will be necessary if heterogeneous knowledge representations are supported.

4.3 Example of a CIM function using DKBS

A simple example will serve to illustrate how the DKBS architecture can be used to coordinate and provide integrated support for the various CIM functions. Let us assume that the system is

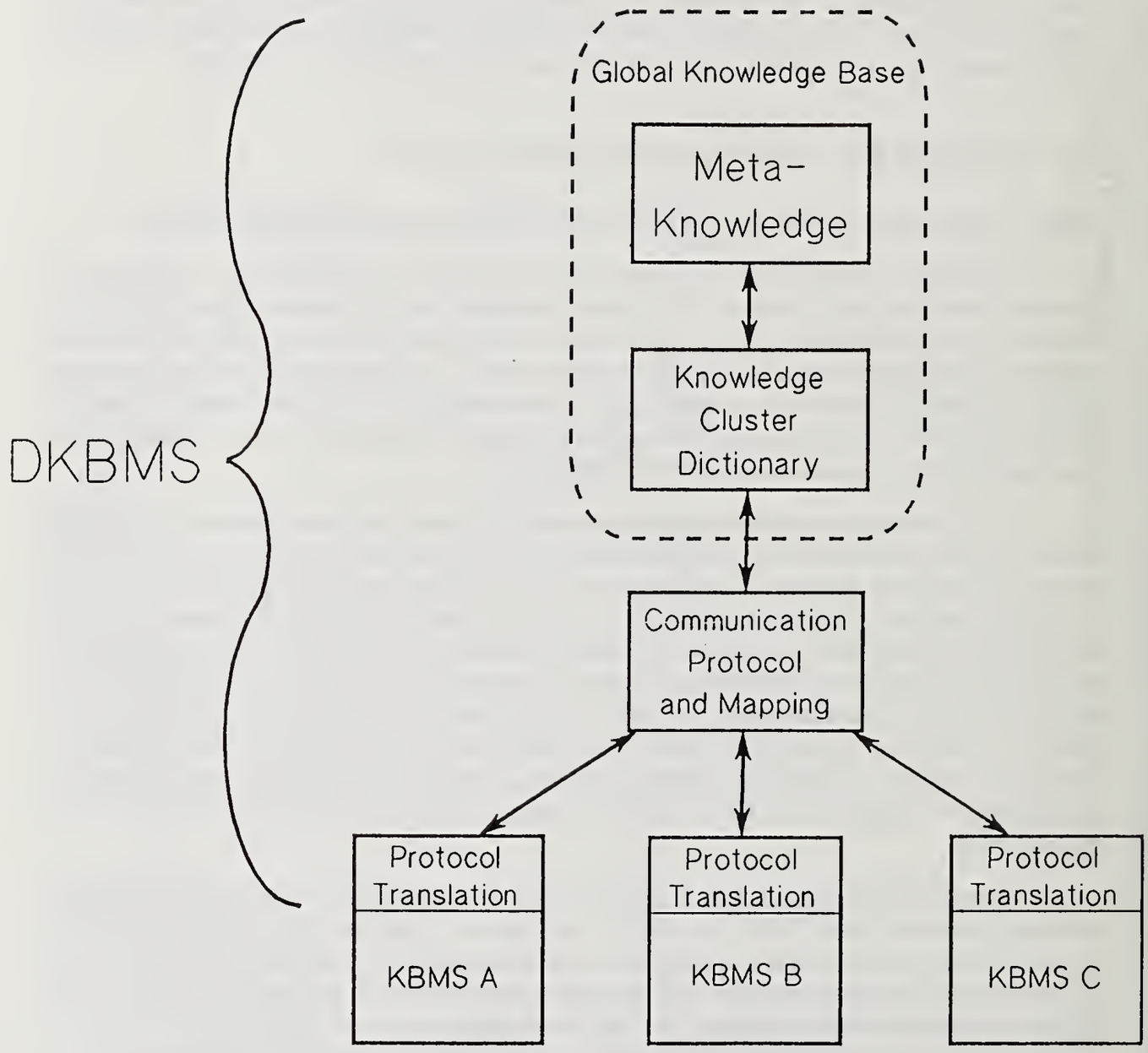


Figure 4: Distributed Knowledge Based Management System

being used to produce an overall production plan for the next 6 months. The following steps may be executed in satisfying this goal (See Figure 3).

1. A human planner would access the Planning DKBMS and input the planning horizon into the system.

2. The Planning DKBMS would decompose this goal of producing an overall production plan into several sub goals some of which may need to be performed in sequential order. For instance the first sub goal may be to generate forecasted demand for the various products. This sub goal may be delegated as a goal to one or more Marketing and Sales KBSs (via the CIM DKBMS). Another sub goal generated by the Planning DKBMS may be to generate an aggregate production plan for each product based on their demands. This sub goal however, can only be executed after the first sub goal has been processed. A third sub goal may be to produce detailed production plans for each time period.

3. The Planning DKBMS would assemble the results from the first sub goal, and pass on the relevant information to the Aggregate Planning (AP) KBS and instruct the latter to process the second subgoal.

4. The AP KBMS would use its inferencing capabilities to determine if it has all the information required to produce an Aggregate Production Plan. This information would include items such as, current and forecasted demand, production quotas, inventory control policies, and shop floor capacities.

5. If this information is not available to the AP KBS, a request will be communicated to the Planning and Materials DKBMS to generate the information.

6. The information required by the AP KBS will be gathered by the Planning and Materials DKBMS and communicated to the former, which will then generate an Aggregate plan.

7. This plan would then be communicated to the Planning DKBMS which in turn would assign the goal of producing detailed plans to the Detailed (DP) Planning KBS.

8. Several iterations of the steps outlined above may be required to generate suitable Aggregate and Detailed plans. Each of these iterations would be controlled by the Planning DKBMS.

4.4 An Object Oriented Design for components of the DKBS

Components of a DKBS can be implemented using the object-oriented paradigm (OOP). OOP provides an effective environment for modeling objects and communication between objects in a distributed system. OOP supports a frame-based knowledge representation in the GKB, supports heterogeneous knowledge representation of individual knowledge and databases, and facilitates a message passing paradigm for the communication protocol between components of the DKBS.

An OOP design defines *object classes* with *instance variables* (properties) and *methods* (a method is a procedure activated by a message sent to an object) [WEG86]. Principles of generalization, aggregation, classification and inheritance are also used by the OOP. An object is a specific instance of a class, and the object encapsulates all variables and methods defined (local or inherited) for that class.

Using this paradigm, we define several of the fundamental object classes in our design for a Planning & Materials (P & M) DKBMS (See figures 3 and 5).

Single instances of the DKBMS, GKB, and ClusterDict classes are created to manage all aspects of this particular DKBMS object for P & M. As discussed in a previous section, the GKB contains meta-knowledge which is used to control the inferencing for communication and consensus in this DKBMS. This knowledge is represented as frames (instances of class Frame) in our diagram. Part of this meta-knowledge may be represented as production rules which is organized within the frame structures. Finally, Cluster objects are used to relate subgoals (premises of production rules) to relevant KBSs. One cluster object may point to several KBS objects.

In the lower part of Figure 5, the object *Plan* refers to other objects such as planners (person(s) responsible for this plan), products, the planning horizon, and a forecast. Each *Person* object references additional objects which describe the opinions (instances of the class Assertion) and heuistics (instances of Rule) which define knowledge used by specific experts. Note that these Person objects are general and may be used in many other aspects of the local knowledge base. Rule premises may include reference to values of Product instances. An inference engine must be implemented to derive conclusions based upon the opinions and heuistics held by one or more planners. Recall that goals (in the action part of a rule) which cannot be resolved within a local KBS are then sent to the DKBMS for the GKB to resolve through communication with other KBSs.

Some of the objects defined in the lower part of this figure may be used in the implementation of the P & M DKBMS, as well as in one or more local KBSs. We present these objects in Figure 5 as an example of one small part of the DKBS for CIM. Although an object oriented design has been specified for the local KBSs, our proposed architecture has the capability to support other knowledge representations as well.

5 Research Issues in designing DKBS for CIM

Researchers at National Institute of Standards and Technology and elsewhere have addressed several issues dealing with the design of distributed databases to support CIM [JOH84; LIB88; MA84]. They are pursuing a hybrid architecture for the heterogeneous CIM environment. Some functions are performed at every node within the system. These include manipulating local data, translating queries and data representations into and out of local form, and providing interprocess and network communications. Distributed management services are assigned to selected sites and a unique master site ultimately resolves global dictionary changes and update conflicts detected

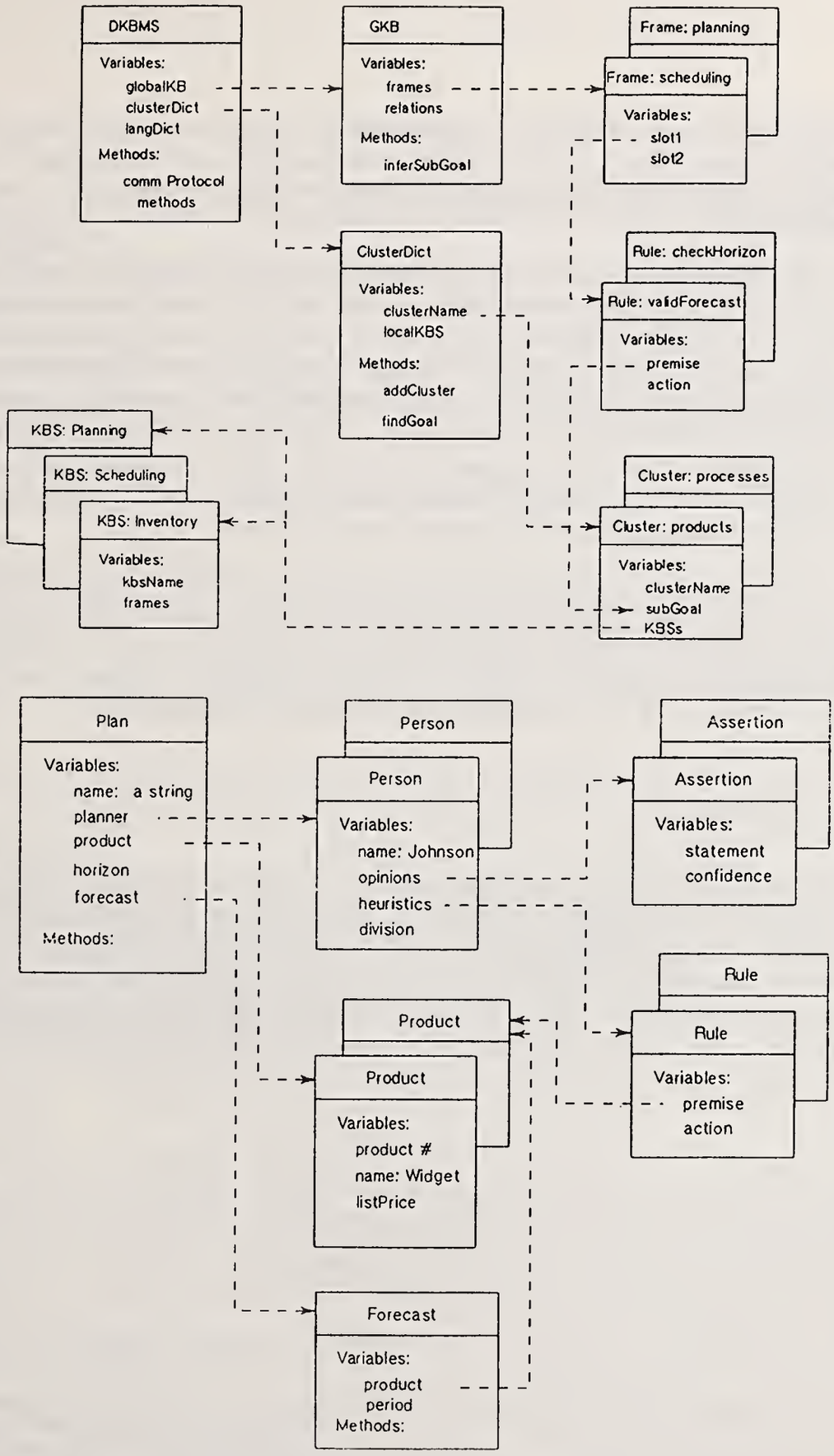


Figure 5: **Objects for Planning and Materials DKBMS**

by the selected sites. While this research is useful, there are limitations which prohibit it from being a viable solution to all of the problems inherent in the CIM environment. As noted in this paper, it necessary to integrate not only databases but also knowledge bases into the system to support CIM.

The contribution of the research described in this paper is the development of an architecture for distributed knowledge based systems to support CIM. Several research issues need to be resolved in designing DKBS for CIM. Design of each individual KBS within the DKBS and definition of the communication protocols between them are two very significant areas for research. The design of the GKB needs to be examined in detail. The process of decomposing goals into sub-goals and assigning to knowledge clusters will have a significant impact on the amount of communication in the system. Approaches such as the blackboard control architectures are being explored by researchers in several domains [HAY85]. Considerable investigation will be required to determine if this architecture is suitable to support CIM.

In our architecture, the meta knowledge required to manage the KBSs has been approached using the Object Oriented paradigm. However, alternative knowledge representations need to be examined. Another problem that needs to be examined is that of conflict resolution and consensus building among KBSs. The location of KBSs and the DKBMS itself is a major research issue. The DKBMS may have to be replicated at several sites in the system.

6 Summary

Efforts are being in many plants to use advanced computer technology to automate and integrate all manufacturing functions. This transformation to a CIM environment has revealed several problems in the design and real-time control of data management systems for CIM. In this paper, we have described both the major manufacturing functions themselves and the data required to carry out those functions. We have discussed an approach for integrating distributed knowledge based systems to support CIM functions. Several research issues for further investigation have been raised.

References

BAT88 Batory, D.S., Leung, T.Y., and Wise, T.E. "Implementation Concepts for an Extensible Data Model and Data Language", *ACM Transactions on Database Systems*, Vol. 13, No. 3, Sept. 1988, pp. 231-262.

BOU86 Bourne, D. A., "CML: A Meta-Interpreter for Manufacturing," *AI Magazine*, Vol 7, No 4, 1986, pp. 86-96.

BRO86 Brodie, Mylopoulos, and Schmidt (editors), *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, Springer-Verlag, June 1986.

CAR89 Carlson, D., and Ram, S. "An Object Oriented Design for Distributed Knowledge Based Systems", *Proceedings of the 22nd Hawaii International Conference on System Sciences*, Kona, HI, Jan 1989, pp. 55-63.

CER84 Ceri, S. and Pelagatti, G., "Distributed Databases: Principles and Systems", *McGraw-Hill Book Company*, 1221 Avenue of the Americas, New York, NY 10020, 1984.

CHA85 Chang, T., and Wysk, R., "An introduction to automated process planning systems", *Prentice-Hall*, Englewood Cliffs, NJ, 1985.

DAV89 Davis, W., and Jones, A. "A Functional Approach to Designing an Architecture for CIM", *IEEE Transactions on Systems Man and Cybernetics*, Special Issue on Manufacturing Systems, forthcoming, 1989.

DAV87 Davis, W. and Ram, S. "Design of Distributed Databases for an Automated Manufacturing Facility", *Proceedings of the ASME Computer in Engineering Conference*, New York, Aug. 1987, pp. 23-29.

DUR87 Durfee, E., Lesser, V., and Corkill, D. "Cooperation through Communication in a Distributed Problem Network", in *Distributed Artificial Intelligence*, M Huhns (ed.) Pitman, 1987.

GAS87 Gasser, L. "Report on the 1985 Workshop on Distributed AI", *AI Magazine*, Vol. 8, No. 2, Summer, 1987.

JOH84 Johnson, H., Baum, L., and Beaudet, R., "IPAD Distributed Database Management Facility-IDF: Architecture Specification", *Boeing Computer Services Company*, 1984.

JON86 Jones, A., and Mclean, C. "A Proposed Hierarchical Control Model for Automated Manufacturing Systems", *Journal of Manufacturing Systems*, Vol. 5, No. 1, 1986, pp. 15-25.

HAY85 Hayes-Roth, B. "A Blackboard Architecture for Control", *Artificial Intelligence - An International Journal*, 1985, Vol. 26, pp. 251-321.

HUH87 Huhns, M. (editor), *Distributed Artificial Intelligence*, Pitman, 1987.

KEL86 Kellog, C. "From Data Management to Knowledge Management", *IEEE Computer*, Jan. 1986, pp. 75-84.

KER88 Kerschberg, L. "Expert Database Systems", *IEEE Expert*, Vol. 3, No. 4, Winter 1988, pp. 50.

LOR83 Lorie, R.A., and W. Plouffe, "Complex Objects and their use in Design Transactions", *Proceedings of ACM SIGMOD Conference on Engineering Design Applications*, San Jose, CA, 1983, pp. 25-33.

LIB88 Libes, D. and Barkmeyer, E., "The Integrated Manufacturing Data Administration System (IMDAS) - An Overview", *International Journal of Computer Integrated Manufacturing*, Vol. 1, No. 1, 44-49, 1988.

MA84 Ma, R., "A Model to Solve Timing-Critical Problems in Distributed Computer Systems", *IEEE Computer*, Jan., 1984, pp. 62-68.

MAD87 Madison, D.E. and T. Wu. "An Expert System Interface and Data Requirements for the Integrated Product Design and Manufacturing Process", *Proceedings, Third International Conference on Data Engineering*, Feb. 1987, pp. 610-618.

SRI87 Sridharan, N.S. "Report on the 1986 Workshop on Distributed AI", *AI magazine*, Vol. 8, No. 3, Fall 1987, pp. 75-85.

SU86 Su, S., "Modeling Integrated Manufacturing Data With SAM*", *IEEE Computer*, Jan., 1986, pp. 34-49.

THO89 Thomas, G. et. al. "Heterogeneous Distributed Data Systems for Production Use", *ACM Computing Surveys*, Special Issue on Distributed Database Systems, forthcoming, 1990.

WEG86 Wegner, P. "Perspectives on Object-Oriented Programming", *Technical Report No. CS-86-25*, Brown University, Dept. of Computer Science, Dec. 1986.

WIN84 Winston, P., "Artificial Intelligence", *Addison-Wesley*, Reading, Massachusetts, 1984.

WRI88 Wright, P., and Bourne, D., "Manufacturing Intelligence," *Addison-Wesley Publishing Company*, Reading, MA, 1988.

UNIFORM DATAFLOW SOFTWARE SYSTEM FOR GLOBAL CIM APPLICATIONS

HUGH SPARKS

Abstract

At present, integration architectures for both local and global automation problems suffer from a variety of shortcomings. These include a strong dependency on particular hardware platforms and lack of a uniform conceptual framework for software development at multiple system levels. This paper describes a unified software architecture based on object-oriented programming techniques. This architecture is supported by two programming languages, HOSE and Alltalk. HOSE is a graphical dataflow language for parallel processors that features hierarchical abstraction. Alltalk is a more traditional object oriented language used to implement iconic control panels for industrial applications. These languages are portable to a variety of hardware platforms that support both parallel processing and a graphics workstation. The paper describes these languages and our experience with several applications in robotics and process control.

1. Introduction

In some respects, the state of the art in system integration methods is analogous to the condition of the housing industry. While the use of personal computers and inexpensive CAD software has revolutionized development of housing designs and the generation of archtectural prints, the actual construction process is still performed in a labor intensive manner with few standard interchangeable "housing objects." Automated and modular construction methods that have been demonstrated and they are capable of producing high quality results at reduced costs, but these techniques have achieved only limited acceptance.

Similarly, CIM systems have been demonstrated that employ advanced software tools for design and simulation. Implementation, however, involves laborious coding of unique and mostly non-reusable software. A variety of higher level software concepts have been investigated to address these problems, but are not widely viewed as practical in the industrial environment, particularly for the implementation of high performance real-time control systems.

A number of recent publications have touched on issues involved in the development of a more unified approach to software and hardware architectures for CIM. Naylor and Volz [NAY88] have identified the need for portable and reusable software which supports multiple levels of abstraction in a distributed hardware environment. Camarinha-Matos and Steiger-Garcao have considered the need for a unified conceptual architecture in CIM information management [CAM87].

We have developed two specialized object-oriented programming tools: HOSE, a hierarchical dataflow language for the description and implementation of industrial control systems; and Alltalk, an interactive programming enviroment for building animated graphical control panels. We have integrated these tools to implement multiple levels of automation software, including the more performance-critical aspects of direct digital control systems and sensor fusion processing.

Portions of this software environment are being used to develop vison-directed robotic systems for manufacturing, material testing, and automated inspection at our company. This paper describes these tools and presents examples of our current research and applications.

2. The HOSE programming language

Signal flow diagrams, a traditional tool for describing control and signal processing systems, have been used as a model for a new graphical programming language called HOSE. This language extends the signal flow concept in several ways: signals are generalized to include user defined data types; object-oriented programming concepts are integrated to support hierarchical abstraction; and parallel processing hardware is supported to enhance performance.

2.1 Key concepts and features

HOSE was developed to support very rapid implementation of direct digital control systems and sensor fusion applications. HOSE combines several key software technologies towards this end: parallel processing, hierarchical abstraction, object-oriented programming, and an iconic human interface for interactive programming. Details of these features and concepts are discussed in the following sections along with a brief introduction to the HOSE language. Further details on HOSE programming are available in [HOS1] and [HOS2].

2.1.1 Sequential vs. parallel programming

Conventional languages such as C or Pascal are sequential and procedural. They are based on an auditory-serial mode of thought. A HOSE program is designed and understood through diagrams. As following examples will demonstrate, this visual mode of perception makes it easier to describe and understand parallel processing systems.

Figures 1 and 2 contrast two software descriptions for a hypothetical robot welding task. The example system contains a robot arm that manipulates a laser welding device. The robot must weld parts together while tracking a seam using a video camera.

```
Locate beginning of the seam
Start the laser
WHILE seam is visible DO
        Process camera image
        Adjust trajectory
        Adjust laser focus
        Advance along the seam
        Adjust camera angle
END  WHILE
```

Figure 1. Sequential description of welding.

This program is typical of industrial software: many concurrent activities must be scheduled and coordinated by a sequential language algorithm. The statements before the WHILE loop express a step-by-step time sequence of events, but the statements inside the WHILE loop attempt to implement an essentially parallel operation. When multiple sequential programs execute in parallel, even less intuitive constructs must be introduced to synchronize their behavior. Such programs are notoriously difficult to understand and debug.

Figure 2 shows a signal flow diagram that describes some of the behavior implied by the statements contained in the WHILE loop of figure 1. Notice that essentially parallel aspects of the seam tracking problem are reflected in the topology of the diagram. Arrows that diverge from a common source terminal imply parallel information flow while arrows that converge on the same object imply synchronization. (Details of the synchronization mechanism in HOSE are presented in a later section.)

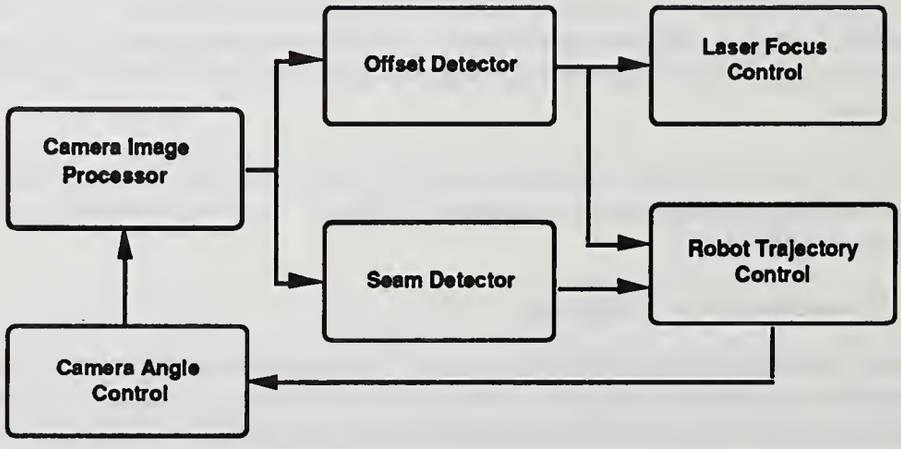


Figure 2. Parallel description of welding.

2.1.2 The dataflow concept

A HOSE program is, in effect, a diagram. Data in a HOSE program flows between processing units generically called *objects*. These objects are the basic processing elements in the language. Objects communicate with each other using signal streams that travel over directed connections.

HOSE diagrams, like electronic circuits, generally indicate continuous and simultaneous motion of data between the processing elements, in contrast to flow charts, which show the sequential flow of control in a step-by-step manner. Flow charts are graphical aids best suited to visualization of essentially sequential processes, while dataflow diagrams are best suited to visualization of continuous and concurrent processes.

2.1.3 Data types for signals

The data signals that flow over connections have associated data types derived from the Pascal programming language. Connections can transport real numbers, integers, Boolean values or user-defined data types such as vectors, strings and records.

2.1.4 Icons represent objects

Just as an electrical circuit has standard symbols for components, such as resistors and capacitors, every processing object in HOSE has an icon. The icon is a graphical representation of a computational object on the display screen. An icon may have terminals represented by short line segments or "legs" like the terminals of electronic devices on a circuit diagram. Terminals define the points where connections are attached to an object. Entry terminals define points where data flows into an object while result terminals define points where data flows out of an object. Each terminal has an associated data type which determines the kind of data it will generate or accept. The graphical editor prevents connections between terminals with incompatible types.

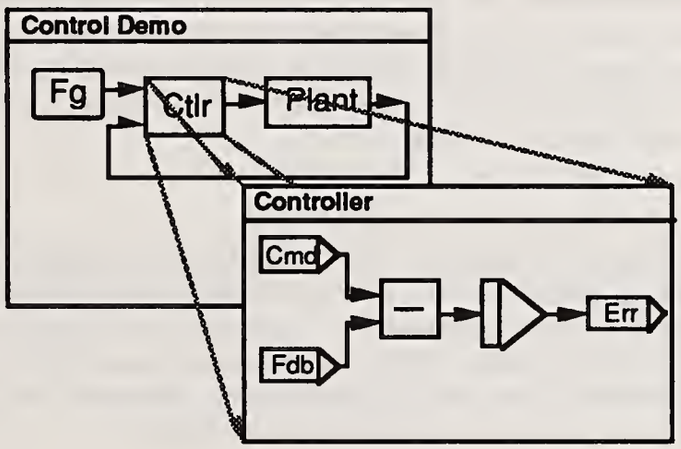


Figure 3. Hierarchical Diagrams.

2.1.5 Hierarchical diagrams

HOSE uses *hierarchical* diagrams to describe a system at different levels of detail. Figure 3 above illustrates a schematic view of this concept. (More detailed examples will follow.) Any object may be implemented by an internal network of other objects. The objects in HOSE organize processing into conceptual units in much the same way that procedures organize a program written in a sequential language.

The suitability of hierarchical systems for complex sensor integration, control, and intelligent automation is well accepted [ALB87]. A system can be designed using an arbitrary number of layers that hide details of their implementation. Object-oriented programming techniques for large scale system description is discussed in [OSS87].

2.1.6 The programming process

A HOSE program is written (or drawn) by creating particular objects and specifying dataflow connections between them. The HOSE interpreter displays the dataflow diagram using windows on a graphics workstation. The developer creates objects and specifies their interconnections using a pointing device (mouse) and special function keys.

2.1.7 Object oriented concepts

When a given diagram is drawn, it may be used as a template to define a new HOSE *class*. A class defines the internal structure of a diagram and the icon used for the exterior view. A class can be used to create multiple *instance* objects, all of which have the same icon and the same internal structure. Instances are distinguished by names assigned by the user when the object is created. The name of an object appears inside the icon's border. An analogy for the class-instance relationship occurs in electronics. An electrical engineer uses standard integrated circuits composed of more elementary circuit elements. In HOSE, this abstraction capability can be extended to any number of levels.

A six-axis robot, for example, requires six servo controllers, one for each joint. In HOSE, a servo controller class is defined by drawing a network that implements a single servo controller. This controller is then used to define a new class. By specifying this class, the programmer can create six instance objects. All six servo controller objects share the same code, although each has its own private internal state information. From the user's point of view, both code and data are encapsulated in the object.

2.1.8 Modeless environment

In HOSE, there are no modes that distinguish running vs. editing a diagram. New objects and connections may be added or removed and values can be introduced at any terminal at any level of the diagram. Such online modifications are extremely useful when debugging or developing a control system. Oscilloscopes can be connected or moved, filters can be added or retuned, all without stopping the running application. This process especially facilitates exploratory or rapid prototyping efforts.

2.1.9 System design and documentation

A HOSE dataflow program is usually developed top-down. The diagram is drawn by an incremental process of refinement. As the design process continues, the internal structure of each object is defined at lower levels. The HOSE system implements a type of software CAD where dataflow diagrams document the system architecture. As lower levels of the diagram are drawn, it becomes possible to execute parts of the HOSE program during the development process. Because diagrams *are* the program, there is always a correct correspondence between the graphical part of the design documentation and the real system.

2.1.10 Parallel processing

The ability to use multiple processors is essential for large applications with hundreds of dataflow objects. As processors are added to the system, HOSE can execute programs faster. Each processor contains a copy of a specialized real-time kernel optimized for concurrent dataflow. The processors communicate using shared memory. For very large processor arrays, alternative interprocessor communication systems for HOSE have been investigated [CHAT88]. In simulation applications, a copy of the HOSE kernel executes on the graphics workstation that displays the diagrams. In this case, no parallel processing hardware is required.

Numerical analysis applications are often well served by symmetric arrays of processors and simple Fortran style data structures. In contrast, many industrial automation and process control problems are "scruffy": they require the cooperation of multiple *diverse* parallel processes. Message flow between these processes requires specialized data types unique to the application problem. Some messages are asynchronous while others are periodic. The frequencies of periodic signals are also different for different subsystems in the application. HOSE is adapted to all these requirements.

2.1.11 Generalization of feedback control concepts

We believe that the extension of feedback control concepts to higher levels of a control system can result in more robust and adaptable application software. As an example, consider a manufacturing process as a hierarchical control problem. We want to manufacture a particular part described by CAD drawing. This drawing may be taken as the *command signal* for the manufacturing system. The manufacturing process hardware consists of machine tools that fabricate various features of the part and material transport systems between the tools. At the highest level of the system, the *feedback signal* is a stream of finished part descriptions. The high-level controller uses the feedback information to modify command streams it generates for sub-processes that fabricate various features of the part. These sub-processes may, in turn, use feedback loops with abstract data types as signals that describe their desired behavior. At the lowest levels of the system, real-valued and boolean-valued signal streams operate mechanical actuators and switches on machine tools, material transport devices and inspection systems.

In a more general case, we can view the manufacturing system as a general-purpose part fabrication facility. In this case, a stream of part descriptions enter the system as a command signal and the controller figures out how to build the part with the available facilities. For simple part prototypes such systems have already been implemented. The term *desktop manufacturing* has been coined to strike an analogy with the revolution already realized in desktop publishing. To extend the desktop manufacturing concept to multiple tools and processes in an industrial environment will require software with unprecedented flexibility and abstraction capacity at all levels of the system.

The extension of feedback control concepts to cell level and factory level automation and the associated difficulties are considered by [KUM87] and [CAM87].

Researchers at the University of Minnesota have developed and applied competitive feedback processes to control higher-level robot behavior. This work features autonomous mobile robots with embedded controls developed with HOSE. Representative examples of this work are discussed in [TAL89], [AND90].

The contribution of HOSE towards realizing this concept stems from its ability to support abstract data types in signal streams and the ability to create hierarchical concurrent control systems. The ability to operate on parallel processor networks allows a HOSE diagram to be scaled up in complexity to the capacity of the computing hardware.

2.1.12 Primary objects

The hierarchical definition of objects can be any number of levels deep. Eventually a level is reached where an object is not implemented by a diagram but instead by a sequential language program: This lowest-level object is called a *primary* object. Figure 4 illustrates a simple primary object that implements a proportional controller.

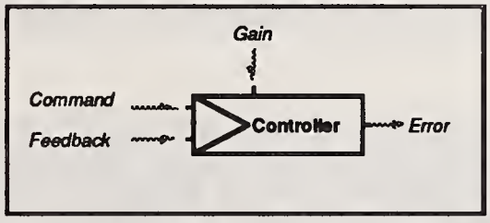


Figure 4. Icon for the proportional controller.

Figure 5 shows the source code that implements this primary class. When this module is compiled, the developer can create any number of Controller instances. Each instance will share the same code, but have its own internal state variables.

```
primary Controller ;

var  myGain : real ;

method setGain(entry gain : real) ;
begin
     myGain := gain
end  method

method doit(entry command, feedback:real;
            result  error:real  ) ;
begin
     error := myGain * (command - feedback)
end  method

initialize
     myGain := 1.0
end.
```

Figure 5. Source code for the proportional controller.

The source code for a primary object is written in a special superset of Pascal. In addition to the normal Pascal constructs, a primary class definition contains special *method* procedures and a set of instance variables. In the Controller primary definition above, there is one instance variable called *myGain* and two methods: *setGain* and *doit*. The *setGain* method has one entry called *gain*. The *doit* method has two entries: *command* and *feedback*. The *doit* method also has one result: *error*. The entry and result parameters for all methods in a primary class are associated with the dataflow terminals displayed on the primary object's icon.

Each processor in the system has a complete set of object code for the methods of every primary class. Each *instance* object of a primary class contains only a the set of instance variables defined by the class and some internal scheduling information. The instance data for all instance objects in a diagram is stored in memory shared by the parallel processors. It is consequently possible for any processor to execute any method in the context of any primary object.

2.1.13 Dataflow scheduling

The fundamental unit of computation in HOSE is the execution of a method in the context of a primary object. A method is ready to run when data samples have arrived for each of the entry parameters declared in the method declaration. For example in figure 5, the *doit* method will run when actual values for the command and feedback entry variables arrive over connections to the primary object terminals. The terminals of a primary object have the same names as the formal parameters to the internal methods.

The HOSE scheduler is a specialized real-time kernel that resides in each of the HOSE system processors. The scheduler executes methods that are ready to run and moves the result data along dataflow connections to other objects. A priority system can be used by developers to control the behavior of the scheduler. Because a method can adjust the priority of other methods, it is possible to implement most popular real-time scheduling algorithms.

2.1.14 Object libraries

Users may create their own libraries of primary or composite objects. We have developed an extensive library of objects to support the implementation of adaptive control systems. Some of the primary objects commonly used are mathematical operators (e.g., add, multiply, square root), waveform generators (e.g., sine, cubic interpolation, random), servo controllers (e.g., PID, adaptive pole-placement), control system design aids (e.g., root locus, spectrum analysis), statistical functions (e.g., autocorrelation, linear regression) and signal processing functions (e.g., digital filtering, adaptive LMS filtering, system identification).

2.2 Typical platform

HOSE has been implemented with high level languages and is consequently portable. Figure 6 shows a VAX workstation linked with multiple Motorola 68020 processors on the VME bus. A implementation also exists for the BBN Advanced Computers Butterfly™ parallel processor. The BBN implementation uses the Sun workstation for editing the dataflow diagrams.

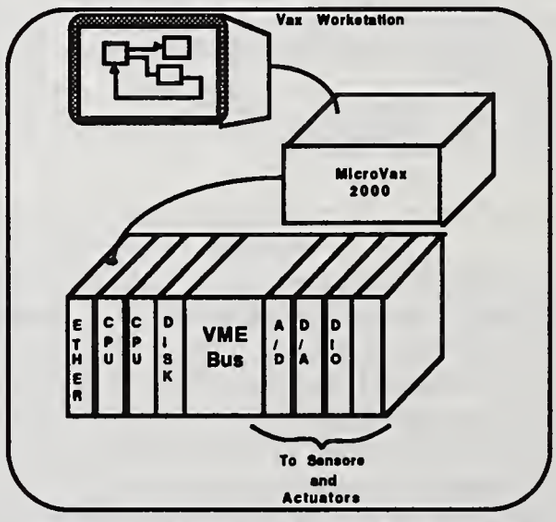


Figure 6. A typical HOSE platform.

2.3 Hose applications

2.3.1 Analog computer simulations

A HOSE program is developed and tested using a methodology similar to that used to work with analog computers. The standard HOSE system is supplied with classes for objects such as integrators, differentiators, summing amplifiers, and other signal processing components. The simulations found in analog computing or signal processing textbooks can be implemented in a very direct manner as HOSE diagrams.

2.3.2 Complex simulations

Large simulations are difficult to implement on analog computers because there is no means of structuring the "software" beyond the flat electrical network. In addition, analog computers have one only kind of signal, represented by voltages or currents in the machine.

HOSE supports type abstractions for diagram structure (classes) and for the data used to represent signals. Higher level parts of a diagram often use signals that consist of complex programmer-defined data structures. HOSE diagrams can be used for multiple system levels, in some cases up to the end-user interface. Future extensions of HOSE will include animated displays and iconic controls to further support end-user applications.

2.3.3 Sensor integration and machine control

Digital interface electronics for sensors and actuators can be added to the bus with the parallel processing hardware. Special classes describe objects that interface with the external world. For example, a digital to analog converter may be represented as a HOSE object with a single real valued entry terminal. When a times series of values flows into this object, it simply converts the values to an integer and deposits them in the converter's output register. Analog input and output objects may be combined with controller objects to build feedback control networks for any continuous process.

Similarly, digital input/output devices that control external electronic switches are represented by HOSE objects with Boolean valued terminals. These objects can be combined with other HOSE components that implement AND gates, OR gates and other logic devices. The resulting logic diagrams can describe and implement most discrete control functions, eliminating the need for conventional programmed logic controllers.

2.3.4 Simulations for control development

Engineers using HOSE for real-time control applications can also benefit from HOSE simulation capabilities. A simple simulation of the application's hardware can be constructed so the control system can be tested during development. When the application hardware becomes available, the simulation object is replaced by an object that controls the external analog/digital interface. The rest of the diagram remains the same.

2.3.5 Real time diagnostics

The diagnostic technician or developer can examine any part of a machine controller while the system is in operation. If necessary, objects can be connected, disconnected, created or destroyed. A number of graphical tools have been developed that allow developers to monitor system operations. For example, a digital oscilloscope object, similar in function to the hardware instrument, can be created and connected anywhere to observe time series waveforms in a workstation window. The oscilloscope allows the user to scale inputs, change the sweep time or trace color just like a physical oscilloscope does. Other diagnostic tools include a fast Fourier transform scope, digital displays and objects that transmit signals external applications.

2.3.6 Control system demonstrations

An important application of HOSE has been in the development of feedback control algorithms and sensor signal processing. The following examples demonstrate some of the HOSE objects used to implement these algorithms. The paper [SPAR90] presents more complex adaptive control and signal processing examples.

2.3.6.1 A feedback control demonstration

Figure 7 shows the top level HOSE program for a simple feedback control system. A function generator, *fg*, is connected to a controller, *pidctlr*, which is in turn connected to a plant simulation object, *plant*. In this case, the plant is a simulation of a one degree of freedom system we want to control. The plant response is connected back to the feedback input terminal on the controller object. The *scope* object is connected to display the function generator

output and the plant response for comparison.

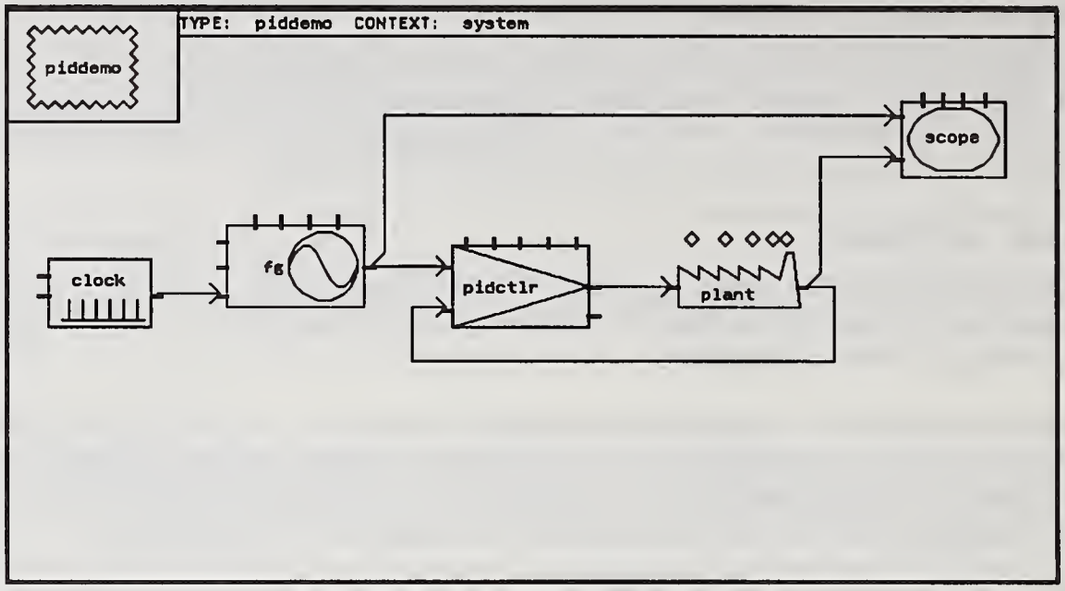


Figure 7. A simple control loop.

The function generator contains an internal queue of segment descriptors. Each segment can have a unique shape and duration. The external clock attached to the function generator drives the whole program. The clock object emits a stream of Boolean samples that trigger the function generator to emit the next real-valued sample of the current segment. When a segment is completed, the function generator moves on to the next segment. Additional segment descriptors can be added to the queue while the clock is running. If the queue becomes exhausted, the output from the function generator is held at the level of the last sample value.

Figure 8 shows the display screen of the two-channel *scope* object. A sequence of waveform segments from the function generator are plotted along with the actual plant response. The scope has terminals to allow adjustments similar to those found on the electronic technician's oscilloscope. The gain, offset, and time base can be adjusted while the scope is in operation. On a color workstation each trace may be assigned a different color.

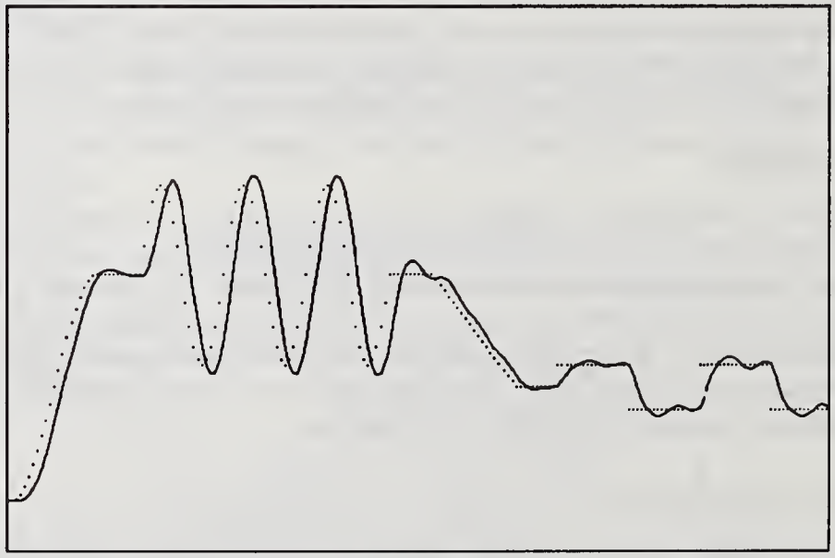


Figure 8. Command (dotted line) and Plant Response (solid line).

Figure 9 shows the implementation of the controller used in figure 7. To obtain this view, the operator selects the *pidctlr* icon on the workstation screen using a pointing device (mouse). By "clicking" the mouse button, the icon view expands into a full window that shows the internal parts of the controller.

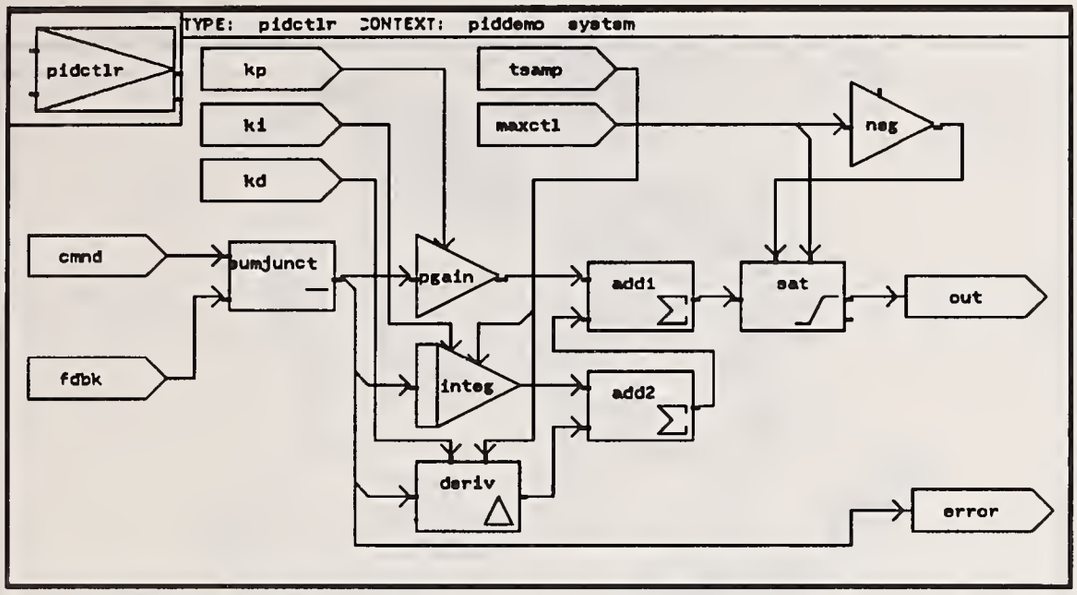


Figure 9. The proportional-integral-derivative controller.

2.3.6.2 Adaptive control and signal processing

HOSE has been used extensively for research and development of adaptive control technology. A toolkit of control and signal processing objects has been created including pole-placement controllers, least-mean-squares process estimators, dynamic displays for FFT spectra, and dynamic pole-zero plots for system identification. Additional control and signal processing applications are discussed in [SPAR90].

2.3.7 Robotics applications

A complete industrial robotics application requires the integration of multi-axis controls with sensors that detect parts and complex PLC (programmed logic controller) arrays to manage discrete logic aspects of the process. HOSE can be used to integrate feedback control, sensor fusion, and process level controls in a single software environment. The following examples will illustrate these techniques

2.3.7.1 A Six axis robot controller

The MTS A200 Robot has six joints driven by rotary hydraulic actuators. In our research program, the analog controller usually supplied with the robot was replaced by an all-digital controller implemented with HOSE. The top level diagram for this controller is shown in figure 10 below.

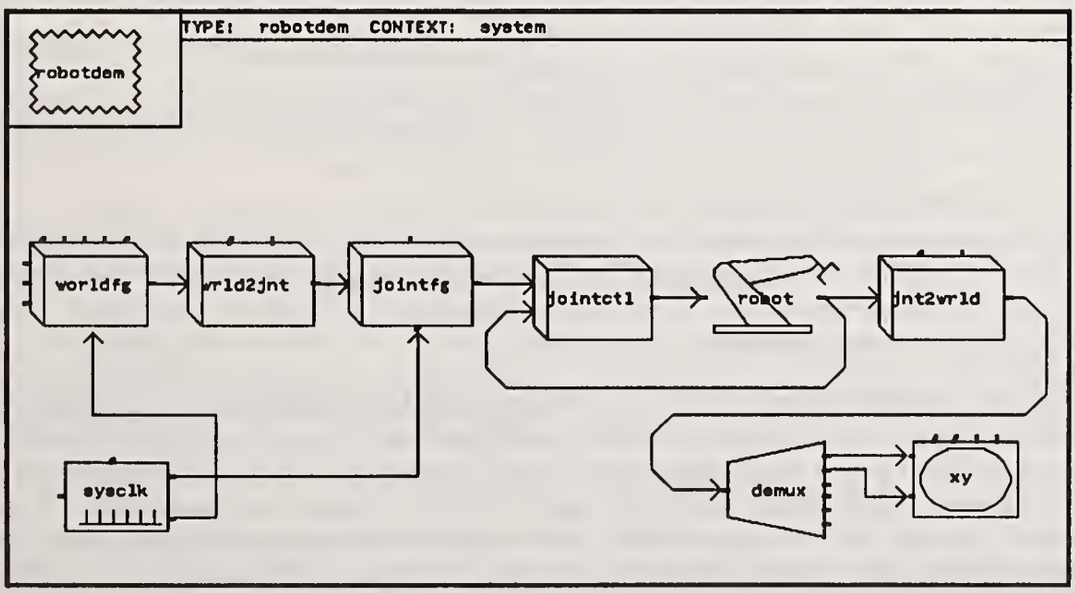


Figure 10. Six axis robot control system.

The entire HOSE diagram required two weeks to develop and test with the simulated robot shown in figure 11. The simulated robot object was then replaced by a robot interface object that contained digital-to-analog and analog-to-digital interface objects. The rest of the controller remained the same.

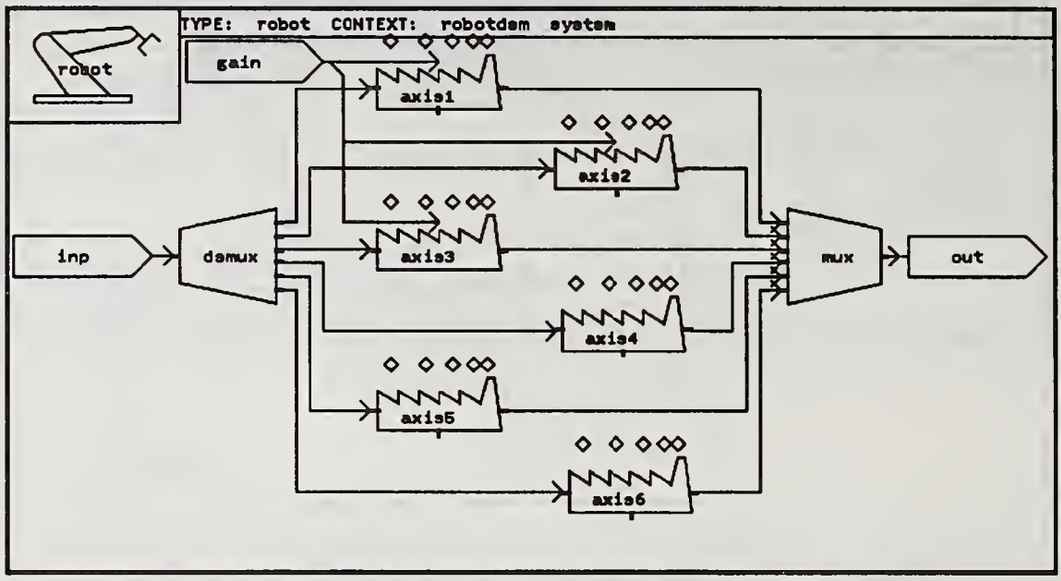


Figure 11. The robot simulator.

In figure 12 below, the *jointctlr* object has been opened into a window. The input terminals for command and feedback (labeled *cmnd* and *fdbk*) are vector-valued digital signals. For each time sample, an entire vector of real numbers travels over the connection from the joint function generator to the command entry, *cmnd*. Similarly, another vector arrives from the robot that contains the actual measured joint angles. This vector enters the feedback terminal, *fdbk*. The *demux* object breaks the vectors in to separate scale signals and distributes them to six instances of the PID controller discussed in the previous example. The output signals from each axis controller are then recombined (multiplexed) and the result is the vector-valued output of the joint controller.

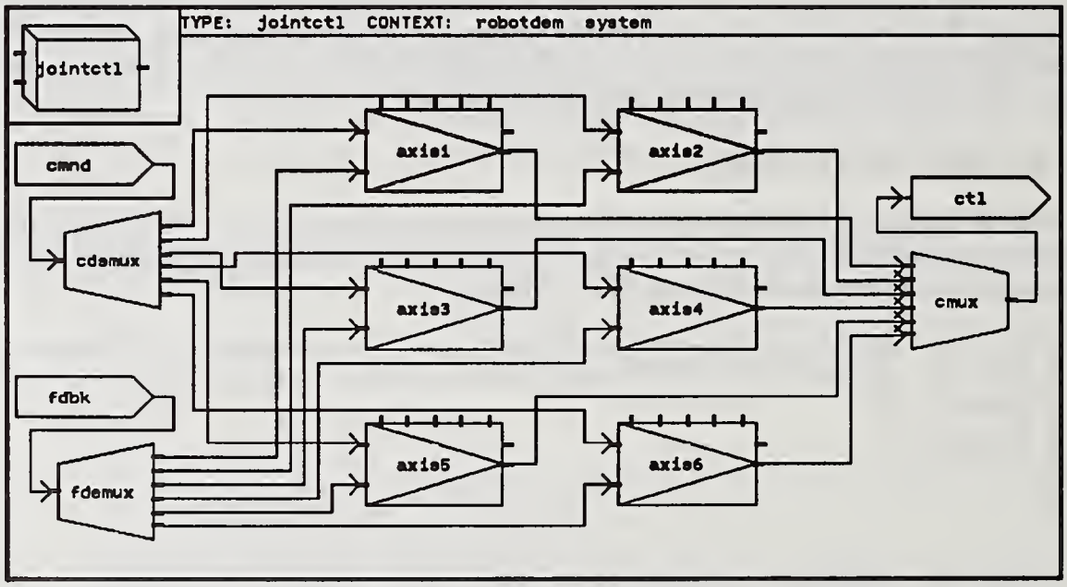


Figure 12. A six axis controller.

2.3.7.2 The LARS welding robot

The first large scale commercial application using HOSE was the Laser Articulated Robot System developed by MTS under contract to the Naval Research Laboratory and Penn State University. (Figure 13 below) The LARS system can track and weld a seam at 200 inches/minute using its vision system without preprogramming. The robot transports and focuses a 15,000 watt carbon dioxide laser. The HOSE diagram implements direct digital control of the six-axis robot, processes the vision system data, and coordinates an elaborate network of safety interlocks. The original implementation used 24 Motorola 68000 processors. A second LARS machine was implemented using only four 68030 processors. Additional details are presented in [CLEV87].

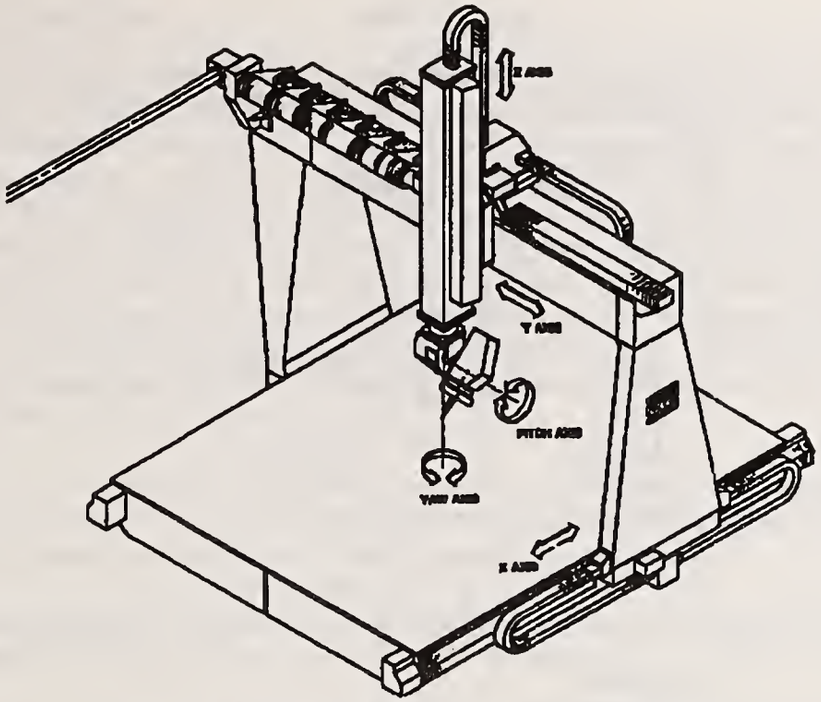


Figure 13. The LARS laser welding robot.

2.3.7.3 Intelligent robot inspection system (IRIS)

The IRIS system implements non-contact inspection of complex three-dimensional parts. A robot manipulator holds a laser offset gage that measures the distance from the gage head to the part surface. The manipulator also carries retro-reflectors for three laser beams that span the distance from the offset gage to a precision reference surface. These three beams are used in interferrometer cavities to provide very high resolution position information for the offset gage head.

The HOSE control system implements sensor fusion for the laser offset gage, the tracking system for the interferrometer beams and feedback control for the six axis robot. The system features *end point control*. The robot controller uses information from the interferometers to position the offset gage. The robot is only required to bring the offset gage within range of the part surface. Consequently, very accurate part measurements may be obtained without requiring a high precision robot. IRIS can provide measurement samples from the moving gage at 1000hz to an accuracy of .006 mm (.00025 inches) over a 1m x 1m x 1m (3'x3'x3') working envelope. Additional material on the IRIS system is presented in [CLEV87]

2.3.8 Application to material processing automation

A research project underway at MTS Systems Corporation [WHIT89] seeks to apply advanced control and sensor integration techniques to advanced material processing applications.

The *nanocrystal* formation application (currently under construction at MTS) will produce samples of metal alloys with extremely fine microstructure. Figure 14 below is a schematic of the experimental apparatus. A liquid nitrogen filled cold finger rotates inside a partially evacuated chamber. Metal vapor from multiple electric evaporators condenses on the cold finger in micro particles. An external actuator drives a scraper that removes the crystals from the cold finger. The crystals are compressed by another actuator to form the final specimen. Details of this process are discussed by [FRO89]. This example is only intended to illustrate how the top level of a HOSE diagram can literally reflect the organization of the application hardware.

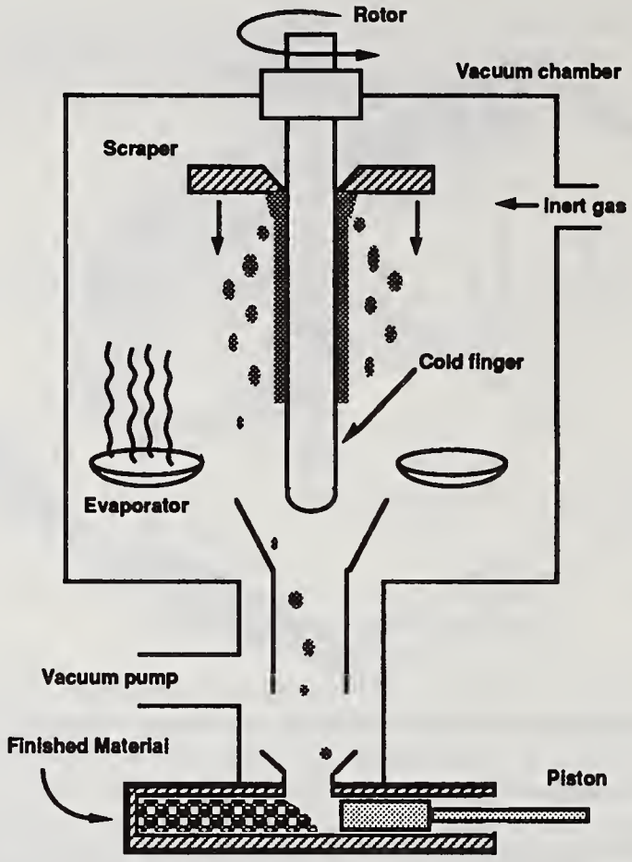


Figure 14. Nano-crystalline material fabrication.

Each of the hardware subsystems is represented on the HOSE diagram (Figure 15) by an object that contains a feedback controller for one process variable. A time series of state vectors enters the diagram from an external data stream. This vector is demultiplexed into six scalar signals for the sub-process controllers.

The goal of the nanocrystal formation process is to enable material science researchers at MTS to experiment with the topology of the process control diagram as well as process parameters in an effort to control attributes of the finished material. The developers of this system are experts in material science. HOSE enables them to utilize high level control and sensor processing objects in a flexible rapid prototyping environment without the need to confront details of concurrent programming techniques.

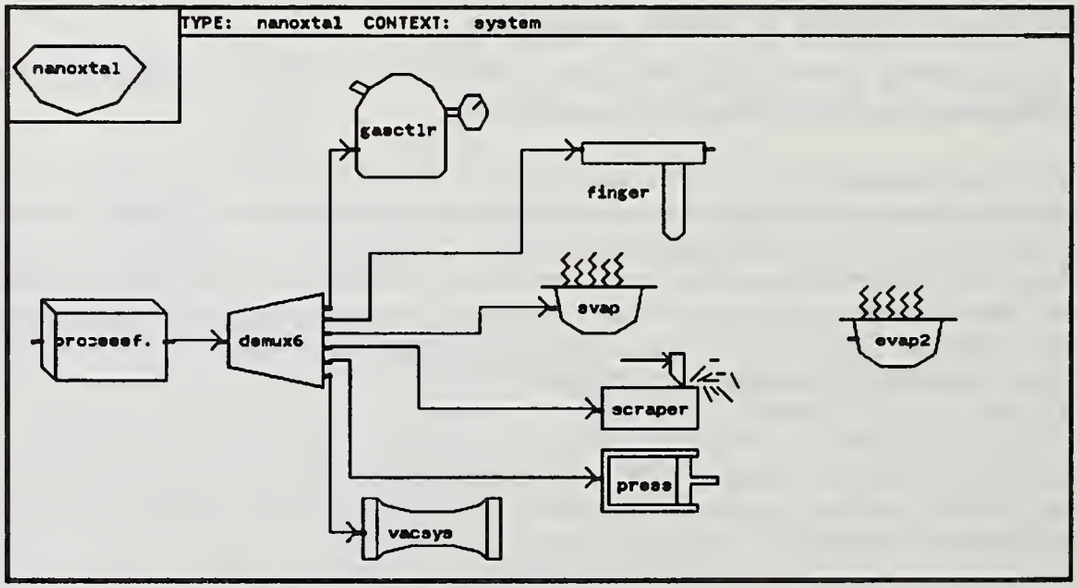


Figure 15. Nano-crystallizer control system.

3 The iconic user interface

Hierarchical dataflow diagrams provide an intuitive and powerful medium for the design and implementation of complex control systems. For the end user, however, a application dependent interface must be constructed. The current popularity of desktop-metaphore graphics workstations and personal computers attests to the intuitive power of the *direct manipulation interface.* In a direct manipulation interface the end user controls the application through the interactive manipulation of images that represent objects in the application domain. One of the earliest applications of object-oriented programming technology was in the user interface to the programming language Smalltalk. In many respects Smalltalk was (and still is) the cannonical object-oriented language. Larry Tesler [TES81], Daniel Ingalls [ING81] and David Robson, [ROB81] present particularly articulate descriptions of object-oriented systems and the associated user interface principles in their inaugural description of the Smalltalk 80 system. The philosopical motivation for the first Smalltalk implementation is presented by [KAY74]. Three key key ideas from these seminal works are summarized below:

Iconic principle: This is the "WYSIWYG" or What You See is What You Get concept. The application interface displays the state of the system by direct animated pictorial representations. Command languages and related techniques that require the end-user to memorize a text-based language are avoided.

Reactive principle: The application is responsive to the user's initiatives at all times and provides immediate graphical feedback about the current state of the system.

Modeless principle: The application supports a consistent "visual grammar": Similar manipulations perform the same generic operations no matter what objects are being manipulated. The application is free from distinct *modes* in which the user must understand new rules for interaction with different application components.

3.1 Implementation issues

The HOSE programming system has an object-oriented iconic interface that satisfies the above criteria, but this interface was developed by distinctly non-object oriented methods. (Object-oriented languages suitable for systems programming were not available to us in 1983.) The effort required to build the graphical interface to HOSE was considerable, and we felt that similar efforts could not be justified for the user interface of a typical custom industrial automation system.

The value of object-oriented language for iconic control panel development is well established. References [BHAS86] and [FRE87] are representative. These favorable results convinced us to employ object-oriented programming for the implementation of our next generation real-time software architecture. Unfortunately, we found there were no "industrial strength" object-oriented programming tools available. In this context, industrial strength refers to the following somewhat commercial considerations:

- Graphics quality: Iconic controls on the workstation screen represent hardware knobs, buttons and displays. The associated mechanical devices are often multi-million dollar systems. The end user's only access to this system is through the graphical interface which will strongly influence their perception of the whole system's quality.

- Performance: Performance issues are particularly acute for iconic controls that allow the user to adjust system parameters with the pointing device (mouse). The animated feedback from the iconic control and the concurrent update of displays that show the state of the system must be very rapid to appease users accustomed to hardware knobs, buttons and oscilloscopes.

- Portability: We wish to maintain the value of our investment in software and retain freedom to select alternative computing hardware for future applications. Some of the best realizations of object-oriented user interface management systems only operate on a single manufacture's workstation. (Most Macintosh applications, for example.)

- Open architecture: It must be possible to efficiently couple external software written with conventional languages. Performance considerations make the use of conventionally compiled code necessary for some components of the system.

When we began evaluating potential object-oriented languages in 1981. Smalltalk and object-oriented Lisp systems were the principle choices. Although we admired many features of these programming environments, they had several drawbacks as well:

- The use of message passing throughout the system down to the lowest level control features limited the performance of the resulting applications. Complex animated displays were unacceptably slow on commonly available workstations.

- The memory management systems introduced delays in response to user actions. In some object-oriented environments, garbage collection effectively stopped the user interface for several seconds. This behavior was unacceptable for the operator panel of an industrial process.

- The memory management software moved objects in memory at unpredictable times. This made it impossible for external application software to efficiently access large data objects.

- The internal representation of objects required to support memory management was not compatible with the natural represenation of data used by other language compilers. This limited our ability to use the object-oriented software as part of an open system architecture.

- Smalltalk (in particular) was an enormous package and required many months or even years of experience to utilize its capabilities effectively. We wanted an environment that could be learned by industrial engineers and application specialists in a reasonable amount of time. The size of the resulting application was also of concern.

To overcome these difficulties, we developed an object-oriented programming language called Alltalk. The design objectives for this language include simplicity, portability, high performance, and the ability to communicate with external applications.

3.2 The Alltalk Programming Language

Alltalk is an object-oriented programming language optimized for building iconic control panels for real-time industrial control systems. The conceptual framework for Alltalk is very similar to that found in other Smalltalk-derivative languages. All programming, testing and debugging is done in a completely interactive environment that utilizes the same iconic interface components available to the end-user. Not being exclusively motivated by the 'purist' philosophy of the Smalltalk developers, we found it possible to realize most of our requirements in a small (80k byte) language interpreter.

Alltalk differs from Smalltalk in several important respects:

- Alltalk is a smaller, simpler language with a more conventional syntax. Consequently, it is easier for programmers to learn and, in our opinion, much easier to read than Smalltalk or the various Lisp dialects.

- Alltalk communicates with other external programs. Through a *remote object* mechanism, objects and their associated code can be developed to run on external parallel processors. To the Alltalk programmer, these objects behave like local objects. (The remote processor communication mechanism is completely transparent.) Code to implement a remote object class is written in ANSI standard "C."

- Alltalk is itself written in ANSI standard "C" and is consequently very portable. Versions currently exist for DEC/VAX, Apple Macintosh, IBM /PC, and various UNIX systems. Alltalk graphics primitives are implemented with the local graphics toolkit of the host workstation. Alltalk graphics can be supported by X Windows (on UNIX or VAX/VMS), Presentation Manager (on IBM PC), or Quickdraw (on the Macintosh II).

3.2.1 A controller example with Alltalk

It is beyond the scope of this paper to discuss the detailed syntax and semantics of the Alltalk language. The following example demonstrates the construction of a simple control panel for a servo control loop that runs on an external digital signal processor. Part of the Alltalk source code that implements this example is presented and discussed briefly. Readers who are completely unfamiliar with object-oriented programming concepts are urged to consult the references [GOLD83] and/or [ROB81]. A qualitative understanding of object-oriented programing concepts is sufficient to understand the example implementation.

The dataflow diagram (figure 16) contains a square wave generator, *fg;* a proportional controller, *servo;* a plant simulation, *biquad,* and data acquisition device, *buffer.* The *buffer* object collects a set of sequential time samples for periodic display or other analysis operations.

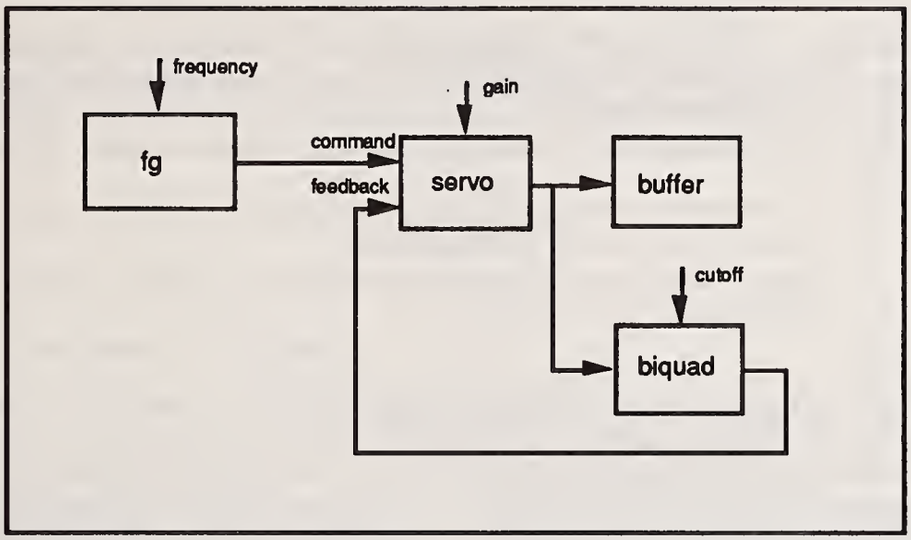


Figure 16. Control loop with terminal labels.

The objects shown on the diagram execute on a remote signal processor. The signal processor operates as a coprocessor sharing the bus with the workstation processor. In the Alltalk environment, remote objects are created *as though* they were local objects. The following *message expressions* create the four objects:

```
fg     := SquareWaveGenerator.new.init
servo  := ProportionalControl.new.init
biqaud:= BiquadraticFilter.new.init
buffer:= DataSampler.new.init
```

An Alltalk message expression consists of an object expression followed by one or more messages. Each message consists of a dot followed by a message selector and, optionally, an actual parameter list enclosed in square brackets. The syntax for Alltalk message expressions is very similar to that used in *Object Pascal.* [TES85]

In the first expression above, the variable *fg* is assigned the value of the message expression that appears on the right side of the ":=" assignment token. The variable *SquareWaveGenerator* contains a class. The message *.new* causes the class to create an instance object which is returned as the value of the message. The *.init* message is then sent to the new instance object to perform certain internal initializations.

The signal processing objects are interconnected to form the illustrated data flow diagram:

```
fg.output.connect[servo.command]
servo.output.connect[biquad.input]
biquad.output.connect[servo.feedback]
biquad.output.connect[buffer.input]
```

In the first expression above, the *.output* message is sent to the square wave generator, *fg;* which returns a *terminal object.* This terminal object understands the *.connect* message and establishes a new dataflow connection. The *.connect* message has a single actual parameter, the expression *servo.command,* which specifies the *command* entry terminal labeled on the controller diagram. (figure 16)

(In the HOSE environment, the operations required to create and interconnect objects require only manipulations with the mouse and the diagram displayed in the workstation window. The HOSE object manager generates a script similar to the expressions above as it communicates with the real time kernel.)

The Alltalk control panel in figure 17 is created by the following statements:

```
scope := ImageRemoteScope.new.init
      .at[100,100].extend[140,100]

gain := RealKnob.new.init[scope.content]
      .at[20,140].graduate[80]
      .setName['Gain']
      .setRange[0.0, 3.0]

freq := RealKnob.new.init[ir.content]
      .below[gain,10].graduate[80]
      .setName['Freq'].setUnits['Hz']
      .setRange[0.0, 0.1]

plant := RealKnob.new.init[ir.content]
      .below[freq,10].graduate[80]
      .setName['Plant'].setUnits['Hz']
      .setRange[0.0, 1.0]
```

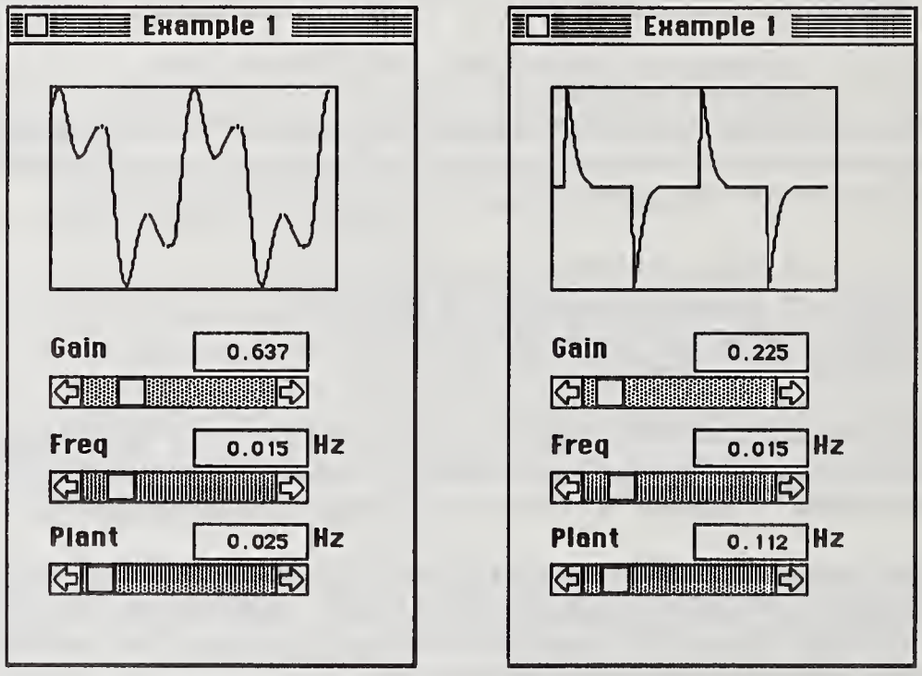


Figure 17. Two views of the control panel

The oscilloscope display in figure 17 can show any signal collected by the *buffer* object. The *gain* knob adjusts the proportional gain of the *servo* object, *freq* adjusts the frequency of the square wave generator, and *plant* adjusts the cutoff frequency of the plant simulation object. The plant simulation in this example is a biquadratic filter modified to be a low pass filter.

The iconic interface controls are connected to the remote objects with these messages:

```
gainKnob.connect[servo.gain]
freqKnob.connect[fg.frequency]
plantKnob.connect[biquad.cutoff]
scope.connect[buffer]
```

Finally we activate the objects:

```
fg.start
servo.start
biquad.start
scope.start
```

The user can operate any of the knobs using the mouse. The display operates continuously while the knobs are adjusted. (All displays on an Alltalk control panel operate continuously and concurrently.) In the left panel of figure 17, the *scope* object displays the output from the *biquad* object.

New objects can be created either locally or in the remote signal processor at any time. Connections can be added or changed at any time. It is not necessary to stop an object to alter the connections or send a message, although in some applications, such as the laser welding robot (section 2.3.7.2), people often feel a need to do so. The following message will connect the input of the data sampler *buffer* to the output of the *servo* object.

```
servo.output.connect[buffer.input]
```

The oscilloscope trace in the right panel of figure 17 now shows the *servo* output signal.

3.2.2 Software development in Alltalk

Alltalk programming is supported by a graphical interface called the *browser* (figure 18). The browser allows the developer to examine the implementation of any class in the system and add, modify, or replace any class or method. The browser window has three scrolling *panes*, the upper left pane contains a list of all classes in the system. By selecting a class with the mouse, the source code for the class and all of its methods are displayed in the larger bottom pane. The upper right pane shows the selectors for methods in the selected class. By selecting a selector from this pane, the source pane scrolls to bring the selected method into view. The menus above the source pane support editing operations and incremental compilation of modified methods or classes. Class *Browser* is itself implemented entirely from other Alltalk classes. The developer can create multiple instances of *Browser* to view several classes in separate workstation windows. The methodology of programming supported by Alltalk is very similar to that used in the Smalltalk system. An excellent summary of this interactive style is presented in [GOLD84].

In figure 18, the browser shows the implementation of class RealKnob, used to make the controls shown in the control loop example above. A RealKnob inherits most of its behavior from IntegerKnob (not shown). Only two methods are needed to implement this class, *init*, and *setFormat*. The class adds one instance variable, *displayDecimals*, to those already inherited from IntegerKnob. See [GOLD83] for a description of the more elaborate Smalltalk browser, from which we have borrowed many ideas.

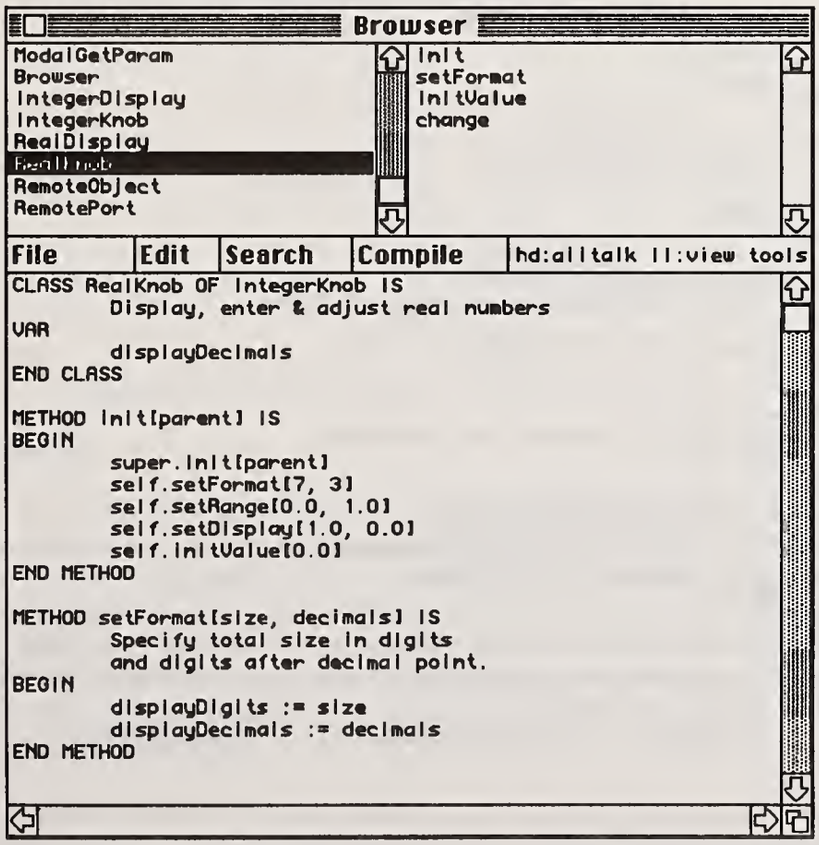


Figure 18. Browser display for class RealKnob.

In figure 19 below, the browser displays the source code for the oscilloscope object used in our control demonstration. This code contains an example of every syntactic construction supported by the Alltalk language. Further information about Alltalk programming and the associated class libraries may be obtained from [SPAR86].

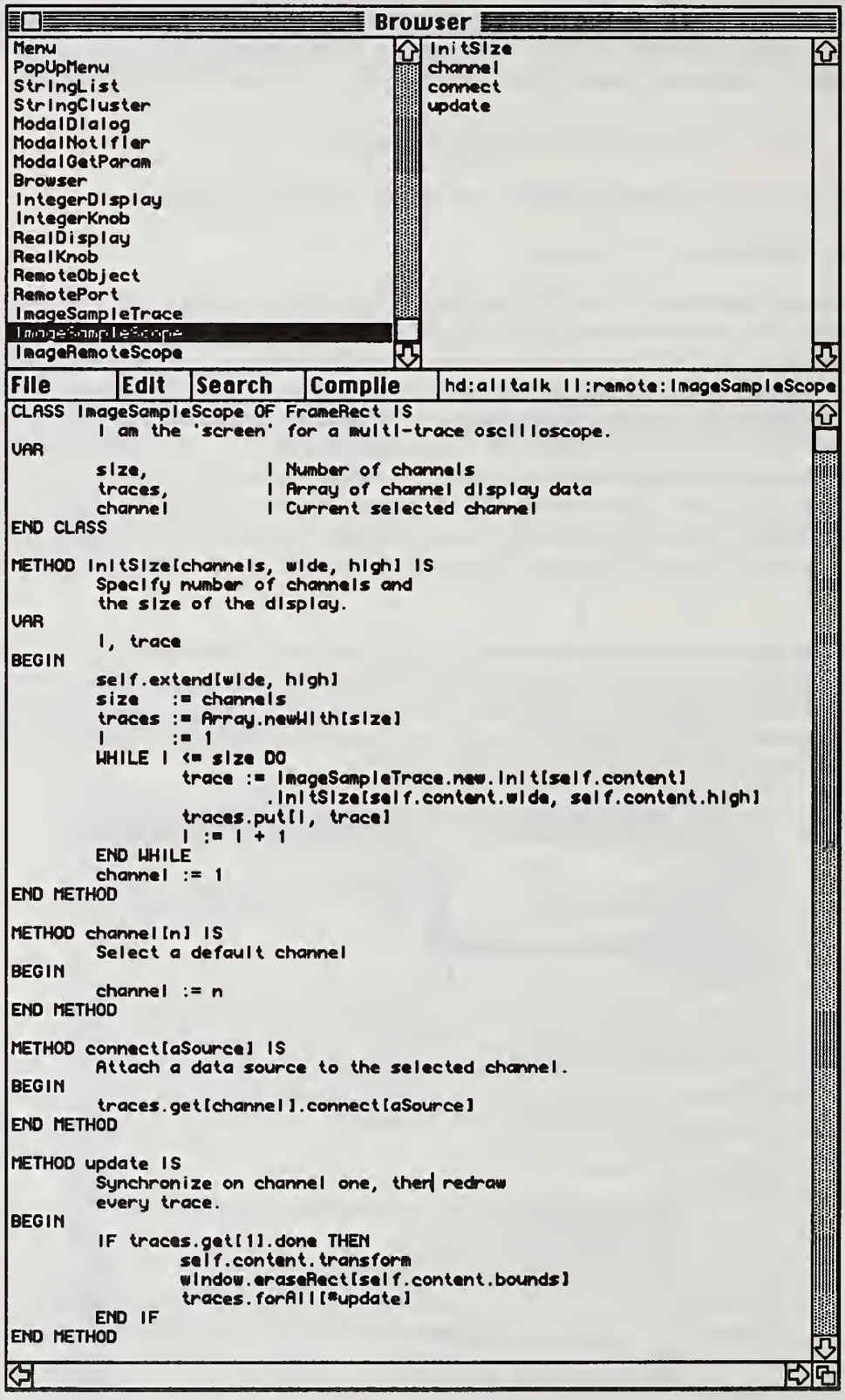


Figure 19. Implementation of an Oscilloscope.

4.1 Summary of results

Our experience has shown that object-oriented programming and development techniques can be applied to industrial automation applications at multiple levels. At the lowest levels of the system where traditional sequential language techniques must still be used, we have adopted object-oriented extensions to the C and Pascal programming languages.

For intermediate levels of the system architecture, we use the HOSE programming system. The graphical dataflow nature of HOSE makes it especially appropriate for the description and implementation of complex concurrent processes on parallel processing hardware.

At the user interface level we use another object-oriented interactive programming environment, Alltalk, which is optimized for the development of animated iconic control panels for real-time systems. The resulting interface is reactive, modeless, and easy to tailor for custom applications.

We wish to provide an interactive graphical programming environment that will support dataflow diagram editing tools and a complete toolkit of integrated iconic interface components. The end-user presentation utilizes animated graphical control panels. Figure 20 below is a schematic representation of our component technologies.

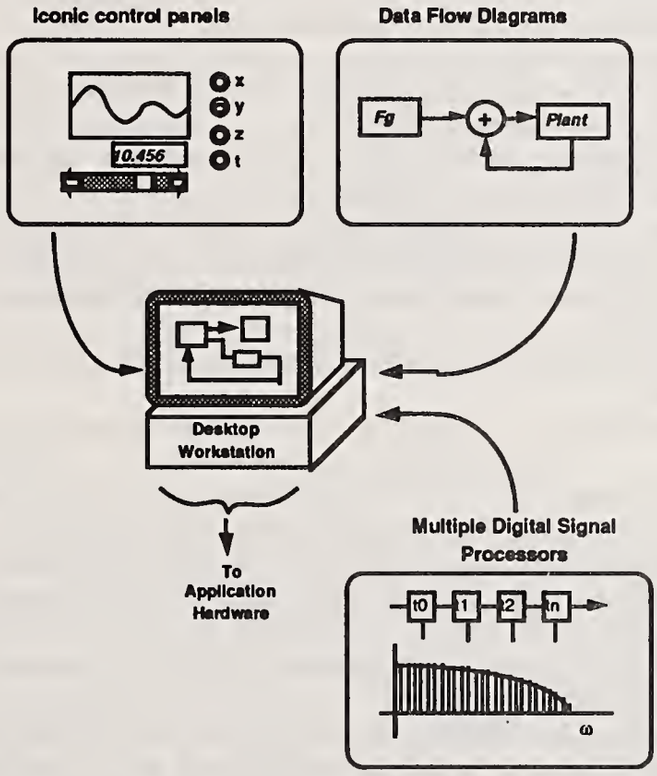


Figure 20. Desktop control system development.

4.2 Future objectives

As figure 20 suggests, we are developing our dataflow language to support execution on multiple digital signal processors. (The controller example in the previous section used only one external signal processor.) Our current desktop workstation is the Apple Macintosh II. We use parallel processing hardware installed on the internal NuBus consisting of one 68020 coprocessor from GreenSpring and multiple Texas Instruments TMS 320C30 floating point signal processors. A parallel effort at MTS will implement the dataflow kernel and associated Alltalk programming environment with the Sun SPARC workstaion supervising multiple Motorola 88000 processors on an attached VME bus. Versions of new primary objects that run on the digital signal processors are being developed with an object-oriented extension of the C language.

Further integration of the Alltalk and HOSE is also underway. We will support the HOSE diagram editor entirely with Alltalk control panel objects. This will allow developers to construct dataflow diagrams and iconic controls within a common user interface framework.

Acknowledgements

The author would like to thank the other members of the HOSE development team, Dave Tillman (the co-inventor of HOSE) Jay Dougherty, Donna Hall, Shingchi Hsu, Neal Goman, Art Dee, Wayne Olsen, Bill Franks, Gary Tock, and Warren Persons. The HOSE control system examples were developed by Brad Thoen who is one of the most creative HOSE users and supporters. The LARS robot HOSE programs were developed by Charles Anderson, Jay Dougherty, Judy Carmine, Dean Hystad, and Phil Frey. Dawn White and Brad Cleveland, members of the MTS Advanced Technology Development Division, have provided a stimulating intellectual environment and have suggested many exciting applications.

References

[ALB87] Albus, J., *Hierarchical Control of Intelligent Machines*, Proceedings, Workshop on Space Telerobotics, JPL Publication 87-13, Vol. 1.

[AND90] Anderson, T. and Donath M., *Animal Behavior as a Paradigm for Developing Robot Autonomy*, Robots and Autonomous Systems, Elsevier, 1990.

[BHAS86] Bhaskar, K.S. and Peckol, J.K., *Virtual Instruments: Object Oriented Program Synthesis*, Proceedings of OOPSLA 86, ACM, 1986, pp. 303-314.

[CAM87] Camarinha-Matos, L., and Steiger-Garcao, A., *An Information System Architecture for Robot Cell Programming*, Computer Integrated Manufacturing: Status and Challenges, ed. I.B. Turksen, NATO ASI Series F: Vol. 48, Springer-Verlag, Berlin, 1987.

[CHAT88] Chatham, B. and Sparks, H., *Butterfly™ HOSE: Graphical Programming for Parallel Systems*, Abstracts of IEEE and USENIX Fifth Workshop on Real-Time Software and Operation Systems, Washington, 1988, pp. 75-79.

[CLEV86] Cleveland, B.,"An Intelligent Robotic Inspection System", SME Robotic Solutions in Aerospace Manufacturing Conference, March 1986.

[CLEV87] Cleveland, B., Tsuchiya, F. and White, D., *Sensors for Real Time Applications*, SME Machine Monitoring Sensors Conference, April 1986.

[CHIU86] Chiu, S., Morley D., Martin, J., *Sensor Data Fusion on a Parallel Processor*, IEEE International Conference on Robotics and Automation, San Francisco, CA., April 1986.

[FRE87] Freburger, K., *RAPID: Prototyping Control Panel Interfaces*, Proceedings of OOPSLA 87, ACM 1987, pp. 416-422.

[FRO89] Froes, F., Suryahanayana, C., *Nanocrystalline Metals for Structural Applications*, Journal of Metals, June 89, pp.12-17.

[GOLD83] Goldberg, A. and Robson, D., *Smalltalk: The Language and Its Implementation*, Addison-Wesley, 1983.

[GOLD84] Goldberg, A., *The Influence of an Object-Oriented Language on the Programming Environment*, in *Interactive Programming Environments, McGraw-Hill, 1984*, pp.141-174.

[HOS1] *HOSE Users Guide*, MTS Document #117599-00-778, MTS Systems Corporation, 1986.

[HOS2] *HOSE Reference Manual*, MTS Document #117600-00-778, MTS Systems Corporation, 1986.

[ING81] Ingalls, D., *Design Principles Behind Smalltalk*, Byte, August 1981, pp. 286-298.

[KAY74] Kay, A., *SMALLTALK, a communication medium for children of all ages*, Learning Research Group, Xerox Palo Alto Research Center, 1974.

[KUM87] Kumar, R., *Feedback Control Theory Approach for Scheduling Flexible Manufacturing Systems*, Computer Integrated Manufacturing: Status and Challenges, ed. I.B. Turksen, NATO ASI Series F: Vol. 48, Springer-Verlag, Berlin, 1987.

[NAY88] Naylor, A., Volz R., *Integration and Flexibility of Software for Integrated Manufacturing Systems*, Design and Analysis of Integrated Manufacturing Systems, ed. W.D. Compton, National Academy Press, Washington, D.C., 1988.

[OSS87] Ossher, H., *A Mechanism for Specifying the Structure of Large, Layered Systems*, in *Research Directions in Object-Oriented Programming*, MIT Press 1987, pp. 219-252.

[ROB81] Robson, D., *Object-Oriented Software Systems*, Byte, August 1981, pp. 74-86.

[SPAR86] Sparks, H., *Introduction to Programming with Alltalk*, available through MTS Systems Corporation Advanced Technology Division.

[SPAR90] Sparks, H. *Object Oriented Dataflow Programming Techniques for Industrial Automation*, in Proceedings of Control Expo 90, Chicago, 1990.

[TAL89] Talbot, J., Anderson, T., Donath M., *Scarecrow, An Implementation of Behavioral Control on a Mobile Robot*, Mobile Robots IV, Proceedings of the SPIE, Vol. 1195, Philadelphia, November, 1989.

[TES81] Tesler, L., *The Smalltalk Environment*, Byte, August 1981, pp. 90-147.

[WHIT88] White, D., *Development of Technology Transfer and Implementation Strategies for Intelligent Processing of Materials*, Robotics and Computer-Integrated Manufacturing, Vol. 4, No. 3/4, pp. 683-

PROF. DR.-ING. DRES. H.C. GÜNTER SPUR
DR.-ING. KAI MERTINS
DIPL.-ING. WOLFRAM SÜSSENGUTH

INTEGRATED INFORMATION MODELLING FOR CIM

1 Introduction

Computerized information processing determines more and more the integration of manufacturing enterprise functions. For that reason information technologies are used as a tool for organization and operation to support the whole manufacturing process. The division of labour can be given up in favour of new organization structures guided by the sequences of entrepreneurial functions /SPU86/.

The joint effort of users, vendors, consultants and scientists will enable the utilization of the potentials of information technologies for the industry in a fast and successful way. The essential tasks towards a enterprise wide usage of information technology are the development of procedures and methods for CIM-planning and the CIM-introduction as well as the development of open CIM-architectures. The efforts towards open CIM-architectures are supported by the work of the German "Kommission CIM" of DIN /DIN87/ (KCIM), founded in 1987, and the results of the related project titled "Scientific Basics and Support of Standardizing CIM-Interfaces" which is financed by the German department of R&D. The presented "Integrated Information Modelling" methodology is an approach to derive a CIM-architecture from a user`s point of view based on different models and modelling methods.

2 The role of integration in achieving enterprise goals

The goals which could be achieved by the integrative utilization of CIM-components (fig 1) were also pursued by the introduction of traditional "island" systems. A further improvement is expected by the integration of entrepreneurial functions using new methods and systems. To reduce the needed times for the development of products and for the production through-put, the integration of sequences of functions with less communication interfaces has to be created. A better meeting of schedules and a higher product quality will be reached by an early consideration of all relevant information. A reduction of cost can be expected by a reduction of the administration of interfaces, by an avoidance of repeated data input and by an increasing of the transparency of the manufacturing process.

An optimal CIM-system support of the whole manufacturing enterprise can be derived from the sequences of functions within the enterprise and from the functional requirements of the single place of work. The information technology should allow an independent formation of the sequences of operations and of conceptions of organization. The criteria for the development of manufacturing structures have to be mainly derived from the entrepreneurial goals concerning the market situation (fig 1) and the production program rather than from the actual capabilities of CIM-components e.g. capacity, functionality or compatibility. The operational demands onto an integrated information processing have to be collected in a neutral requirements specification from the user`s point of view. This specification should enable to derive the software specification and the implementation of the CIM-System. In this way the designing of the technical systems will loose its central role in the planning process, rather it will be only a part of the whole manufacturing development within an enterprise.

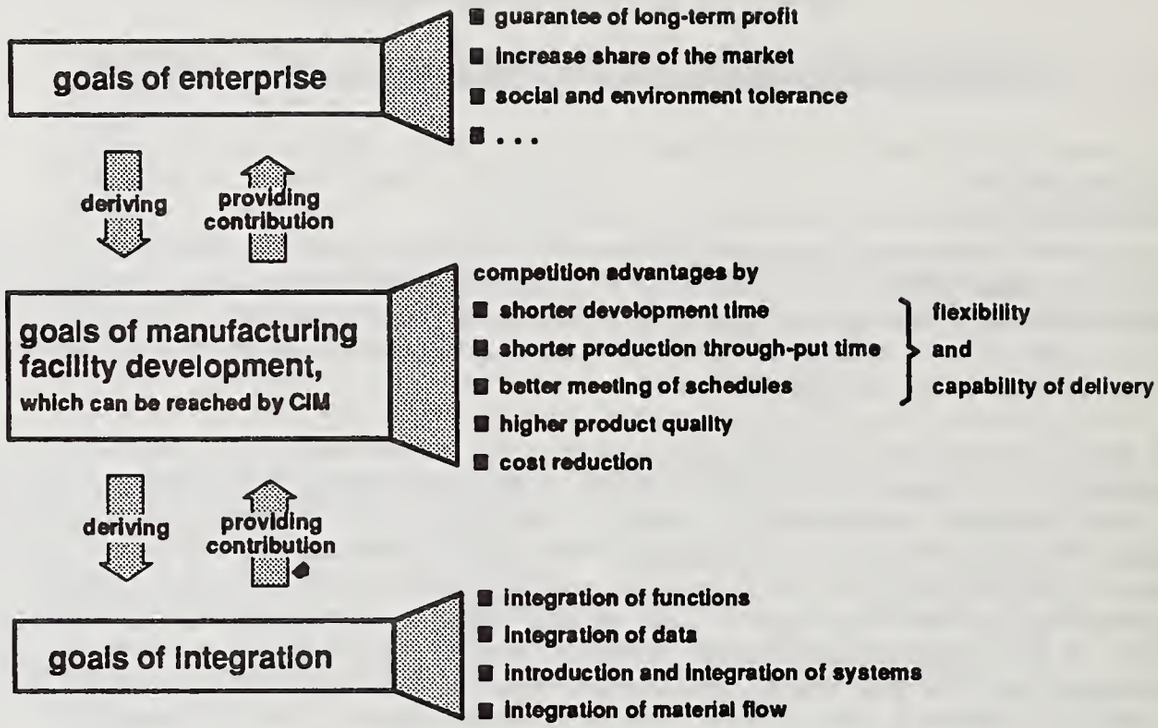


Figure 1: Hierarchy of goals within the CIM-introduction

3 CIM-architecture and manufacturing enterprise modelling

Architecture can be defined as "art of construction" or "style of construction", CIM-architecture could be interpreted as a "style of constructing" or "designing" the information processing within a manufacturing enterprise. The application of heterogeneous systems from different vendors has to be implemented within the architecture. The whole system has to be designed flexible to be open for prospective enlargements, and to enable the exchange of obsoleted components. That requires a reference between the application functions and the data of the systems of different vendors, as well as transparency and an easy technical access to the already existing data within the enterprise. That means, that CIM-components of different vendors not only need a common architecture in the level of hardware and of protocols, but also concerning the application functions, data and data types from the user`s point of view. A uniform understanding of functions and data and the deduction of the system support out of the requirements description of the enterprise can be reached by modelling the manufacturing enterprise. In this sense the model of the manufacturing enterprise and the modelling methodology has to become a part of an CIM-architecture (fig. 2).

Today the modelling of different parts of an enterprise is enabled by various methods which are used for different purposes. The "Integrated Information Modelling" approach will integrate different modelling approaches and will lead the user from a general architecture given by predefined model structures for CIM-implementation to a particular model and architecture of the system support in his own enterprise.

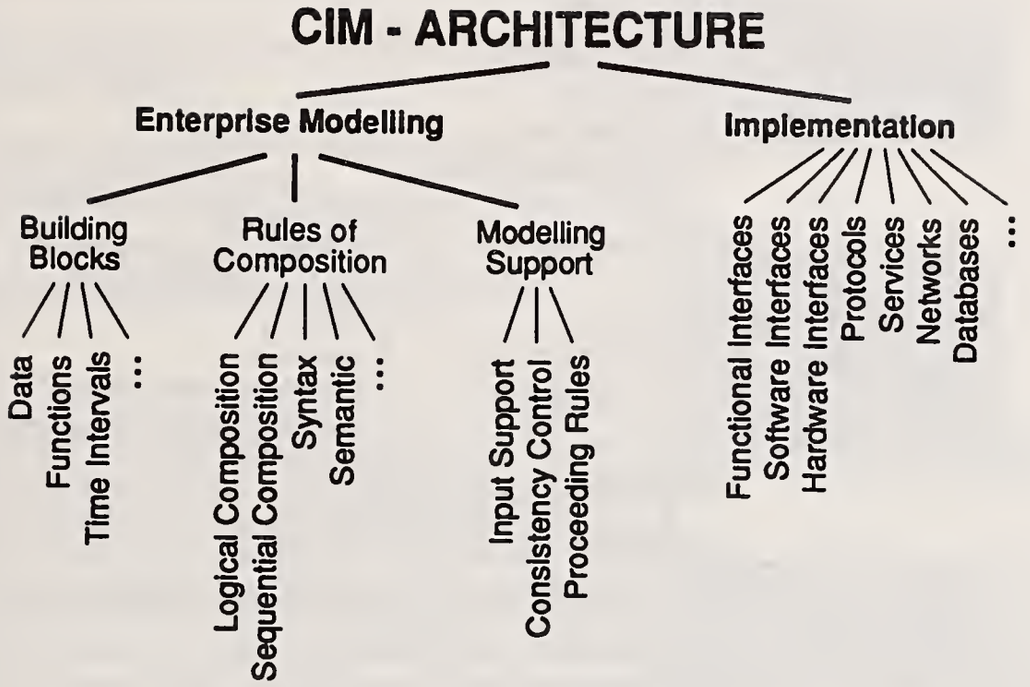


Figure 2: Modelling and Implementation as parts of a CIM-architecture

4 Existing modeiling approaches

4.1 Evaluation criteria and requirements

In the domain of manufacturing enterprise planning and computerized information processing different modelling methods have been developed. They were compared in cover to their usability for the enterprise modelling and CIM-architecture development. The comparison was done by three main criteria (fig. 3):

1. Represented Subjects.

2. Modelling capabilities.

3. Domain of application.

The **represented subjects** can be distinguished into functions, data, decisions, time, space and physical means. Every production system has aspects from each kind of subject but most of the methods allow only the effective representation of one or two subjects.

Modelling capabilities were determined by the kinds of models which were represented or produced, by the description method and by the provision of a structured proceeding. Some approaches provide only a static frame model others provide description means and a structured proceeding for the generation of particular models. Within some of the approaches these particular models are only a static descriptions of the reality within others a runtime model can be generated and simulated.

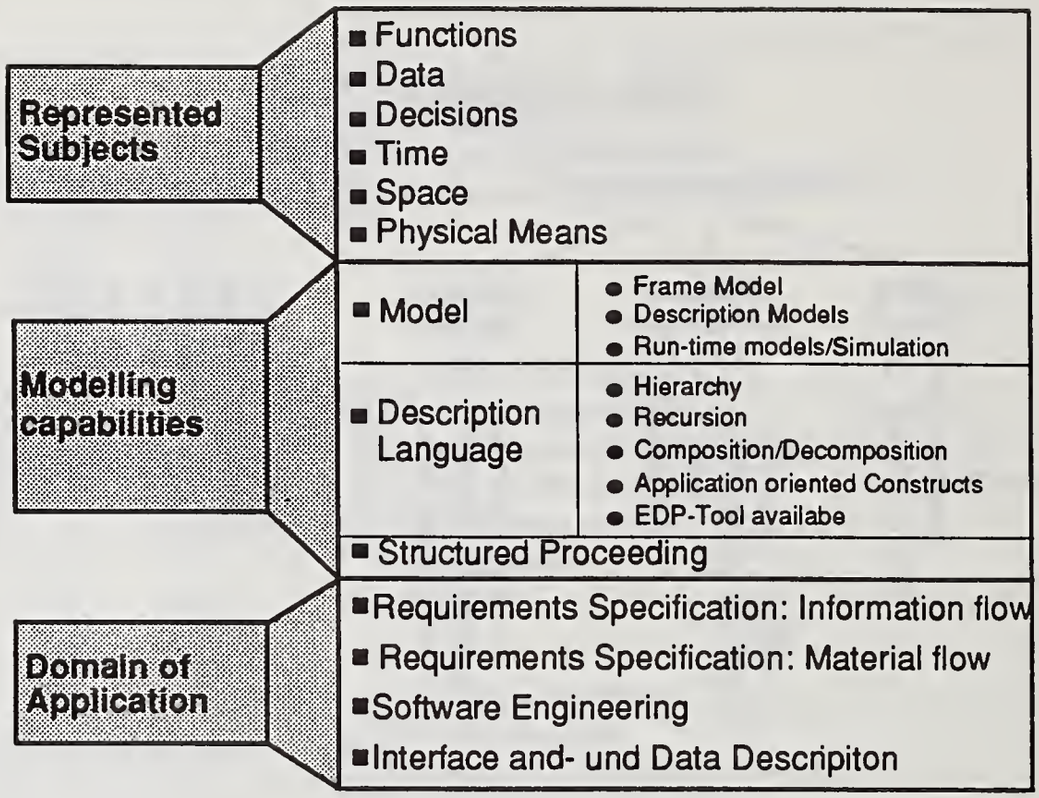


Figure 3: Evaluation criteria for modelling approaches for CIM

The **domain of application** is given by different purposes of modelling. Within the CIM planning and development two main levels of modelling can be distinguished (**fig. 4**). The first is the definition of requirements by modelling the enterprise from the user's point of view. The aim of this task is to clarify and to specify the requirements for the enterprise information processing and the corresponding technical components. The second level is the design and development of software for CIM-components. Other domains of application of powerful modelling approaches are the description of interfaces as well as the planning of material flow systems.

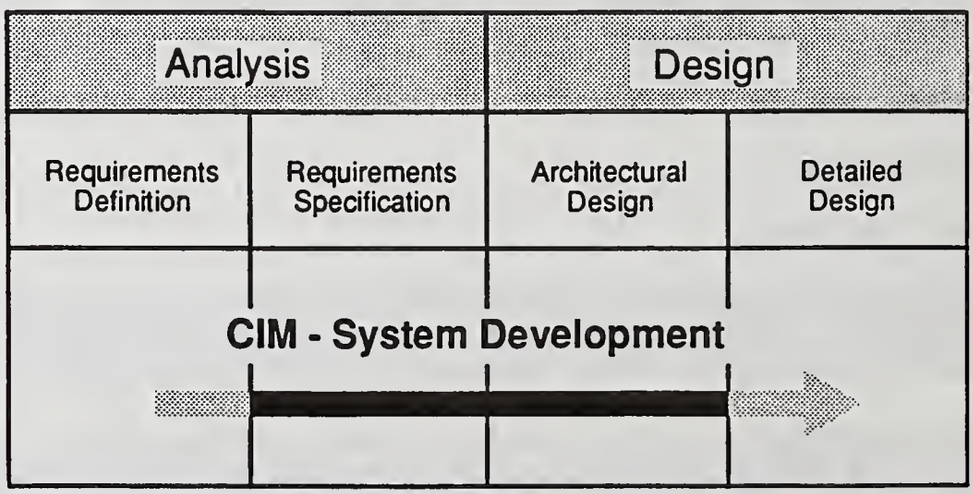


Figure 4: Levels of Modelling

General requirements for modelling methods for CIM are:

- A complete representation of all relevant elements, processes and features of a manufacturing enterprise.

- Realistic models and a simple modelling, so that it is easier for the users to participate with in the planning process.

- The integrated usage of the modelling methods for the requirements specification, for the software design and for the implementation.

4.2 Comparison

Table 1 shows the comparison of important modelling methodologies from different application domains. The following discussion concerns some methodologies and models which were developed for the CIM-planning and development.

At the end of the seventies the **IDEF** modelling language was developed within the ICAM-program in the U.S.A.. It is based on the **SADT**-method and was primarily used in the aircraft industry. Other models followed using this or a similar method. The common basic approach is a functional description of the manufacturing enterprise and the description of the data relations between the functions. The models mainly differ in branches and in areas of production, in which they were applicated, as well as in the degree of detail and the capability of functions on the lowest level of degree. In 1984 Harrington /**HAR85**/ formed a functional model of the manufacturing process including four main functions and 18 different singular functions in four levels of hierarchy.

For the analysis of manufacturing enterprises, the determination of integration potentials and for data storage approaches a functional reference model of industrial production was developed at the Centre for Production Technology in Berlin /**DUE86, MER88**/ using the SADT-method. The essential requirement was the independence of the model from the influence of various operational characteristics such as area of application, manufacturer's field, production quantity and product complexity. It includes seven main functions in the so called A0-level which were detailed over 4 levels into 70 singular functions.

In the frame of the ESPRIT program further modelling approaches were pursued by the projects "Design Rules for CIM Systems" /**YEO84**/, "CIM-OSA" (Open System Architecture) /**AMI88**/, and "OPEN CAM System". In 1984, within the "Design-Rules" Project 16 "elementary production functions" were described with the flow chart technique to lay down the requirements for CIM building blocks. The description considered functions as well as data. The usage of the flow chart technique leads to a restriction in the degree of detail.

In the CIM OSA project 21 project partners have been working on an European CIM-architecture since 1985. CIM-OSA released a framework for the development of an open CIM-architecture and an IDEF-based functional modelling method and announced several methods for other modelling purposes as well as prototypes of an integrating infrastructure. The mentioned framework was the main input for the draft ENV "Framework for modelling" worked out by the CENCENEC/AMT/WG-Architecture /**CEN90**/. The framework includes

Criteria of Evaluation / Modelling Approach			SADT IDEF	ISO TC 184 Shop Floor Reference Model	CEN/C. Framework for Modelling	ESPRIT Open CAM System	ESPRIT CIM OSA	GRAI-Method	STEP-EXPRESS	Manufact. Message Service UMS	Entity-Relationship Method	Decision Tables	Extended Petri Nets	IFK Models
Represented subjects	Functions		+	+	+	+	+	o	-	o	-	o	o	+
	Data		+	o	+	+	+	o	+	+	+	o	o	o
	Decisions		-	-	-	+	o	+	-	-	-	+	+	o
	Time		-	-	-	-	(+)	o	-	-	-	-	o	+
	Space		-	-	-	-	(+)	-	-	-	-	-	-	o
	Physical Means		-	-	o	o	(+)	o	-	+	-	-	-	o
Modelling Capability	Model	Frame Model	-	+	+	+	+	-	+	+	-	-	-	-
		Description models	+	o	-	+	+	+	+	o	+	+	+	+
		Runtime models/Simulation	-	-	-	-	(+)	o	-	-	-	+	+	+
	Description Method	Hierarchy	+	+	-	+	+	+	+	-	+	o	-	+
		Recursion	+	+	-	+	+	+	-	-	+	o	-	+
		Composition/Decomposition	+	+	-	+	+	+	-	-	+	o	-	+
		Application orientated Constructs	o	o	-	o	o	+	-	-	-	-	-	+
		EDP-Tool available	+	-	-	o	(+)	-	+	o	+	+	-	+
	Structured Proceeding		+	-	+	o	(+)	o	o	o	+	o	o	+
Application Domain	Requirements Specification: Information Flow		+	+	+	+	+	+	-	-	-	o	o	o
	Requirements Specification: Material Flow.		-	o	-	+	-	-	-	-	-	o	o	+
	Software Engineering		+	-	o	o	o	-	-	-	+	o	o	-
	Interface and Data Specification		o	-	o	-	o	-	o	+	o	-	-	-

+ suitable/available o reduced suitable/available - not suitable/available ( ) announced

Table 1: Comparison of modelling approaches

- three levels of modelling: (User) requirements, (system) design and implementation,

- three levels of genericity of models: generic building blocks, partial models and particular models,

- and four views onto the model: the function view, the information view, the resource view and the organizational view.

Up to now it is an empty framework without any building blocks or methods. CIM OSA and other institutes and companies are making proposals for the "filling of the framework" with methods, constructs and partial models.

Another modelling approach is the definition of a "Product Model". The "Product Model" in the sense of Krause /KRA86/ represents a logical structure for all relevant data and methods of the product by definition of information layers. Each layer represents a semantical complete part of the whole product information. Within the STEP-standardization work /ISO/ different partial models of a product model were developed and were brought into the standardization work. The "Product Model" approach is more data concerned than the approaches mentioned before.

A more vendor related modelling approach was presented by Digital Equipment in 1986 as a "CIM-architecture" /FLA86/. Goal of this CIM architecture was to decouple the four functions of the data processing (in-/output; processing; storage and transfer) by defined interfaces, protocols and "handlers" from each other as well as from operating systems and hardware. In each of the layers the technology could be adapted to the state of the art without changing the technology in the other layers.

Summing up, it can be marked, that most of the approaches deal with functions and data structures or one of them. Other aspects are were in some approaches. Concerning to the description of functions a method similar to the IDEF-method is used mostly. A practical applicability of this method is provided by a simultaneous computer aided support only. Further on, main aspects for the planning of CIM-systems are unconsidered within the most approaches, such as

- a quantitative modelling of the data exchange,
- a modelling of concurrent processes,
- a modelling of the time aspects within a system,
- and the relation of functions to the objects of the systems.

The approach of the product model lays an emphasis onto the structuring of data and methods for product development and design. By an extension towards a "resource model" and a "control model" it will be possible to consider the informations within the manufacturing enterprise in their entirety.

Based on this situation the "Integrated Information Modelling" approach will be introduced next. It tries to show a solution for an integrative modelling of functions and informations of CIM-systems as well as providing a platform for the interlinking of methodologies which cover other aspects.

5 Integrated information modelling

5.1 Classes of objects in a manufacturing enterprise

In systems theory functional, hierarchical and structural concepts were distinguished. A system is separated from its surroundings by a "system border". Those things that cross the border were called the "operands" of a system. Systemtheoretical descriptions of machine tools and manufacturing systems distinguish traditionally three kinds of operands /SPU72/ by their **constitution**:

- material,
- information,
- energy.

For the modelling of the material flow and the manufacturing process some approaches /AMI88, ISO89/ make the basic distinction into

- material,
- information,
- resources.

This distinction assumes that the material processing is the main task in manufacturing and information and resources were needed for this. For a wider view there is a problem: information itself can be a product and needs resources and control to be processed, otherwise also information can be a resource, too. Therefore it seems useful to differentiate the operands by another criteria. The "Integrated Information Modelling" approach distinguishes the operands of a manufacturing enterprise by their **intended purpose** into

- products,
- orders,
- resources .

Within this distinction products and resources can consist of material, information or energy. Orders only consist of information. Fig. 5 shows that the distinction by constitution and by the intended purpose are two views onto the same operands. The distinction into material, information and resources mingles the two views and doesn't fit into fig. 5.

The three kinds of operands lead to the main classes of objects in a manufacturing enterprise from the user`s point of view. The objects of each class have a specific generic structure, means it is possible to predefine a frame for their structural and functional behavior. Within the modelling process of a real enterprise the real objects have to be related to one of these three classes.

5.2 Activities in a manufacturing enterprise

Everything that happens in a manufacturing enterprise as part of the manufacturing process can be described by activities. For the modelling purpose a generic activity model is defined in respect to /ISO89/. The content of an activity includes in particular depends on the level of detailing within the modelling process.

Operands of the System "Mfg. Enterprise"		distinguished by Constitution		
		Material	**Information**	**Energy**
distinguished by Intended Purpose	**Products**	Physical Components of the Products	Informational Components of the Products (Description of Products)	Production of Energy
	Orders	— — —	Order to execute an Activity	— — —
	Resources	Physical Resources - Machines - Computer	Informational Resources - Knowledge - Skills	Energy
		Human Resources		

Figure 5: Operands of the system "manufacturing enterprise"

In general activities process and modify objects which were classified above into products, orders and resources. In fig 6 the modified objects are represented by arrows from the left to the right. For the execution of the activities there are two perquisites:

1. An **Order** to execute the activity. The order is represented by an arrow from the top. Orders for the execution of activities (arrow from the top) come either from outside of the system or have to be generated for that purpose by another activity. While modelling this activity the order has to be represented as an arrow coming out of the activity block pointing to the right because this object "order" was generated or changed.

2. **Resources** which are capable of executing the activity. Normally several resources are necessary. The resources that execute an activity are represented by an arrow from below. Resources for the execution of activities (arrow from below) come either from outside the system or have to be provided for that purpose by another activity. Modelling this activity the resource has to be represented as an arrow from the left through the activity block because it is an object which was processed by this activity.

Within the modelling process three levels of representation of activities are differed as shown in fig 6: action, function and the complete activity. This allows modelling on different levels of particularity.

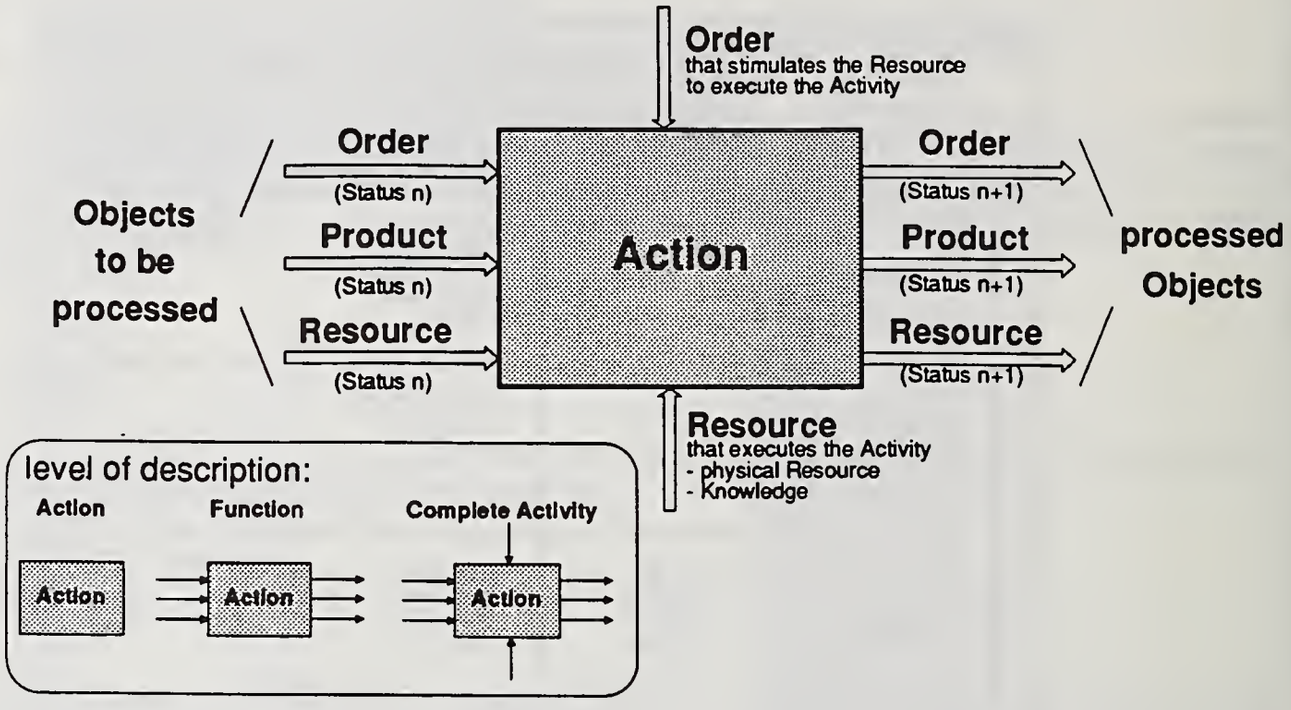


Figure 6: Generic activity model

The definition of the three object classes and the generic activity model provide the basic constructs for the generation of particular manufacturing enterprise models. In order to provide a modelling method a detailed specification of the constructs and a structure for their integration to a manufacturing enterprise model are needed.

5.3 Kernel of a manufacturing enterprise model

5.3.1 General

The kernel of the enterprise model will be derived from the classes of the objects and from the generic activity model. Two main views onto this kernel can be distinguished: the function model and the information model (fig. 7). Both views are interlinked by using the same objects and activities but they represent them in different ways and different grades of detail and different context. An extension towards other views will be possible, the way to achieve this will be shown later.

5.3.2 Function model

The functional model represents the reality of the manufacturing enterprise by the information processing activities and their logical and temporal interlinking. According to /HAR85/ also the activities of the flow of material and processing of material are able to be represented as informations, for example the execution of work plans or NC-programs.

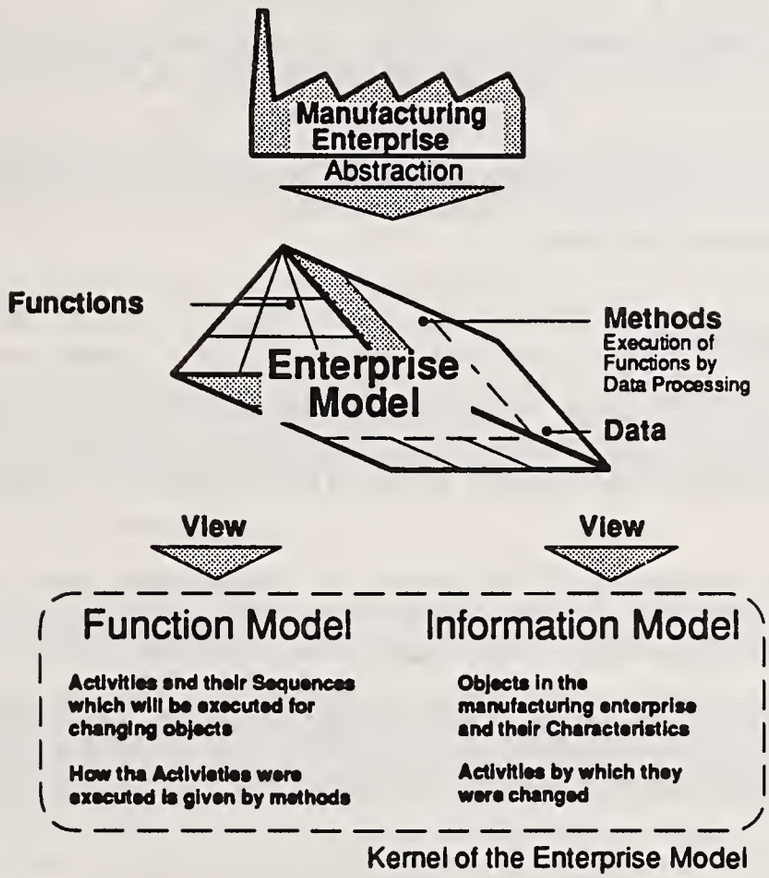


Figure 7: Kernel of the enterprise model

For the first structuring of the function model only the functions and not the complete activities should be represented. That means the process of "changing the objects" is the main task first, (the arrows from above and below are faded out). The functions could be used in a hierarchical way differing the grade of detail.

In general three levels of modelling are differed: function elements, sequences of functions and partial autonomic units.

The function elements represent the discrete steps in processing the objects within the manufacturing process. It seems possible to derive a lot of function elements from different perspectives and to them as a general standard. These predefined function elements could be provided to the user who has to complete them for modelling his particular enterprise.

The sequences and interlinking of functions could only be described sometimes in a general way. Using them within the modelling of a particular enterprise they have to be modified by using the defined function elements.

The definition of standardizable function elements and sequences of functions enables an uniform understanding about a wide field of application systems. For example the order control in a whole enterprise could be modelled by a set of function elements.

The description of partially autonomic factory units as a conclusion of functions and their inter-

linking to larger networks is useful for the separated modelling of areas of an enterprise if a defined interface concerning functions, decisions and responsibility could be found. These units should represent the scope for decisions which are autonomous workable within a larger enterprise.

The development of particular function models has to be extended by the order and resource flow, the analysis of concurrence of functions and their correlated influence. To that simulation and other methods have to be used.

5.3.2 Information model

The structure of the objects within the "Integrated Information Modelling" will use the main features of "object oriented" approaches: the close relation between functions and data of an object, inheritance and the class concept. Further a fully hierarchical modelling of objects has to be enabled.

The collection and structuring of the data of all objects which were identified within the modelling process lead to a particular enterprise information model. For this purpose a structuring frame for the representation of the relevant data in manufacturing enterprises is needed.

The distinction into the three classes of objects and their generic internal structure yields a predefined structure of the enterprise information model. Three submodels, the product model, the control model and the resource model were defined (fig 8). The internal structure of the

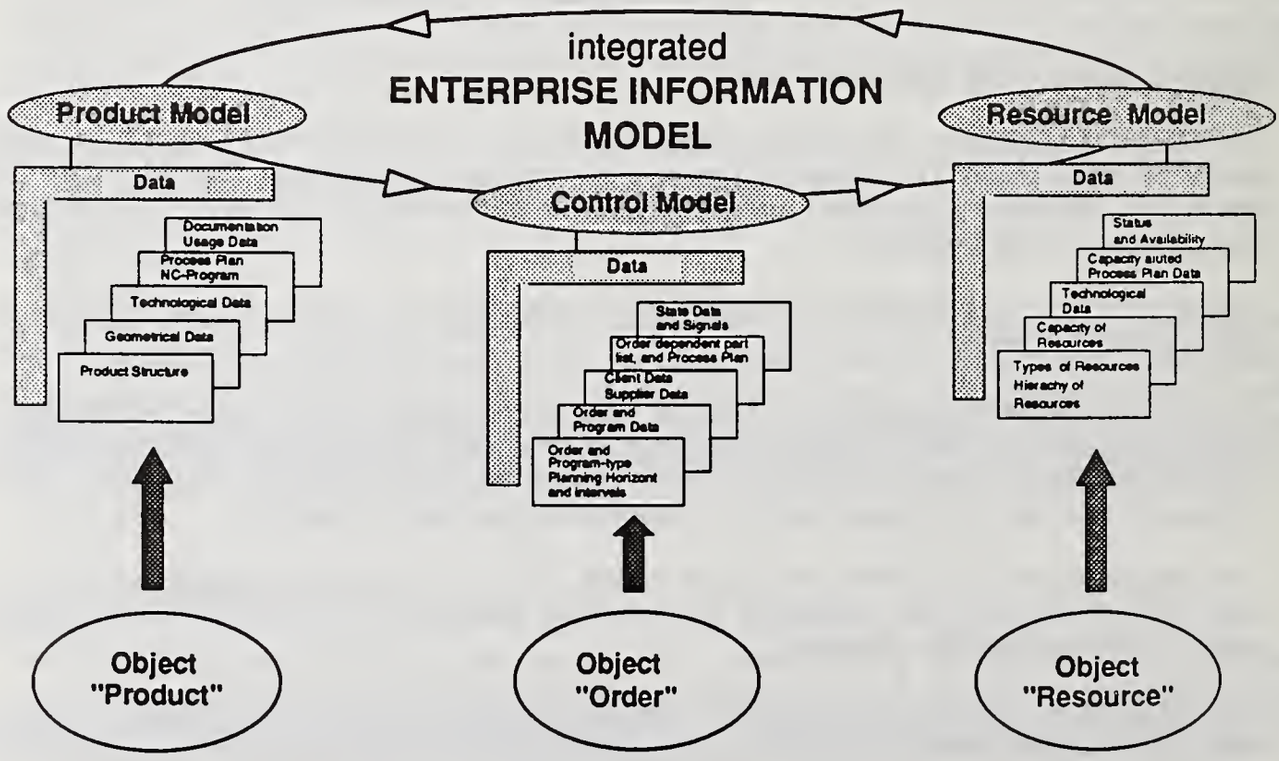


Figure 8: Enterprise Information Model

submodels is represented by layers which enable a grouping of the data of the objects by dif-

ferent criteria. This approach is similar to the concept of the STEP partial models for product data.

The information model approach enables to change from a EDP-system orientated data storage to a data storage which is related to the main objects of the manufacturing process. The independence of data from a specific EDP-system secures the extension and interchangeability between several systems. A particular enterprise information model will provide the preconditions for a general use of data bases and support the perception of priorities within data exchange.

The first step for the development of a particular information model of an enterprise is the cataloging of data and their related objects in "data dictionaries" as a uniform list of contents. Different kinds of lists were defined in the layers of the enterprise information model. The complete information model will be generated by relating the data of the different objects. Examples for relations are workpiece to workplan or NC-program to machine (logical), customs order to production order; bill of material for design to bill of material for production (temporal and organizational). Further references to related functions or to methods of objects in the function model have to be established and defined.

5.4 Integration of other models

As pointed out in chapter 4 nearly all modelling approaches use functions and data within their models and methods. The definition of **objects** which integrate functions and data provides the kernel of the manufacturing enterprise model as pointed out above. On this common basis the integration of other modelling methods and models could be achieved if the additional methods use the same functions and data as defined in the kernel model. In this way additional models could be derived from the kernel model as well as the kernel could be generated by the additional models.

To clarify this main feature of the "Integrated Information Modelling" concept in the next chapter a reference model for the enterprise related CIM-planning and introduction will be presented as an example.

6 Reference model for enterprise related CIM-planning and introduction

The enterprise related CIM-planning and introduction is characterized by some problems the reference model should help to solve. Only two should be mentioned:

- The initiative for the introduction of new CIM-components comes out from the different departments of an enterprise. Within those proposal normally only the requirements of the particular department were considered. Requirements which came out of a view of the whole enterprise were faded out.

- In the process of CIM-planning and introduction many people are involved. They need a common description of the whole CIM-concept to work into the same direction.

The reference model provides a structured description of the CIM-system support over the whole enterprise as well as relevant additional information. It provides a description of the CIM-status and should be used for the phase of analysis as well as for the different future stages of realization up to the target solution. The reference model for the CIM-planning and introduction

is based on the distinction of three general levels of consideration,

- the level of the functions of the manufacturing enterprise from the user`s point of view,
- the level of functions of data processing functions, i.e. processing, storage and retrieval, and transport of data,
- and the level of information technology that realizes the data processing functions.

The model comprises eight layers, which represent the interlinked fields of design within the enterprise related CIM-planning /SPU88, SPU89, SPU90/ (fig 9).

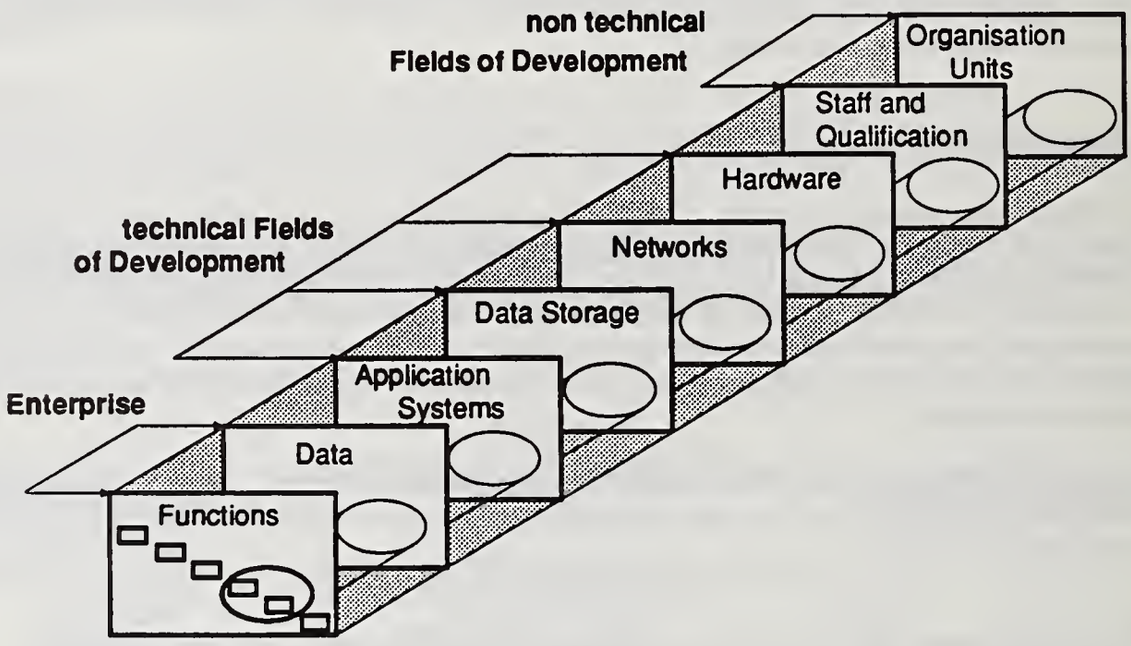


Figure 9: Reference model for enterprise related CIM-planning and introduction

The layers "functions" and "data" are derived from the manufacturing enterprise. They represent the manufacturing process itself and the required information processing independent from the technical solutions. They are equivalent to the kernel of the enterprise model presented above.

The layers "application systems", "data storage", "networks" and "hardware" are the technical fields of development within the CIM-planning. They represent the technical solution. The description of these subjects is also closely related to the first two layers respective to the kernel of the enterprise model. E.g. the "data storage" layer includes the relation of the data storage systems to the data as well as rules for the rights to access or modify the data and the responsibilities for the retaining.

The layers "organization units" and "staff and qualification" are non technical fields of design,

which have to be worked out simultaneously to the other fields. These layers contain additional items and subjects about the addressed aspects. All descriptions are related to the functions and data in the first two layers, that means to the kernel of the enterprise model.

All layers have to be designed by qualified methods. The tube through them shows the interdependency between them. It is implemented by the objects with their functions and data. The CIM-planning process normally starts with the functions and their organization. But also the existing conditions in the other levels are part of the planning. Restrictions concerning the technical capabilities and the historical growth of the enterprise have to be considered. These demands for the planning process caused by the mutual influences of the different layers could be handled in an efficient way by using the interlinked layers of the reference model.

This example shows how the kernel of the manufacturing enterprise model was extended by additional represented subjects, with an additional method for an additional purpose. In the same principle way several other models can be integrated into a common enterprise model.

7 Conclusion

The "Integrated Information Modelling" approach was developed for the integration of the different partial projects of the KCIM-project which will be continued until 1992. The essential difference in relation to other modelling approaches is the object orientation with its definition of three main object classes. The future work is determined mainly by the development and evaluation of the object structure to enable the deriving of information models by modelling the objects.

Another field of work is the definition of the relevant function elements, methods for their execution and data interfaces within the different domains of the manufacturing process. This work is done by some partial projects of the KCIM-project. The "Integrated Information Modelling" and the generic function elements could provide a common basis for the standardization of software for the computer integrated manufacturing, that means for the implementation of open CIM-architectures.

References

AMICE Proj. Team: CIM-OSA: Reference Architecture Specification. AMICE-Consortium Bruxelles 1987.

CEN/CENELEC: Framework for Modelling. Draft European Prestandard (ENV). DOC.: CEN/CENELEC/AMT/WG-ARC 1989-80, Denmark 1990.

DIN-Fachbericht 15: Normung von Schnittstellen für die rechnerintegrierte Produktion (CIM). Berlin, Köln: Beuth Verlag 1987.

Duelen, G.; Seliger, G.; Mertins, K.; Süssenguth, W.; u.a.: CIM-Grundsatzuntersuchung. Unveröffentlichte Studie. IPK 1986

Flatau, U.: Digital`s CIM-Architecture, Rev. 1.1. Digital Equipment Corporation, arlboro, MA U.S.A., April 1986.

Harrington, J.R.: Understanding the Manufacturing Process. Key to Successful CAD/CAM Implementation. Marcel Dekker, inc: 1984.

ISO: TC 184/SC4/WG1 DOC N 210. STEP

ISO TC 184/SC5 DOC N 148, Technical Report: Reference Model for Shop Floor Production, Part 1.

Krause, F.-L. Fortgeschrittene Konstruktionstechnik durch neue Softwarestrukturen. Vorträge des Produktionstechnischen Kolloquiums: Berlin PTK '86. Wien: 1986

Mertins, K.; Süssenguth, W.; u.a.: In Meins, W. (Hrsg): Handbuch Fertigungs- und Betriebstechnik. Organisation und Planung rechnerintegrierter Betriebsstrukturen CIM. Wiesbaden: Vieweg 1988.

Schwarz, K.: Manufacturing Message Specification (MMS). Die offene Verständigung verteilter Systeme in der industriellen Automation, ISO DIS 9506, Karlsruhe 1988.

Spur, G.: Optimierung der Fertigungssystems Werkzeugmaschine. München, Wien: Carl Hanser Verlag, 1972.

Spur, G.: CIM - die informationstechnische Herausforderung. Tagungsband zum Produktionstechnischen Kolloquium Berlin. Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik (IPK), Berlin 1986.

Spur, G.; Mertins, K.; Süssenguth, W.: Wege zu einem unternehmensspezifischen Referenzmodell der rechnerintegrierten Fertigung. ZwF 83 (1988) 10, S. 481-485.

Spur, G.; Mertins, K.; Süssenguth, W.: Integrierte Informationsmodellierung für offene CIM-Architekturen. CIM-Management 2/89, S. 36-42. München: Oldenbourg, 1989.

Spur, G.; Mertins, K.; Süssenguth, W.: CIM Management für Planung und Realisierung. IO-Management 6/90 Zürich, 1990.

Süssenguth, W.; Jochem, R. Rabe, M Bals, B: An Object Oriented Analysis and Design Methodology for Computer Integrated Manufacturing Systems. Proceedings TOOLS '89, November, 13-15, 1989, CNIT Paris, France.

Wieneke, B. Rechnerunterstütztes Planungssystem zur Auslegung von Fertigungsanlagen. München: 1987

Yeomans, R.W.; u.a.: Design Rules for Computer Integrated Manufacturing Systems. Projektdokumentation ESPRIT-Projekt 34, 1984.

A Systems Theoretic View of Computer Integrated Manufacturing

Frank P.M. Biemans and Chris A. Vissers

March 12, 1990

Abstract

The global and specific characteristics of a CIM architecture have far reaching consequences for the general health of the production organization that applies the architecture. A theory is therefore needed to design "good" CIM architectures.

We argue that good CIM architectures specify unambiguously, at a high level of abstraction, and in generic terms a production organization as a configuration of components, while allowing us to understand how the components affect the performance of the production organization as a whole.

We demonstrate a theory to design such CIM architectures for a production organization that, in addition to realizing production targets, can honor requests to change its product portfolio, production capacity, or production costs.

1 Introduction

We all witness the frequent publication of CIM architectures. The abundance of distinct architectures [AMBF83,PW78,Coh88,SME88,NBS85,BV89,JM84,ISO86] and the typical absence of any justification for a specific proposal may suggest that any CIM architecture is workable and that no justification is required. However, as we hope to show, the specifics of a CIM architecture do have far reaching consequences for the general health of the production organization that applies the architecture. A theory is therefore needed to design a "good" CIM architecture and to assess its "quality". This paper describes and demonstrates such a theory.

To demonstrate that the specifics of an architecture do matter, we discuss some of the undesirable effects of typical CIM architectures observed in practice or published in literature. We then develop an explicit statement regarding the *purpose* of a CIM architecture and the *criteria* that it should satisfy –a statement that is noticeably absent in the majority of the proposals for CIM architectures. We then introduce an approach to develop a CIM architecture that satisfies the proposed purpose and criteria. Finally, we demonstrate the approach. First, we review in detail the development of a CIM architecture of a "Cell/Line", a component of a production organization. Second, we review the major steps taken to develop a CIM architecture for an entire production organization.

2 Properties of CIM architectures

Let us temporarily assume that the purpose of an "architecture" is to show how a production organization can be controlled. In other words, an architecture is a design that, when fully implemented,

will allow a production organization to adequately respond to its production targets.

A common, though not universal [Bak89], denominator of CIM architectures is their hierarchical structure of "controllers". These controllers execute various manufacturing planning and control tasks and are interconnected so that they can exchange messages. Controllers at the bottom of the hierarchy steer equipment, whereas controllers at higher levels of the architecture are concerned with "higher level" tasks such as "planning". This is roughly where the commonality of many architectures ends; their specific characteristics are different. We will review some of these specific characteristics and show how they affect the effectiveness, flexibility, and complexity of the architecture.

2.1 Allocation of tasks

Let us investigate the effects of the distribution of tasks across levels in the hierarchy of a CIM architecture. Consider the situation illustrated in Figure 1. A transport system connects multiple workstations. Parts enter the transport system and are transported to some assembly workstations. Subsequently, they are transported to a couple of workstations to undergo quality tests. If any of these tests is negative, they are transported to a repair workstation and, then, to a test workstation again. If all tests of a part are positive, it exits the transport system.

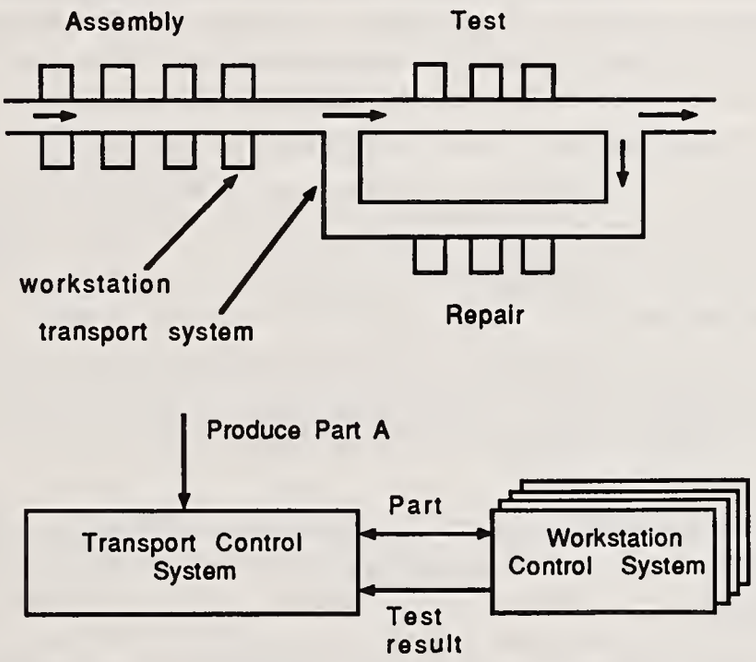


Figure 1: A production line and its control system

Let us now consider a control architecture, illustrated at the bottom of Figure 1, for the system just described. As Figure 1 illustrates, each workstation has its own control system that allows it to recognize a part and to execute a pre-determined operation on the part. The transport system has its own control system as well. This control system accepts commands to produce (final) parts, determines which workstation each part has to visit, and steers the conveyor belts to bring parts to their destination. It also accepts information about the part quality from test workstations and can thus decide whether to transport a part to a repair workstation, to a test workstation, or to

the exit point of the transport system.

With this control architecture, parts will get processed, tested, and possibly repaired. However, the architecture imposes a rigid control scheme, incapable of flexibly accommodating changes in products, operations, or layout. We discuss three causes of the limited flexibility.

1. The transport control system selects the "route" of workstations for a part. As a result, changes in products, operations, and workstations cannot be isolated from the transport control system. The interdependencies of changes will discourage the making of changes at all and thus limit the flexibility of the entire production organization. Consider, for example, the introduction of a new product. It is inevitable that some workstations be re-programmed to execute the new operations. But, due to the proposed architecture, the modifications cannot be limited to the workstations; the transport system has to be instructed about the route of the new products, and, if the workstations have to be re-balanced, about the changed routes of the existing products.

   Similarly, the transport system has to be informed if a workstation's position is changed, or if a workstation is modified to execute new operations.

2. A workstation has to recognize a part to execute a pre-determined operation. Consequently, the part type determines which operation is executed; a workstation cannot be asked to execute a specific operation on a part. Thus, it cannot, for example, be asked to populate a bare Printed Circuit Board with one set components, or, with another set of components, to produce different PCBs depending on their demand.

   Similarly, it is not possible to select a workstation or operation for a part on the basis of the current load to the workstations. The transport system, which selects workstations and operations, has no information about the load and capacity of the workstations.

   Also, the fact that each workstation has to recognize a part to determine which operation should be executed implies that each workstation has to be equipped with part recognition capabilities, such as bar code readers. This can be quite expensive. Moreover, a workstation cannot start an operation before a part has arrived. Hence, it cannot start changing tools before a part has arrived to reduce the effective idle time when it has to change over to other parts.

3. The transport system needs to be informed about the test results by a few, specific workstations. As a result, the transport control system has to be updated if the location of a test workstation changes.

The disadvantages listed above are caused by the specific allocation of tasks to the components of the control system. Consider the control system illustrated in Figure 2, which has a different allocation of tasks. In this case, the workstations execute the operation that is explicitly requested rather than the operation that is implied by the type of arriving parts. A workstation accepts a request to execute operations on certain parts and processes the parts as requested when they arrive. The transport system is much like a workstation: it transports parts from one location to another on the basis of explicit requests. The responsibility to select the route of parts and the operations and to request for operations and transportation is allocated to a third control system, the coordinator.

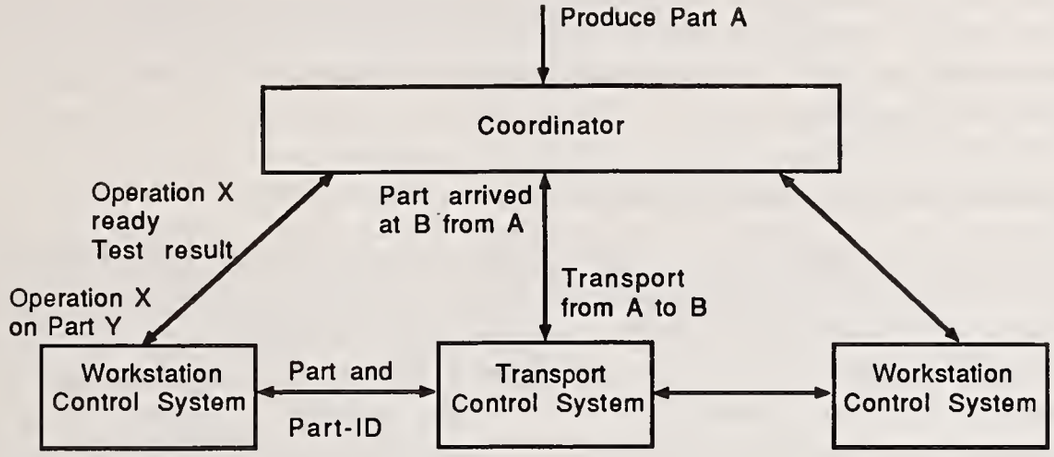


Figure 2: Alternative, more flexible control system

The transport control system reports whether a part has reached location B from A. The workstation control system reports whether an operation is ready, and may report test results. Transport system and workstation can exchange parts and can provide the recipient of a part with the part identification. For each part, and depending on test results and on the operations that the part has undergone already, the coordinator determines which workstation a part should visit and which operation it should undergo.

In this control system, the effects of various changes are limited. Changes in the layout of the transport system primarily affect the transport control system, changes in operations primarily affect the workstation control system, and changes in the routes of parts primarily affect the coordinator.

Let us quickly contrast this control system with the one described earlier. To introduce a new part, the coordinating controller has to be instructed which operations the part should undergo and at which workstation. But the transport system need not be changed at all; it continues to transport parts, on request, from location A to B. Further, the transport system is insensitive to changes in the operations executed by workstations due to the absence of operation dependent interactions of transport system and workstations. A workstation can also execute multiple operations on a part depending on the commands it gets. Moreover, the coordinator can select workstations and operations depending on the load of transport system and workstations since it is informed about their status and can give commands at the latest possible moment. If alternative operations exist for a part, the coordinator can select one of them and command a workstation to execute the selected operation. To continue the enumeration of advantages of the control system, it is not necessary that each workstation has part recognition equipment. The reason is that the workstations and transport system can track the parts when they travel through their own domain and communicate the part identification when they exchange the parts. Further, a workstation that receives an operation command before a part arrives, can already start the necessary change over procedures. Finally, it is not required to update the transport system if the test workstation is relocated since the all workstations interact with the transport system in a similar manner irrespective of their position and operations.

We conclude that the architecture of Figure 2 is less complex and more flexible than the one of Figure 1. This is largely caused by the specific allocation of tasks in the architectures. *Given the formidable, inherent complexity of manufacturing control, we cannot afford CIM architectures that introduce unnecessary complexity.*

2.2 Precision of architecture definitions

A CIM architecture is supposed to prescribe how a production organization can be controlled. The architecture is to be interpreted and implemented to build a real, say computerized, control system. Obviously, the architecture need to be defined unambiguously, lest different interpretations lead to incompatible control systems. And yet, many published architectures are far from precise. They assign, for example, "long-term planning" and "short term planning" to different control levels. But what is long term or short term? And what is to be planned? Undoubtedly, these questions will be answered differently by different factories, or, for that matter, by different people in the same factory.

CIM architectures have to be unambiguously defined to be practically relevant.

2.3 Generality of a CIM architecture

One could argue that the definitions of long term and short term, discussed in the previous Section, could easily be made more precise. Long and short term, for example, could be defined as referring to planning horizons longer and shorter than one month respectively. However, such definitions have a limited applicability: different factories use different horizons, depending on the products they make, the markets they serve, etc.

By contrast, consider the distinction between "production planning" and "scheduling", whereby production planning is pro-active and scheduling reactive with respect to demands for parts. Production planning occurs on the basis of stochastic forecasts of demand. The purpose of production planning is to determine which jobs should be executed to create inventories of intermediate parts. These inventories help a production organization dispatch products quickly if the forecast demands materialize. Thus, production planning is characterized by balancing the risks of being unable to satisfy demand and of producing undesirable inventories. It is a pro-active measure to prepare a production organization for expected request for parts. By contrast, scheduling is reactive with respect to requests for parts. Scheduling occurs on the basis of deterministic demand, i.e. the jobs, which have been selected as a result of the production planning. It amounts to planning which workstations to use, when, and for which operations to realize those jobs in time. It is not relevant whether a job is executed to realize firm or forecast demand; it has to be executed as a result of the production planning decisions. The purpose of the scheduling is to realize the jobs timely and efficiently.

The distinction between pro-active and reactive planning is more generic than the distinction of long and short term planning. *CIM architectures should be specified generically to ensure their applicability across different production organizations or to ensure their applicability in a single production organization, which is likely to change over time.*

2.4 Level of abstraction of a CIM architecture

Architectures can be described at several levels of abstraction. The level of abstraction should always be stated, lest the architecture be interpreted wrongly. Take the architecture illustrated in Figure 2. Unless their level of abstraction is explicitly defined, the boxes in this Figure could be interpreted as abstract decision making processes, as physical computers, or human beings, etc.

Abstract decision making processes are described by their function or task, irrespective of their physical implementation. They can be physically implemented in numerous ways, depending, for example, on the required costs and performance of eventual physical system. To illustrate the concept of abstraction, consider a factory. One can view it as an abstract process, accepting raw materials and production targets, and dispatching products. One can also view it as a physical system, i.e. as a building with equipment.

Both, descriptions of abstract functionality and physical implementation, pay an important role in the design of systems. But, the functionality has to be established before a physical implementation can be undertaken [BB82,Bro75]. *Abstract CIM architectures, describing control components in terms of their tasks and interactions, should form the basis for a physical implementation of a control system.*

We have discussed the relevance of specific characteristics of CIM architectures. However, many publications propose different architectures without justifying why certain characteristics were chosen to be different. This may be caused by a lack of theory addressing the purpose of a CIM architecture, the criteria it should satisfy, and strategies to develop one.

3 Purpose of a CIM architecture

We propose to focus on the "integrated manufacturing" of "CIM" when defining the purpose of a CIM architecture. In other words, the purpose of a CIM architecture is to show how a production organization can be integrated, i.e. composed of components. *A CIM architecture should describe a production organization as a configuration of interacting components and show how these components affect the performance of the overall production organization.*

The above purpose of a CIM architecture is practically relevant. What ultimately counts is the "bottom line", overall performance of a production organization, say, the variety and frequency of the production targets that it can realize. Such performance measures apply to a production organization as a black box, as illustrated in Figure 3. But how to realize the overall performance targets? A production organization is an intricate fabric of many people and systems with a variety of responsibilities in materials management, product design, scheduling, etc. It is not clear how their individual behavior, or the improvement thereof, affects the performance of a production organization as a whole. An architecture is therefore needed that shows how the components relate to the whole.

To illustrate the difficulty of controlling a production organization so that all its components harmoniously contribute to the overall organization, let us discuss one of the many coordination problems in a production organization. Typically, the maintenance department is encouraged to frequently halt a machine for service whereas the foreman is encouraged to skip maintenance so that the machine is longer available for production. Thus, a conflict of interest exists between maintenance and production. Local improvements of maintenance or production will intensify the

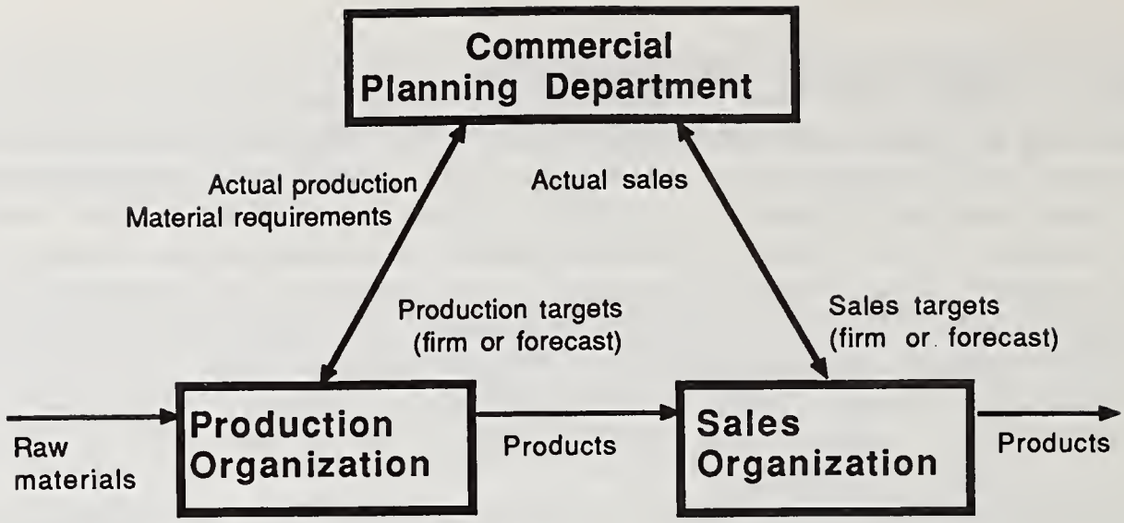


Figure 3: Production organization as a black box

conflict and, most likely, will not fully contribute to improvements of the production organization as a whole. Coordination problems, like the one discussed, are not readily visible but, nevertheless, can lead to damaging sub optimizations.

In addition to coordination problems, production organizations suffer from the fact that it is often difficult to assess whether local improvements pay off for the entire organization. Take the improvement of a warehouse so that its response time to dispatch-requests be reduced. This improvement may fail to improve the performance of the production organization as a whole if it leads to increased output of the warehouse that causes congestion of production lines and that therefore increases the overall throughput time.

To summarize, one of the essential problems in improving the overall efficiency and flexibility of production organizations is to determine how local control activities affect the overall organization. A CIM Architecture that reveals the effect of local control activities on the overall performance of a production organization would be most useful.

To further de-emphasize the use of computers and to adhere to international terminology [ISO83], we refer to a CIM architecture as a *"Reference Model for a Production Organization"*.

Level of Abstraction. As mentioned before, we focus on the required function or task of a production organization and its components before considering their physical implementation. A Reference Model should, therefore, depict a production organization in terms of a structure of interacting components and in terms of the tasks of these components.

Note that an "abstract" Reference Model can be used in multiple ways. It can be used, for example, as a specification of the functionality of a computerized factory control system; an implementation process should produce the eventual physical system. A Reference Model, describing tasks and information flows in a production organization, can also be viewed as a normative template of a non-automated or partially automated production organization, which, while unlikely to be fully implemented, can serve as a reference for the evolution of existing organizations or as a basis for their evaluation [ea89,CMvS+88].

Scope. We should also define the scope of a Reference Model. What is the system for which a Reference Model needs to be designed? In this paper, we take the black box production organization depicted in Figure 3 as our scope.

4 A systematic approach to design a Reference Model

A Reference Model should show how components of a production organization affect the overall production organization. We should, therefore, first determine the required task of the overall organization and then identify components that can cooperate to realize the overall task.

Black Box Production Organization. With reference to Figure 3, we can describe the overall task of a production organization in terms of its interactions with its environment. It negotiates with a Commercial Planning Department to establish firm and forecast production targets. It outlines its material requirements, reports about the progress of the actual production, accepts raw materials, and dispatches products according to the production targets.

Decomposition. To identify the components of the production organization, we have to decompose the overall organization [Bie89]. A black box as depicted in Figure 3 can be decomposed in numerous ways. However, we are only interested in decompositions that yield an easy to understand configuration of components since the result of the decomposition, the Reference Model, should allow us to understand how components affect the production organization as a whole.

Interaction patterns of components can become very complex depending on the structure of the decomposed system. We should therefore pursue decompositions that produce systems with components that can be viewed as distinct entities in the context of the overall system. Such systems have a *'natural'* structure and therefore the conceptual integrity and unity of viewpoint that allows a human to master them as a whole despite its complexity [BB82].

Separation of Concerns. To obtain a structure with distinct components, we apply the orthogonality concept: *separate independent concerns* [Par72,VL86]. Relatively independent tasks require different kinds of information, except for the moments that information is to be shared through interaction. Such tasks can thus be assigned to distinct components that interact sparingly.

Separation of concerns tends to be a difficult design activity. The decomposition should not deprive certain components of information needed to do their tasks. Separation of concerns therefore requires a deep insight as to whether certain information is relevant for certain tasks. To assess the relevance of particular information for particular tasks, one needs to know how these tasks are executed and how they affect each other. Typical text books, which discuss such tasks as inventory management, scheduling, product design, and process planning, in isolation rather than in each other's context are of little help.

To identify the tasks to be executed by system components, we should analyze the interactions of a system and its environment: which tasks should be executed to generate and process the information exchanged in the interactions?

Interaction. Decomposition is intimately linked with "interaction" since the components of a decomposed system interact to realize the overall task. We use "interaction" in a non-traditional

way. It does not refer to the traditional input-output concept but denotes, in more abstract terms, mutual exertion of influence. In an interaction, system components *negotiate* to establish values that satisfy their private constraints [Bri85]. However, we abstract from these negotiations and describe interactions in terms of their results, the established values shared by all components involved. Take the production target interaction illustrated in Figure 3. This interaction establishes production targets that are agreed upon by the production organization and the commercial planning department; it does not reveal any negotiation that leads to these targets. If required, one could describe the negotiations explicitly by replacing abstract interactions with more detailed interactions.

We will now illustrate the approach to develop a Reference Model.

5 Illustration of the approach

Rather than discussing the decomposition of an entire production organization, we discuss the decomposition of a component of a production organization, a "Cell/Line", in some detail.

Cell/Line. We define a Cell/Line as a component of a production organization that can accept commands to execute jobs in certain time slots. Jobs are material processing activities that lead to the assembly, disassembly, storage, or dispatch of parts. The parts needed to execute a job ought to be available for a Cell/Line at the beginning of the time slot in which it has to execute the job. They may be supplied from outside the Cell/Line, or may be kept in storage by the Cell/Line itself. Figure 4 illustrates a Cell/Line.

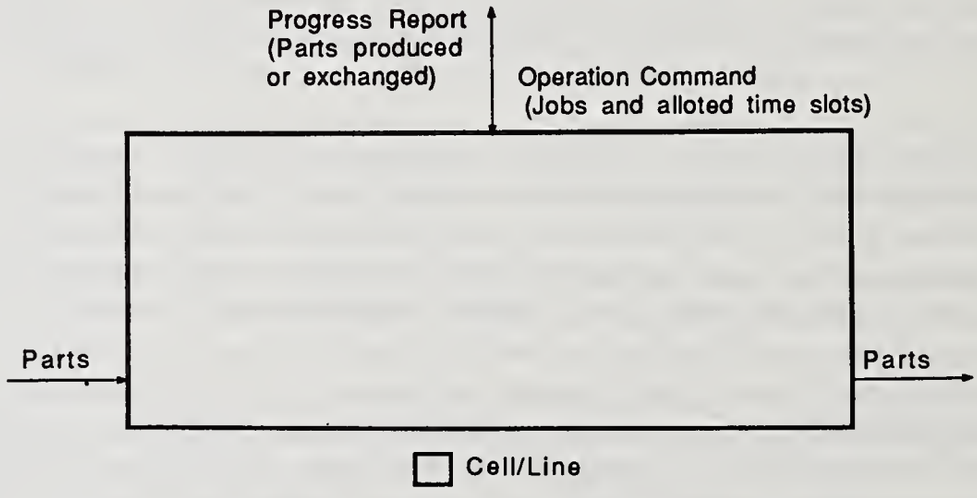


Figure 4: A Cell/Line

A Cell/Line may be requested to execute a variety of jobs to process different parts, in different quantities, and in different time slots. We require that a Cell/Line can commit to certain jobs: its acceptance of a command to execute a certain job in a certain time slot implies a commitment to realize the job. Table 1 lists some interactions that allow one to command a Cell/Line to execute certain jobs and to retrieve information about the progress of the execution of jobs. The Operation

Command interaction of Table 1 establishes a commitment to execute a job in a certain time slot, but does not reveal the negotiations of Cell/Line that lead to this commitment.

Interaction	Arguments	Explanation
Operation-Command	Parts, Partner Parts Parts, Partner Parts Time-slot Command-Id	Accept *parts* from a *sender* (e.g. another Cell/Line or Supplier), and *parts* already in stock, and transform them into *parts* to be dispatched to a *receiver*, and *parts* to keep in stock, in a *time-slot*. *Identification* for future reference.
Progress-Report	Progress Command-Id Parts, Partner Parts Parts, Partner Parts Time-Slot	Report about the *progress* of the execution (for example 'completed', 'busy', or 'waiting'.) of a previously-received *command*. Which *parts* have been accepted from a *sender*, and together with *parts* already in stock been transformed into *parts* dispatched to *receivers*, and *parts* kept in stock, in a *time-slot*.

Table 1: Interactions of a Cell/Line

Note that a Cell/Line as discussed here is not just a collection of machines. Rather, it is an abstract "black box" that provides a service, i.e. the execution of jobs. We do not know yet how this black box will be internally constructed. It is reasonable, of course, to assume that the physical machines will be among its components, but it will also contain components responsible for the coordination of those machines.

Analysis of Interactions. We now analyze the interactions of a Cell/Line and its environment to identify which relatively independent tasks the Cell/Line should execute. We focus on the ability of a Cell/Line to accommodate requests to produce varying types and quantities of materials, in varying time slots.

To execute jobs, a Cell/Line needs resources that can process material. We call the material processing activities of these resources "operations". Consider two assumptions regarding these resources:

1. multiple resources are utilized to execute a job so that resources process material and pass it to another resource for further processing; and

2. most resources can execute several operations.

Such resources would allow a Cell/Line to be efficient and flexible. Efficient, because it can combine the operations of relatively few resources to execute a relatively large number of different jobs. Flexible, because it can exploit the versatility of the resources to accommodate varying kinds of jobs.

To actually realize the potential efficiency and flexibility, however, the Cell/Line has to *schedule* the operations, i.e. determine which operation should be executed, when, and by which resource. Note that we view an *"operation"* as a material processing activity that can be scheduled in a useful

manner; it cannot be split into material processing sub activities that can be scheduled in a more useful manner.

Separation of Concerns. A natural division of tasks presents itself, i.e. the *scheduling* and *execution* of operations. We could thus envision a Cell/Line consisting of two kinds of components. A first kind of component accepts requests to execute jobs, determines which operations are to be executed to realize a job, and schedules the operations. A second kind of component executes the operations. Let us now explore whether scheduling and execution of operations are sufficiently independent to be separated.

A component given the responsibility to schedule operations, would need to know the:

- jobs to be executed, and their time slots;
- availability, capacity, and capability of resources;
- availability of parts; and
- costs and precedence relations of operations.

Some of these data tend to vary slowly during the time a Cell/Line has to execute a job. Consider, for example, the:

- precedence relations of operations, which depend on the design of parts;
- costs of operations, which primarily depend on the material characteristics of parts and on the technological implementation of the resources; and
- capacity and capability of resources, which primarily depend on the technological implementation of the resources and material characteristics of parts.

Because they change relatively slowly, these data can be viewed as given constraints for the Cell-/Line component that has to schedule operations rather than as data to be provided by other components.

By contrast, the list of jobs to be executed and their time slots may change more often since new jobs may be requested at any time. Similarly, the availability of parts and resources cannot be viewed as given constraints during the generation of a schedule. These availabilities depend on the timing and allocation of operations, and therefore on the generation of a schedule. Moreover, resources may fail to produce the required materials so that it is important that the actual availability of materials and resources be known when generating or executing a schedule.

To conclude, if a Cell/Line component were to schedule operations, it would have to consider the jobs to be executed, have to schedule operations to realize these jobs, and have to be informed whether execution of the schedule results in the desired availability of parts and resources. We concluded earlier that an operation should be viewed as a unit to be scheduled; splitting the operations and scheduling the split operations has no effect on the performance of a Cell/Line. It is therefore not important to know how an operation is executed, which sub tasks realise an operation, to be able to schedule it. Consequently, the scheduling task can be separated from the task to execute operations provided that the execution of operations results in information regarding the availability of parts and resources. Hence, we have identified the following, relatively independent, globally defined Cell/Line tasks:

- **Scheduling of Operations:** determine which operations on parts should be executed, by which resource, and when to realize jobs in committed time slots; and
- **Execution of Operations:** accept parts, execute operations on them, dispatch the processed parts, and report about the availability of parts and resources.

Decomposition. Applying the techniques discussed in Section 4, we decompose a Cell/Line into two distinct components and assign the relatively independent scheduling and execution tasks to them. We call these components "Cell/Line Controller" and "Workstation" respectively.

Figure 5 illustrates how Cell/Line Controller and Workstations constitute a Cell/Line. Table 2 lists some interactions of Cell/Line Controller and Workstations. These interactions allow a Cell-/Line Controller to command a Workstation to execute a given operation and also to acquire the information regarding the availability of materials and workstations.

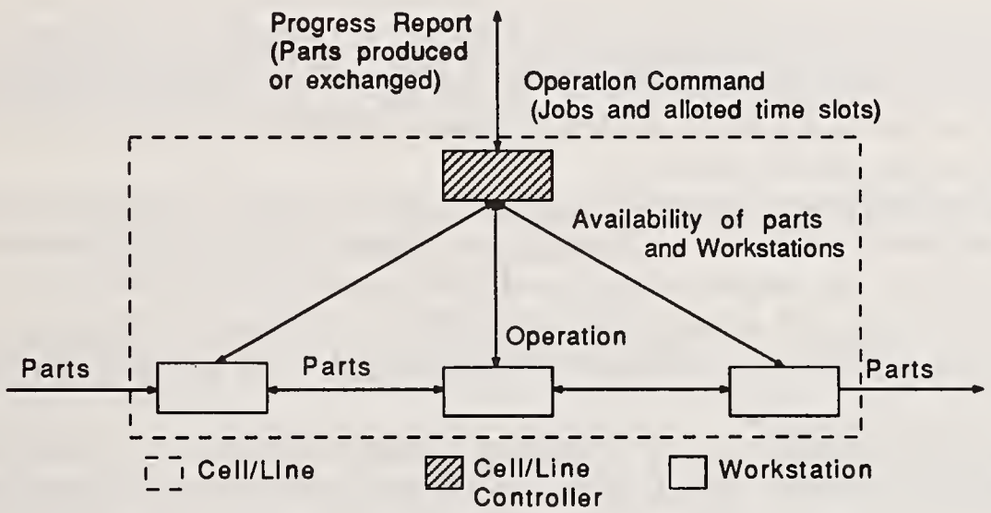


Figure 5: Internal organization of a Cell/Line

Note that operation commands for a Workstation, unlike those for a Cell/Line, do not contain time slots as parameters. The reason is that the duration of an operation can be predicted with a reasonable accuracy. Once materials and Workstation are available, the operation can start and takes a more or less given amount of time. By contrast, the time it takes a Cell/Line to realize a job depends on the way it schedules operations. The schedule will be different each time a Cell/Line has to execute a different combination of jobs. Hence, it cannot simply be assumed that a job be executed in a given amount of time. Rather, the time slot for a job need to be established explicitly when the Cell/Line accepts a command to execute the job.

General Requirements for Cell/Line Controllers. The scheduling task of a Cell/Line Controller may imply a variety of sub tasks to ensure that operations be executed by Workstations as scheduled. We developed specifications for a Cell/Line Controller [BS89], for example, that can ensure that the flow of parts such as screws, picture tubes, and coils, is coordinated with the flow of parts with which they are to be assembled; determine the sizes of batches and series of

Interaction	Arguments	Explanation
Operation-Command	Parts, Partner Parts Parts, Partner Parts Command-Id	Command to accept *parts* from a *sender* and together with *parts* in stock transform them into *parts* to be dispatched to a *receiver* and *parts* to keep in stock, Command *identity* for future reference.
Progress-Report	Progress Command-Id Parts, Partner Parts Parts, Partner Parts	Report about the *progress* (for example 'completed', 'waiting', or 'busy'.) of a *command*: which *parts* accepted from a *sender*, and which *parts* in stock were transformed into *parts* dispatched to a *receiver* and *parts* kept in stock.

Table 2: Interactions of Cell/Line Controller and Workstation

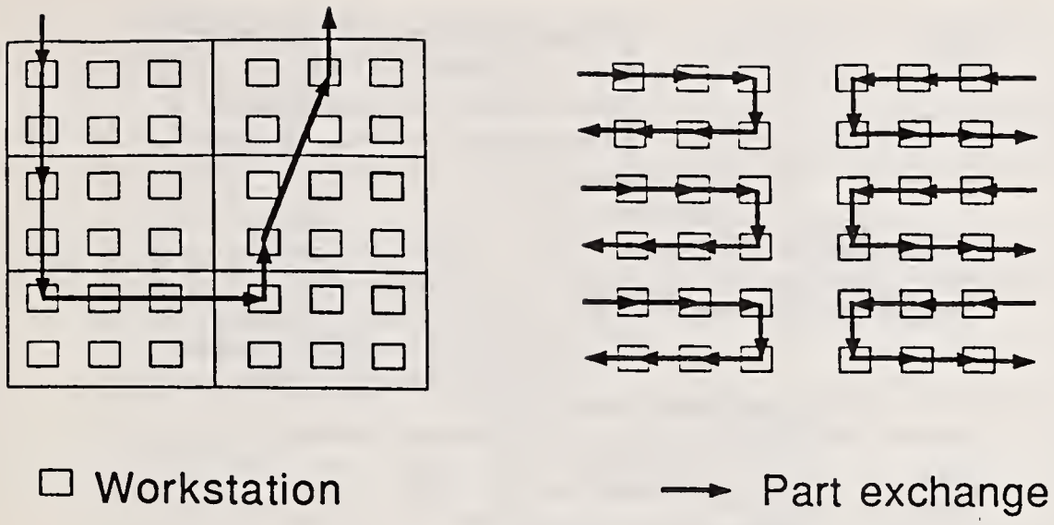
parts processed by Workstations; determine which of the parts waiting for a Workstation should be processed first; determine when the execution of a job should start; determine whether parts that fail tests executed by Workstations should be repaired by repair Workstations; ensure that 'buffer Workstations' contain sufficient parts.

Note that the Cell/Line Controller is an abstract system. We have not determined whether it should be a computer or a human or a hybrid solution. The eventual Cell/Line Controller would look different depending on the solution being pursued. Computerized solutions may need, for example, interactive Gantt charts and a wide variety of displays of performance measures for alternative schedules to allow humans to override the schedules proposed by the Cell/Line Controller.

Layout of Workstations. As described in Reference [Bie89], a production organization may have multiple Cell/Lines. They should be able to execute their jobs independently from each other: the capacity of one Cell/Line to execute jobs should not be affected by the execution of jobs by another Cell/Line. In reality, however, Cell/Lines do affect each other's capacity since they exchange parts. Nevertheless, the required independence can be realized if Cell/Lines exchange parts only sparingly. The Workstations of a Cell/Line should therefore form 'self-contained' groups, which exchange parts amongst themselves much more frequently than with Workstations of other groups.

The relative frequency with which Workstations exchange parts should therefore determine whether they belong to the same Cell/Line. In practice, however, it is often assumed that the geographical configuration of Workstations determines whether they are grouped together in Cell-/Lines [PW85].

We have seen situations as illustrated in Figure 6, where Workstations were allocated to the same Cell/Line because they were located in the same room. However, the Workstations within one room did not exchange parts; parts visited Workstations in different rooms. Hence, it would make more sense to allocate the Workstations that exchange parts to one Cell/Line, despite the fact that they are in different rooms.



The left picture shows Workstations that are grouped together for environmental reasons. They have, for example, different requirements regarding the cleanliness of the air, like in Integrated Circuit manufacturing facilities. The right picture shows Workstations that are grouped together because they exchange parts frequently.

Figure 6: Self-contained groups of workstations exchanging parts, be they geographically close or dispersed.

We have reviewed the development of a Reference Model for a Cell/Line. The model has a natural structure, is generic, abstract, and precise, and therefore meets the requirements discussed earlier in this paper. The model also shows how components of a Cell/Line affect the behavior of a Cell/Line as a whole. The strategy applied to develop a Reference Model for a Cell/Line, i.e. the analysis of interactions, decomposition, and separation of concerns, can also be applied to develop a Reference Model for an entire production line.

6 A Reference Model for an entire production organization

We have applied the design techniques discussed above to develop a Reference Model for an entire production organization [Bie89]. We briefly review the result and provide pointers to detailed descriptions.

To cope with the formidable complexity of a production organization, we designed the Reference Model in two phases. First, we made a simplifying assumption that the production organization can only manufacture products according to a given set of production targets. This *'basic'* production organization, already illustrated in Figure 3, cannot change its product portfolio, production capacity, or production costs.

Second, we developed a Reference Model for a 'flexible' production organization. A flexible production organization, illustrated in Figure 7, has adaptive capabilities to change the way it executes its basic tasks. Thus, it can honor requests to change its product portfolio, production capacity, or production costs.

As we will explain later, a Reference Model for a basic production organization is a spring-board

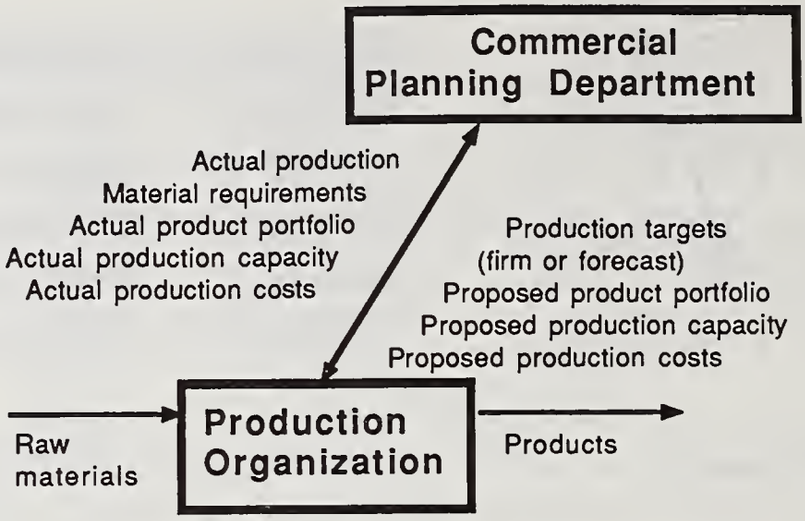


Figure 7: A flexible production organization

for the design of a Reference Model for a flexible production organization.

Reference Model for a Basic Production Organization. Application of the design techniques discussed in Section 4 to the black box production organization illustrated in Figure 3 leads to the Reference Model illustrated in Figure 8. Note that this Figure merely illustrates the Reference Model; the Model itself is described extensively in reference [Bie89].

As Figure 8 illustrates, the Reference Model portrays a basic production organization as a layered structure of 'controllers', which are concurrently operational and negotiate through interactions. Each controller (e.g. Factory Controller) views the components (e.g. Cell/Line) below it as a black box; it does not need to know the internal structure of these components (e.g. Cell/Line Controller and Workstations).

The Reference Model distinguishes controllers on the basis of their tasks. Its structure is generic because the analysis used to develop the structure did not assume any specific characteristic of a specific production organization, or specific products. Similarly, the task definitions are generic. They only describe the most essential characteristics of the controllers' tasks.

To summarize, the Reference Model provides us with a mental image of the decision making apparatus that constitutes a production organization. The model is abstract and generic. Due to its abstraction, it focuses on the decision making in a production organization rather than on the physical means that can make these decisions. Due to its generic nature, it can be used to model a wide variety of production organizations.

To model a specific production organization, we must choose specific tasks and structure for the controllers of the model without violating the generic constraints. We choose specific tasks by defining how they are executed. We choose a specific structure by defining how many controllers exist at each layer and which controllers may interact.

But choosing the specific tasks and structure is an important decision making process in production organizations that have modify their capacity or product portfolio to adapt to changing market requirements. This kind of decision making is not modeled in the Reference Model for basic

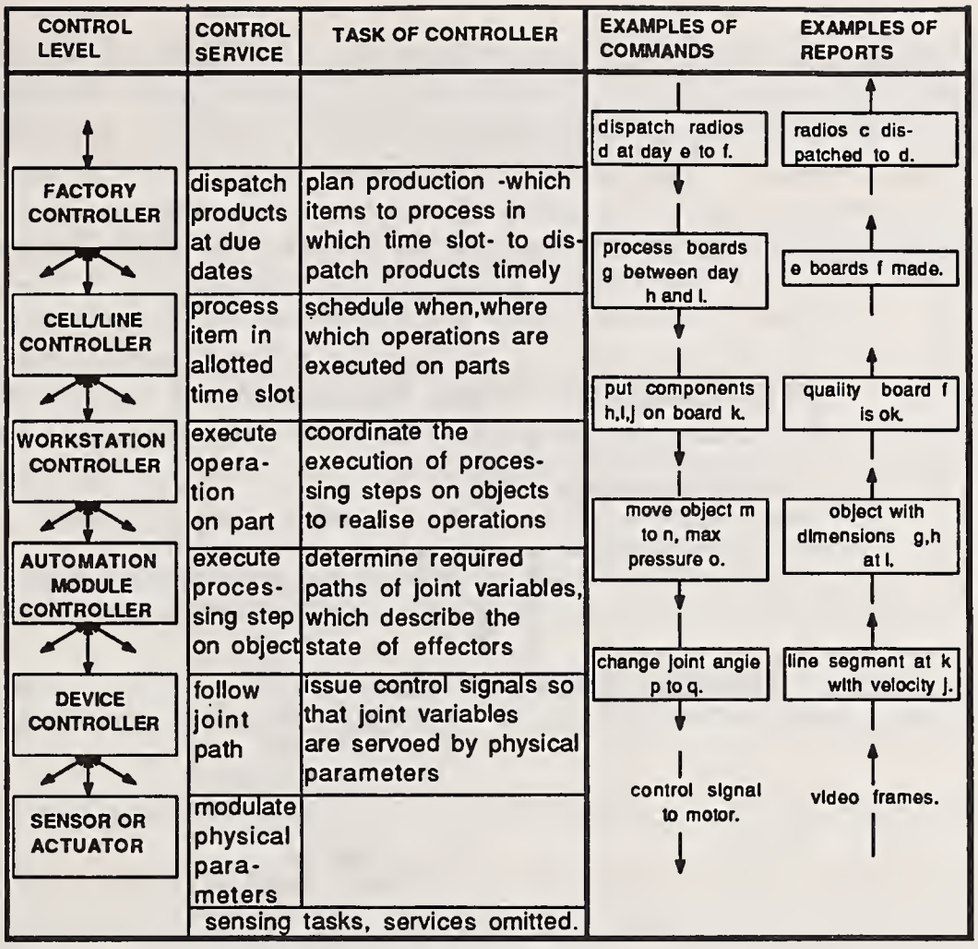


Figure 8: Reference model of a basic production organization at a glance

production organizations; it will be part of the Reference Model for flexible production organizations.

Flexible Production Organization. A flexible production organization has to honor requests for changes in its "application", i.e. its product portfolio, production capacity, or production costs. Like a basic production organization, it should execute the basic manufacturing task, the manufacturing of products in a given application. However, unlike a basic production organization, it should also be able to adapt the way it executes the basic task to requirements imposed by changing applications.

A natural separation of concerns presents itself: the concern to execute the basic manufacturing task and the concern to adapt the way this task is executed. We assign the responsibilities for these concerns to an "Executor" and a "Manager" respectively. The Manager and Executor interact to adjust the way the Executor performs its basic manufacturing task. We will discuss their interactions in a moment.

Structure of an Executor. The structure of an Executor and the structure of a basic production organization, illustrated in Figure 8, could well resemble each other. The reason is that both, Executor and basic production organization, execute the basic manufacturing task. However,

the Executor's structure has to be adjusted specifically for each application, whereas the structure of a basic production organization is a generic one. It is generic in the sense that it does not presuppose a specific configuration, apart from the layered configuration of controllers, or specific tasks, apart from the global characteristics of the tasks. As Figure 9 illustrates, we therefore propose that the Executor exhibit the generic characteristics of a basic production organization, but allows the Manager to make application-dependent adjustments in its configuration and tasks.

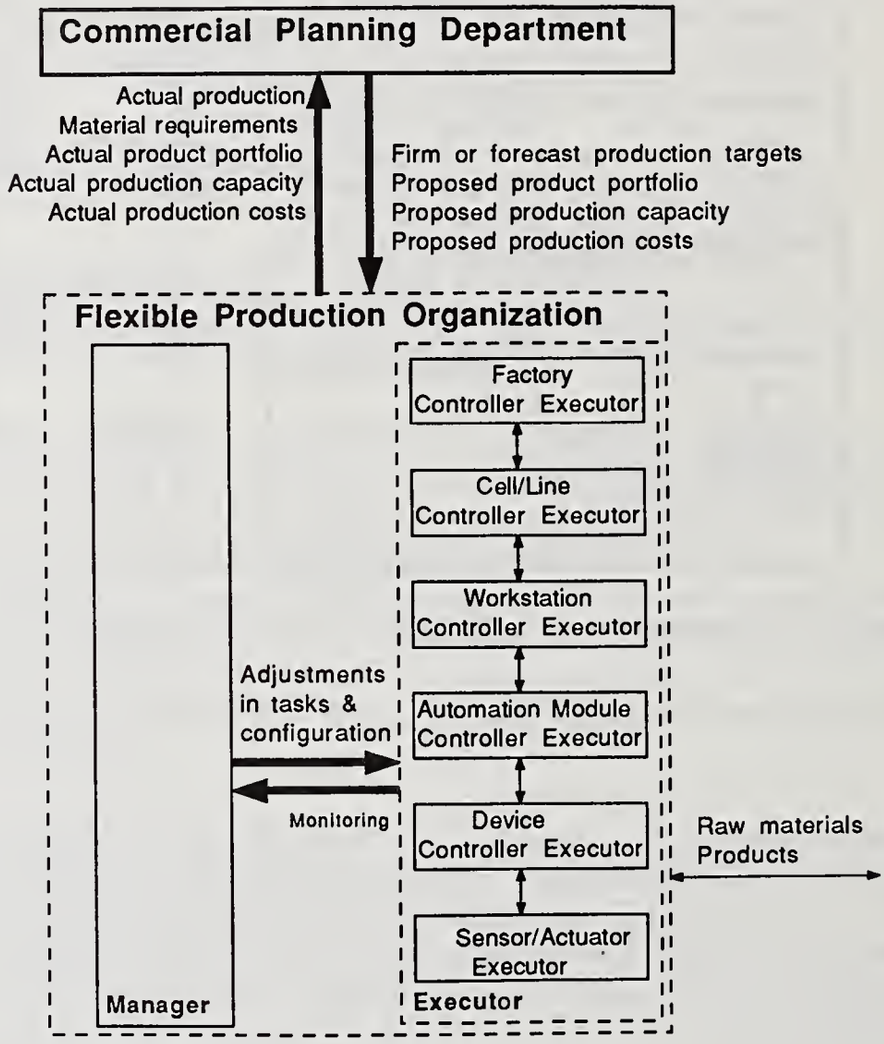


Figure 9: Flexible production organization existing of Manager and Executor

Adjustments in Configuration. A Manager can change an Executor's configuration by adding or removing controllers and by giving each controller a "configuration description", which tells it with which other controllers it may interact.

Adjustments in Tasks. A Manager can change the way an Executor controller executes its

task by providing it with "control procedures". A control procedure defines which steps a controller should realize in a given situation and which Executor components it may use as resources for the execution of those steps. Think of a scheduling procedure, telling a Cell/Line Controller how it should determine the sequence of operations for its Workstations.

Monitoring. We have discussed the Manager's adjustments in the Executor's configuration and tasks. To assess the effects of its adjustments on the Executor's ability to operate efficiently in a designated application, and to decide whether re-adjustments are needed, the Manager should also observe the Executor by "monitoring" it.

Figure 9 still portrays the Manager as a single, monolithic component. We can decompose this component further, using the design techniques explained in Section 4. Figure 10 shows an intermediate result of the decomposition. The Manager has been decomposed into four components, i.e. a:

- **Master Planner**, which determines whether production targets are feasible, or whether product portfolio, production capacity, or production costs can be modified by:
  - providing the Product&process Developer with product specifications and requests to develop process plans and Automation Modules so that these products can be made; and
  - providing the Executor Supervisor with specifications of the required Executor.
- **Product&process Developer**, which can develop process plans on the basis of product specifications and develop Automation Modules required to execute the selected process plans;
- **Executor Supervisor**, which can modify the Executor on the basis of the specifications from the Master Planner.
- **Monitor**, which provides information about the performance of Executor components.

This concludes the overview of the development of a Reference Model for a flexible production organization. We have demonstrated how the techniques introduced in Sections 4 can be used to develop, step by step, a Reference Model that meets the criteria discussed throughout this paper. The model describes relatively simple components of a production organization and meanwhile allows us to keep sight of the relations of these components and of the relations of the components and the organization as a whole.

7 Conclusion

A CIM architecture describes a design of a control system for a production organization. To allow an effective and successful implementation this design should not unnecessarily complicate the manufacturing control, should be specified unambiguously, at a high level of abstraction, and in generic terms. Typically, architectures fail to meet these requirements. This is probably caused by a lack of theory addressing the purpose of a CIM architecture, the criteria it should satisfy, and strategies to develop one.

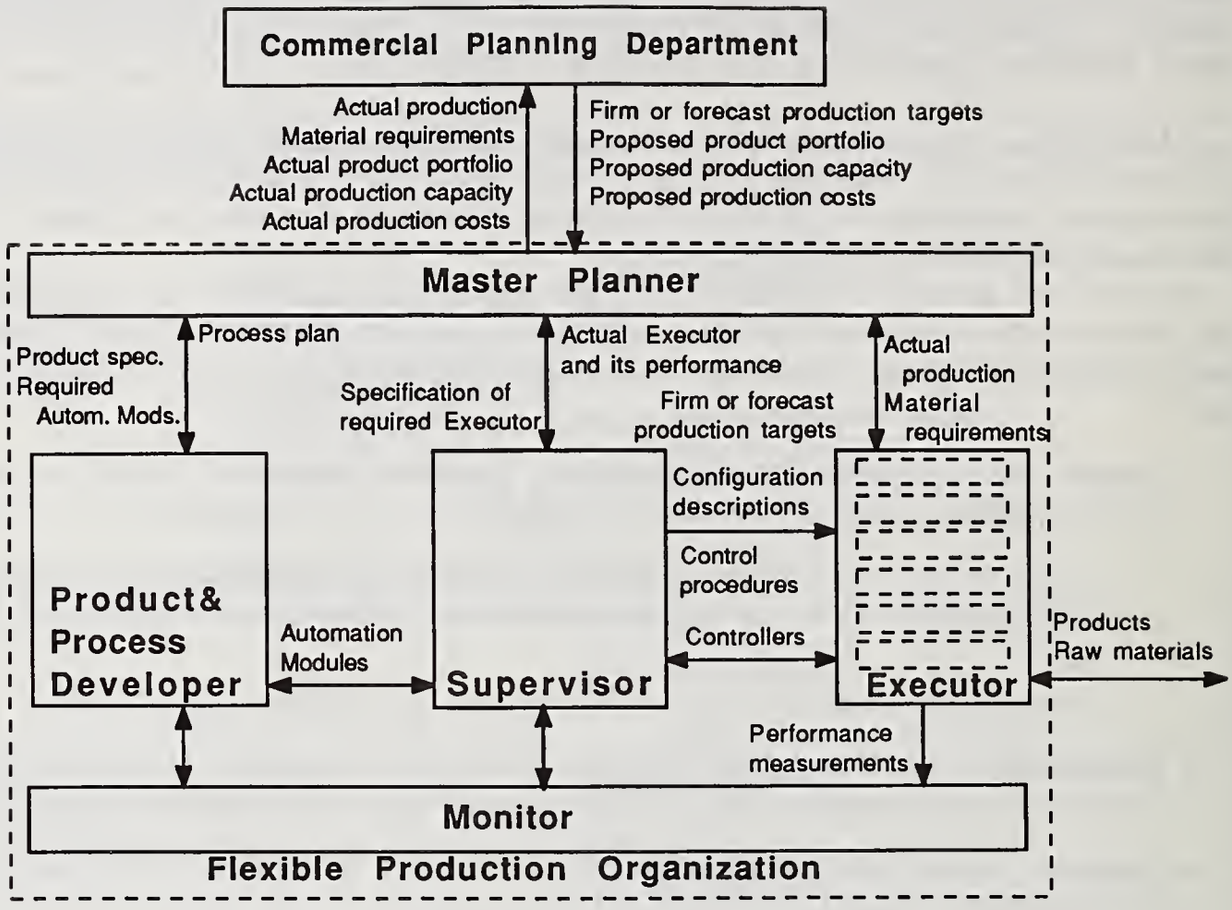


Figure 10: Components constituting a flexible production organization

We proposed that CIM architectures be developed to describe a production organization as a configuration of components in such a way that they allow us to understand how the components affect the behavior of the production organization as a whole. A CIM architecture should therefore have a "natural structure", i.e. a structure with the conceptual integrity that allows a human to master a production organization as a whole despite its inherent complexity.

We introduced an approach to develop a CIM architecture with a natural structure. This approach relies heavily on the orthogonality concept: allocate independent concerns to distinct components. We have demonstrated the approach by developing a CIM architecture for a flexible production organization, which in addition to responding to production targets, can honor requests to change its product portfolio, production capacity, and production costs.

The resulting architecture has found wide-spread application in a diversified, international, electronics company. The architecture is used as a company standard [BBH+89], copies of which have been requested by more than 2500 senior managers in product divisions. The architecture has also been used by "efficiency squads" as a template for the analysis of existing production organizations [ea89,CMvS+88]. Finally, the architecture has been successfully used as a high level design for the development of computerized control systems [WB90].

References

[AMBF83] J. Albus, C. McLean, A. Barbara, and M. Fitzgerald. Hierarchical Control for Robots in an Automated Factory. In *13th ISIR/Robots 7 Symposium*, 1983.

[Bak89] J. Bakker. *DFMS: Architecture and Implementation of a Distributed Control System for FMS.* Ph.D. Thesis, Delft University, 1989.

[BB82] G. Blaauw and F. Brooks. Computer Architecture, 1982. lecture notes, Twente University of Technology, The Netherlands, and University of North Carolina at Chapel Hill.

[BBH+89] J. Beukeboom, F. Biemans, C. Hehl, S. Sjoerdsma, and H. van Veen. CAM Reference Model–version 2.0. Technical report, Philips, CFT-Rep. 01/89, 1989.

[Bie89] F. Biemans. *A Reference Model for Manufacturing Planning and Control.* 1989. Ph.D. thesis Twente University of Technology, ISBN 90-9002961-3.

[Bri85] H. Brinksma. A Tuturial on LOTOS. In M. Diaz, editor, *Proc. of. the IFIP WG 6.1 5th. Int. Workshop on Protocol Specification, Testing and Verification.* North-Holland, 1985. also published as: Provisional LOTOS tutorial ISO/TC 97/SC 21 N.

[Bro75] F. Brooks. *The Mythical Man-Month.* Addsion-Wesley, Reading, Mass., New York, 1975.

[BS89] F. Biemans and S. Sjoerdsma. Functional Specifications for a Cell/Line Controller–to be published. Technical report, Philips Laboratories-Briarcliff, 1989.

[BV89] F. Biemans and C. Vissers. A Reference Model for Manufacturing Planning and Control Systems. *Journal of Manufacturing Systems, Vol 8, No 1*, 1989.

[CMvS+88] M. Curley, M. Moor, W. van Schaik, P. Trouwen, and A. Wiersma. CAM Architecture for BU Display Components. Technical report, Philips, proprietary information, 1988.

[Coh88] G. Cohen. Main Components of a Computer-Integrated Manufacturing Reference Model. *CIM Review*, pages 28–36, 1988.

[ea89] R. Beukeboom et al. CAM-SKE 90, CAM and Sector Control, PHILIPS, proprietary information, 1989.

[ISO83] ISO/TC97/SC16. Information Processing Systems, Open Systems Interconnection, Basic Reference Model, International Standard ISO/IS 7498. Technical report, ISO, 1983.

[ISO86] ISO/TC184/SC5/WG1. The Ottawa Report on Reference Models, version 0.1. Technical report, 1986.

[JM84] A. Jones and C. McClean. A Cell Control System for the AMRF. In *Proc. 1984 ASME International Computers in Engineering Conference, Las Vegas, Nevada, August, 1984*, 1984.

[NBS85] *Proc. on Factory Standards Model Conference.* National Bureau of Standards, Washington D.C., 1985.

[Par72] D. Parnas. On the Criteria To Be Used in Decomposing Systems into Modules. *CACM*, 15(12), 1972.

[PW78] H. Van Dyke Parunak and J. White. A Synthesis of Factory Reference Models. In *Proc. IEEE Workshop on Languages for Automation, Vienna, Austria, August 1987*, 1978.

[PW85] H. Pels and J. Wortman. Decomposition of Information Systems for Production Management. In *Proc. of IFIP W.G. 5.7 Working Conference on Decentralized Production Management Systems*, 1985. also published in 'Computers In Industry' Vol.6, Nr. 6, pp. 435-453.

[SME88] *Cell Controllers.* Society for Manufacturing Engineers and its Computer and Automated Systems Association, April 26-27, 1988.

[VL86] C. Vissers and L. Logrippo. The Importance of the Service Concept in the Design of Data Communication Protocols. In M. Diaz, editor, *Proc. of. the IFIP WG 6.1 Int. Workshop on Protocol Specification, Testing and Verification*, pages 3–17, 1986.

[WB90] R. Wendorf and F. Biemans. Structured Development of a Generic Workstation Controller. In *Proc. CIMCON '90, International Conference on CIM Architecture*, 1990.

STRUCTURED DEVELOPMENT OF A GENERIC WORKSTATION CONTROLLER

ROLI G. WENDORF FRANK P. M. BIEMANS

Abstract

Developing CIM controllers is a complicated design activity. It requires mastery of the formidable complexity of manufacturing environments as well as of the hardware and software components required to build the controllers. We discuss system engineering techniques that can be applied to structure this complexity into manageable design phases.

The starting point is a conceptual design of a factory wide CIM system, which describes, in manufacturing terminology, the tasks, information requirements, and performance requirements of a suite of CIM controllers. Formal, structured techniques are then available for translating the conceptual description of controllers to their detailed requirement specifications, design, and computer based implementation. We demonstrate the translation techniques by reviewing the development of a particular CIM component, a Workstation Controller, currently in industrial use.

1 Introduction

Many commercially available factory controllers for CIM provide *computer* solutions for industrial automation: general purpose computer systems which have been tailored for the factory environment with real-time operating systems, graphics support, and a large number of I/O ports [AB87,Soc88]. While such platforms are useful ingredients of a CIM system, they do not relieve the user of the burden of developing complex manufacturing control solutions. Developing such CIM applications is difficult, since it requires a rare combination of skills: mastery of the formidable complexity of manufacturing, as well as system engineering.

We participated in a company-wide program to develop a suite of CIM controllers. Unlike the *computer* solutions mentioned above, these controllers provide *manufacturing control* solutions: application software to control a transport system, control a warehouse, schedule operations, etc. The programming effort required by the user to tailor these controllers for specific factories is minimal. The controllers have other advantages as well. For example, user interactions with the controllers use manufacturing terminology rather than computer terminology. Furthermore, the CIM controllers are generic, and hence applicable to a wide variety of situations, including the handling of new products. Finally, the controllers can easily be integrated into a factory-wide CIM system responsible for a variety of control tasks such as machine control, machine maintenance, and scheduling, because they can exchange information according to specified protocols.

In order to develop such a suite of CIM controllers, we had to:

- determine which task a specific controller should execute;
- develop the protocols that govern the interactions of the controllers with one another;

- develop algorithms for controllers to provide adequate performance;
- develop techniques to implement the controllers.

We provide an overview of these activities below.

We described the tasks of the controllers in a 'Reference Model for Manufacturing Planning and Control' [Bie89,BV89]. This model describes an idealized production organization, which receives production targets, accepts raw materials, and dispatches products. More specifically, the model describes a production organization as a configuration of controllers, each with its own tasks, information requirements, and performance targets. The controllers are responsible for production planning, inventory management, material requirements planning, scheduling, transportation, machine control, servo control, sensing, product design, machine design, process planning, master planning, monitoring, configuration management, manufacturing engineering, and so on.

Based on the task description of a controller, given by the Reference Model, we develop its external specification. This external specification portrays a controller as a black box and defines its abstract interactions with its environment. Thus it specifies the protocols that describe which information the controllers exchange, when and under which conditions.

We express the external specifications in a formal language. The major motivation for the use of formal languages is their inherent precision. We can thus develop specifications that are interpreted unambiguously by different parties who are requested to implement the CIM controllers. We chose the language LOTOS [Bri86], which has been developed by the International Standardization Organization for the specification of computer protocols, but is equally suitable for the specification of CIM controller protocols [BB86].

We develop algorithms for the controllers which permit satisfactory performance in typical factory scenarios. Given the dominance of flow lines, for example, we develop algorithms that allow specific CIM controllers to schedule flow lines.

Given an external specification for a controller, we implement the controller in several steps. We first translate the external specification into an internal specification. The internal specification describes an abstract internal mechanism that shows how a controller can interact with its environment in accordance with the external specification. The internal specification, typically expressed in LOTOS as well, shows how a controller can be built and therefore supports the physical implementation of the controller. Subsequently, the internal specification is translated into programming language code in "C".

The various translations can be performed in a well-structured manner. LOTOS allows a smooth transition from external specifications to internal specifications, and reference [Wen88] describes a technique to translate LOTOS specification into "C" code.

In this paper, we discuss how the above development steps – determination of tasks, external specification, internal specification, implementation in "C" – are carried out. We use the development of a Workstation Controller, one of the CIM controllers, as an example to illustrate our structured development approach. The development reported here has led to a prototype Workstation Controller, which formed the basis for an industrial product[Sys89].

2 Workstation controllers

As mentioned before, the Reference Model [Bie89] analyzes which tasks should be executed to run a production organization and assigns these tasks to controllers.

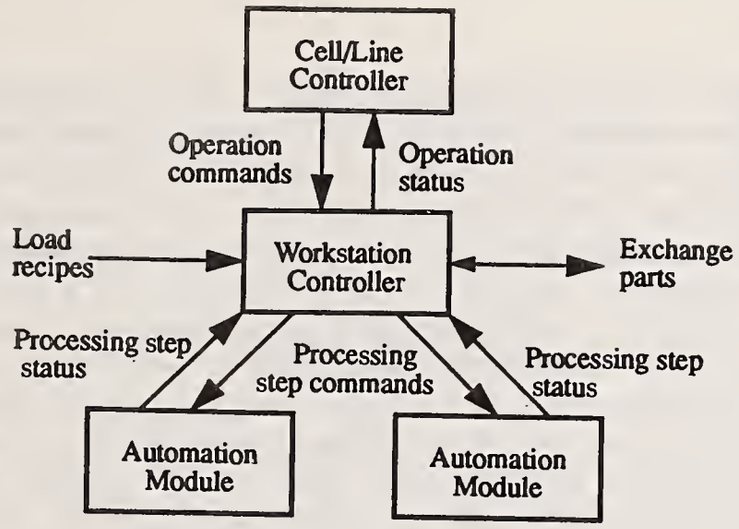


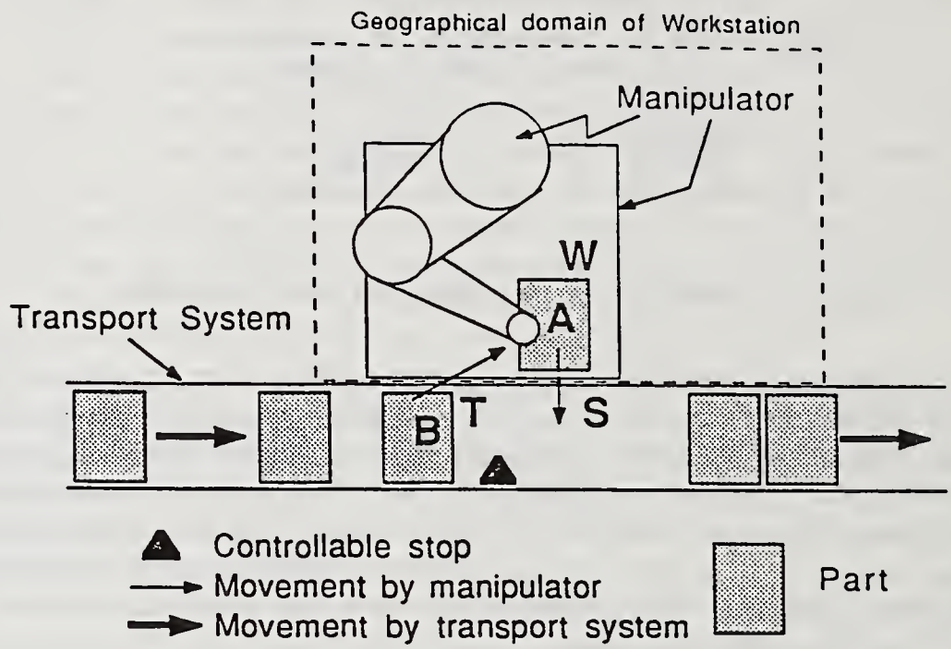
Figure 1: Interactions of a Workstation Controller

The model distinguishes between the scheduling and execution of operations. Scheduling is the selection of operations to be executed, the time of their execution, and the resource that has to execute them. Scheduling affects the completion time of products as well as the utilization of resources in all kinds of production organizations. In job shops, parts have many alternative routes of operations and resources; scheduling typically aims at maximizing the utilization of the resources. In flow shops, the routes are more constrained. One of the major scheduling concerns is the selection of the sequence in which parts enter the flow shop so as to minimize their lateness and earliness with respect to their due dates.

We define an "operation" as a material processing activity that can be scheduled in a useful manner. It is the smallest, indivisible entity for scheduling purposes. An operation consists of "processing steps" such as grinding, polishing, heating, or displacing materials. Unlike operations, processing steps are not scheduled; they are always executed in the same temporal order. For example, operation A is executed by displacing, heating, and quenching a material. The three processing steps, displacing, heating, and quenching, must always immediately follow each other; there is no need to schedule them separately.

A Workstation Controller (WSC) is a component of a production organization that can execute operations by coordinating Automation Modules, which execute processing steps. Figure 1 gives an abstract view of a WSC, illustrating how it interacts with its environment rather than its illustrating its physical implementation. It accepts commands from a Cell/Line Controller, which is responsible for the scheduling of operations. These commands instruct the WSC to execute certain operations, and accept parts required for the operations from other Workstations. The WSC sends commands to the Automation Modules so that they execute the required processing steps. It dispatches parts when an operation is completed, and reports to the scheduler (Cell/Line Controller) that the operation is completed.

As Figure 1 also shows, a Workstation Controller accepts "recipes". A receipe describes which processing steps have to be executed, by which Automation Module, and in which order to execute a certain operation. Receipes are typically provided by an engineering department in a production organization. They have to be modified or replaced when the Workstation Controller is required to execute new operations.



During the execution of an operation, the Workstation Controller commands the 'Manipulator' Automation Module to put object A at location W. The Manipulator is not needed for the next processing step to be performed on A and therefore becomes 'idle' after having put A on W.

Now Part B arrives. Suppose that the Workstation Controller has been commanded to execute an operation on B. As a first processing step for the execution of this operation, the Workstation Controller should command the Manipulator to pick up B and to put it at location W. However, it should refrain from giving the Manipulator this command despite the fact that it is idle because there is no room for B within the domain of the Workstation.

Trying to pick up B by the Manipulator would result in a dead lock: B cannot be put anywhere, and there is no free manipulator that can make room for B.

Figure 2: Resolving competing claims on Automation Modules

A Workstation Controller relieves a scheduler (Cell/Line Controller) from the task of coordinating the execution of processing steps. The Cell/Line Controller determines which operations should be executed, and leaves the execution to Workstation Controllers. A Workstation Controller therefore bridges the logistical portion of a production organization, responsible for scheduling, and the technological portion, responsible for the execution of processing steps. A Workstation Controller can also facilitate a quick introduction of new products. As described above, it can easily be instructed to execute a new operation by providing it with a new recipe.

Viewed in isolation, the task of a Workstation Controller, coordinating Automation Modules as prescribed by recipes, may look rather simple. However, complexities arise when a Workstation Controller is required to concurrently execute multiple operations with conflicting claims on Automation Modules, space, tools, etc. Figure 2 illustrates such a situation.

We have developed a Workstation Controller (WSC) [BS86] that can execute multiple operations, allows suspension or cancellation of operations, can exchange parts with other workstations along with information that identifies the parts, can manage a local buffer of tools or parts, and can collect test data. In the following sections, its development steps after task description will be reviewed.

3 External specifications

In the previous section, we defined the task of a Workstation Controller (WSC) as executing operations by coordinating Automation Modules so that they execute processing steps. We should now develop an external specification for a WSC, which prescibes how a WSC should interact with its environment to realize its tasks.

The task description suggests that a WSC be engaged in four basic interactions:

- Operation Commands, which define which operations the WSC should execute;
- Operation Status, which reports that a certain operation is completed;
- Processing Step Commands, which define which processing step an Automation Module should execute; and
- Processing Step Status, which report that a certain processing step is completed.

A basic external specification for a WSC completely defines these interactions and their temporal ordering. The temporal ordering defines, for example, whether the WSC may accept an Operation Command before it has finished operations requested by previous commands. Such a basic functional specification is relatively simple, and conceptually close to the the task definition of a WSC. Obviously, the functionality offered by a WSC according to these basic specifications would be limited. However, we can extend the basic specification to enhance the WSC's functionality. We could, for example, specify that it would accept commands to suspend previously ordered operations.

We express the external specifications in a formal language, with explicitly defined syntax and semantics. Formal languages are more precise than natural languages and are therefore more likely to produce specifications that are interpreted unambiguously. It is important that the formal language should be able to properly express the external characteristics of CIM systems. We chose

the language LOTOS. As we will illustrate, its generic concept of interaction, temporal ordering principles, mechanisms for process abstraction, and abstract data types allow us to produce external specifications that are conceptually close to the tasks and interactions of CIM systems that we described sofar [BB86].

We give a basic specification of a WSC in LOTOS, and illustrate how such a specification can be extended. We do not require our readers to be LOTOS experts; a brief introduction to LOTOS should suffice to understand the flavor of the given specifications.

3.1 Overview of LOTOS

LOTOS allows us to specify the externally observable behavior of a system, i.e. its interactions with its environment, and their temporal ordering. It specifies a system as a "process" that interacts with other processes. More specifically, it specifies the preparedness of a process to interact. It is assumed that interactions take place if two or more processes are prepared to interact. Compare this with the process of shaking hands. One person may want to shake hands, but hands will only be shaken if there is another person that is prepared to shake hands.

LOTOS processes interact to establish information values. Such a value could, for example, define the operation a WSC has to execute. It depends on the preparedness of the participating processes which values may be established. A "!" is used to indicate that a process wants to establish only one value; a "?" is used to indicate that a process can establish multiple values.

Suppose, for example, that one process is described by "! 3" and another by "? x : Natural_Numbers". The first process can establish "3", the second any natural number. In this case, the value "3" will be established when the interaction takes place since it is the only value acceptable to both processes.

Processes can interact if they share "gates". LOTOS notation "process A[a]" indicates that process A can interact at gate "a". LOTOS notation "A[c,d] |[c]| B[c,e]" indicates that process A interacts at gates c and d, and process B at gates c and e. Further, processes A and B have to synchronize their interactions at gate c. This means that a third process could interact with A at gate d, or with B at gate e. But, when the third process interacts at gate c, it has to interact with both A and B.

Compare this with the LOTOS notation: "**hide** c **in** (A[c,d] |[c]| B[c,e])". Now, gate c is not accessible for a third process. The third process can interact with A at gate d or with B at gate e. It cannot interact at gate c, but A and B can interact at gate c.

To define a process, say A, that can interact at gate b, accept any natural number, and the value 5, we write:

process A[a] : **exit** :=

a ? x : Natural_Numbers ! 5
; **exit**

endproc

"exit" indicates that the process terminates. ";" indicates sequentiality: in the above example, process A has to accept a natural number before it exits. There are operators to define other temporal orderings of interactions as well. "[]"indicates selection of interactions: one of a set of options will be chosen, depending on the preparedness of the environment to participate in the interactions. ">>" indicates sequentiality of processes. Finally, "|||", "|[]|", and "||" indicate

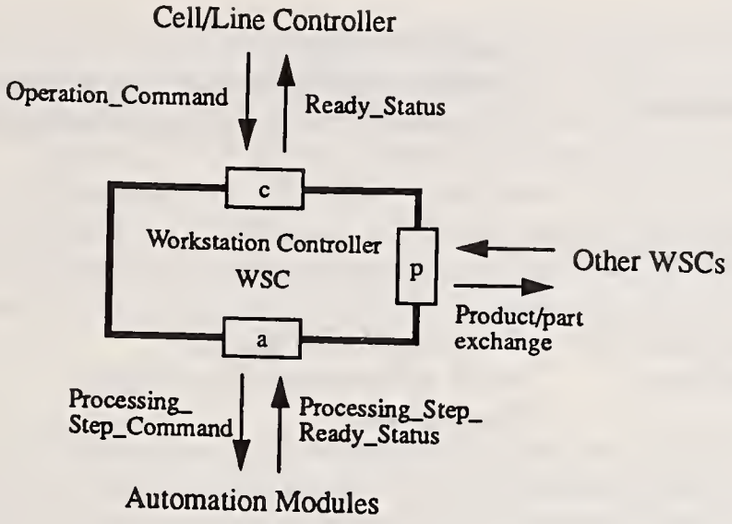


Figure 3: A workstation controller viewed as a LOTOS process

parallel behavior. "|||" indicates that processes operate in parallel without mutual interactions. "|[a]|" indicates that processes operate in parallel but synchronize on gate "a". "||" indicates that processes operate in parallel and synchronize on all their common gates.

3.2 Skeleton specifications

The process of developing the WSC specifications is shown by an example below. Figure 3 shows a WSC modelled as a process which interacts at gate c with a cell/line controller, the CIM controller that schedules operations and commands a WSC to execute operations. The WSC interacts with other workstation controllers at gate p, and automation modules at gate a. The cell/line controller can issue a *Operation_Command* to request the workstation controller to execute a certain operation. On receiving the command, the workstation controller accepts the parts required for the operation from other workstation controllers, and commands automation modules to execute processing steps. On completion, the WSC sends the processed part to another workstation controller, and returns a *Operation_Status* to the cell/line controller.

Using the specification style in [VSvS88], we divide the basic external specification for the WSC into four parts: three interface processes (*C_Interface, P_Interface, and A_Interface*) to interact at gates c, p, and a respectively, and the *Operation_Execution* process, which shows how the WSC interactions at the three gates are ordered. Part of the basic, skeleton specification is shown below. Comments are indicated with a "*".

```
process WSC[c,p,a] : noexit :=
                                  * WSC interacts at gates c, p, and a.

  (C_Interface[c]                 *defines interactions with Cell/Line Controller
  |||                             *while, in parallel
  P_Interface[p]                  *defines interactions with WSCs
  |||                             *while, in parallel
  A_Interface[a])                 *defines interactions with Automation Modules
```

```
  ||                                  *while, in parallel
  Operation_Execution[c,p,a]          *defines relations between interactions with Cell/Line
                                       Controller, WSCs, and Automation Modules.

where
process C_Interface[c] : noexit :=

  c ? opr : Operation_Command         *accept operation at gate c
  ;c ! Operation_Status(opr)          *next, offer status value at gate c
  ;C_Interface[c]                     *next, repeat.

endproc
endproc
```

Similar to the *C_Interface* process, the *A_Interface* process would define constraints on the exchange of processing step commands and status between the WSC and Automation Modules. The *P_Interface* process would define constraints on the acceptance and dispatch of parts by a WSC. The *Operation_Execution* process is:

```
process Operation_Execution[c,p,a] : noexit :=

(
  c ? opr : Operation_Command         *accept operation command
  ;p ! in_part(opr)                   *next, accept part referred to in operation
  ;Recipe_Execution[a](opr)           *next, execute recipe for the operation to process part
  >> p ! out_part(opr)                *next, dispatch part processed by operation
  ;c ! Operation_Status(opr)          *next, report ready status indication
  ;stop                               *next, this process dies
)
|||                                   *meanwhile, in parallel
  Operation_Execution[c,p,a]          *execute other operations.


where
process Recipe_Execution[a](opr : Operation_Command) : exit :=
  (* Process sending commands to Automation Modules and interpreting
  their status so that an operation gets executed *)
endproc
endproc
```

The example defines a bare skeleton for a workstation controller, which is limited in many respects. However, it provides a simple, comprehensible framework, conceptually close to the task definition given by the Reference Model, to which further functionality can be added step by step.

First reconsider the process *WSC[c,p,a].* According to the specification, the WSC can first accept an operation command. This is defined by the process *C_Interface* and *Operation_Execution. C_Interface* shows that the WSC will have to return an operation status after the receipt of an operation command. However, it has to synchronize this sending of the operation status with *Operation_Execution.* The process *Operation_Execution* ensures that the operation status will not be returned until the processed part has been dispatched by process *P_Interface.* The WSC is thus specified as a parallel composition of relatively simple processes. The specification of the WSC can be refined relatively easily by refining the specifications of some individual processes.

3.3 Refinements

Consider the following refinement. In the skeleton specification, the workstation controller can execute only one operation at a time. This is because *C_Interface* requires that an Operation_Command is followed by Operation_Status before another Operation_Command can be received. The parallel processing of Operation_Commands will be allowed if the process *C_Interface* is redefined as follows:

```
process C_Interface[c] : noexit :=
(
  c ? oper : Operation_Command          *accept operation command
  ;c ! Operation_Status (oper)          *next, return operation status.
  ;stop                                 *next, terminate process
)
|||                                     * Meanwhile, in parallel
  C_Interface[c]                        * behave as descibed above: accept command etc.
endproc
```

Note that according to the above specification, multiple *C_Interface* processes execute in parallel each capable of accepting an operation command. In effect, the WSC is now capable of accepting multiple operation commands in parallel.

Similarly, we can extend the specification "WSC" to:

- show how recipes are stored and processed when the workstation controller executes an operation;
- allow the workstation controller to accept multiple products for an operation, or to use some of the products it keeps in stock;
- exchange products with other workstation controllers using hand shake protocols, and physically move the products in or out by commanding Automation Modules.

4 Internal specifications

We have demonstrated how the external specification of a WSC can be developed. We should now consider how a WSC can be built that interacts with its environment according to the external specification. As a first step towards physical implementation, we develop an internal specification for a WSC. Such a specification describes a WSC internally as a configuration of interacting

modules. In this paper, we focus on the specification of a WSC "kernel", which incorporates all the application and configuration independent functionality. A WSC for industrial use would require extensive facilities for system configuration, entering application dependent information, and operator interactions, in addition to a kernel.

We show the development of internal specifications by carrying on with the example of the previous section. Concentrating on the process *Operation_Execution* defined in the previous section, which describes the kernel functionality, we propose to achieve this behavior internally through three modules called G, H, and M [AMBF82]. See also Figure 4.

Let us quickly review the functionality of these modules.

4.1 H module

The H Module accepts operation commands from the Cell/Line Controller, and commands Automation Modules to execute processing steps. It checks the commands received, and executes recipes. In executing a recipe, the H Module waits for part arrivals, arranges WSC set-up, coordinates automation modules, etc. It also provides the M Module with information to allow it to bookkeep the status of the workstation. For example, when it sends a command to an automation module (AM), it will inform the M Module that the AM is busy executing a certain processing step. Similarly, when it commands an AM to move a part, it reports to the M Module that the location of the part has changed. The H Module informs the the Cell/Line Controller when an operation is completed, or cannot be completed due to some failures. The H Module can concurrently execute multiple similar or different recipes. It also accepts suspend and cancel commands, which interrupt the normal execution of operations.

4.2 G module

The G Module accepts status messages from Automation Modules, which indicate how far various processing steps have progressed, and the results from these steps. The G Module can be programmed by users to provide special status information. Users supply "Status Recipes", which define which status message should be sent to the operator or the Cell/Line Controller when certain events occur.

4.3 M module

The M Module maintains a "World Model" which bookkeeps the processing steps Automation Modules are currently executing, and the locations of parts within the domain of the workstation.

The M Module ties the G and H modules together. The G Module supplies the status for bookkeeping by the M Module whereas the H Module executes recipes on the basis of that status. When the H Module, for example, has to dispatch a certain part, it can ask the M Module for the location of that part.

4.4 A formal internal specification

To show the process of developing internal specifications, we leave out processes *C_Interface*, *P_Interface*, and *A_Interface*, which describe interactions with the environment on individual gates.

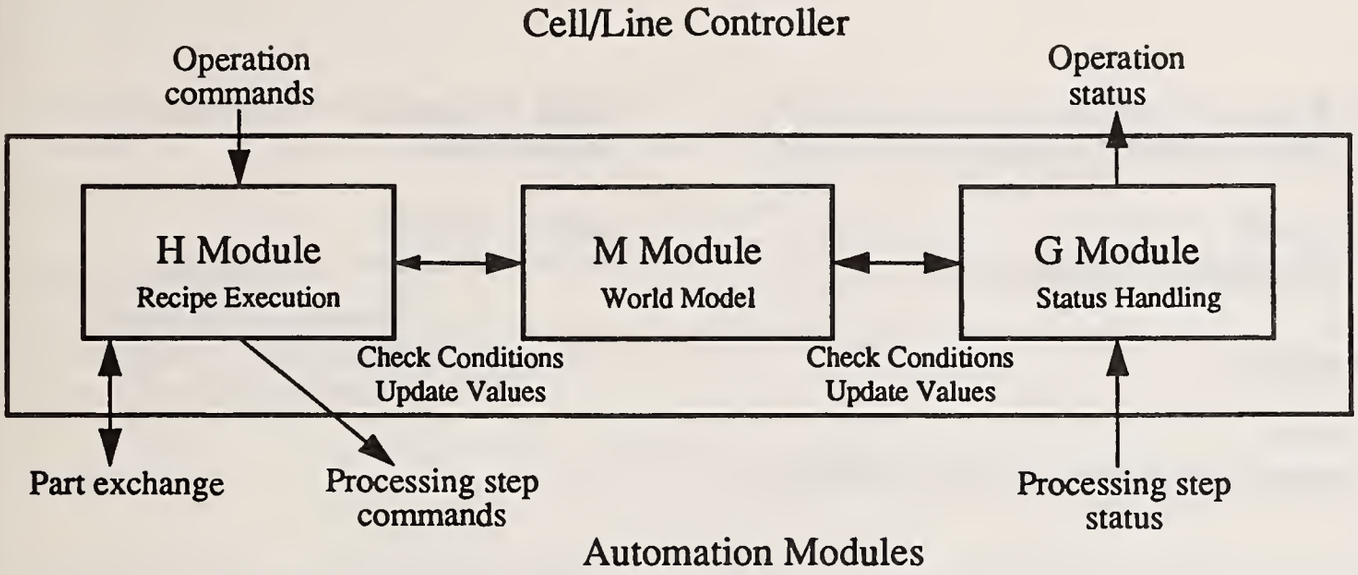


Figure 4: Internal organization of WSC

We replace *Operation_Execution* in the external specification by an process *Operation_Execution* that is defined as follows:

process Operation_Execution[c,p,a] : **noexit** :=

```
hide m in                          *gate m not accessible for Cell/Line controller etc.
(
  (G_Module[a,m]                   *G Module
  |||                              *and, concurrently operational
  H_Module[c,p,a,m])               *H Module
|[m]|                             *and, concurrently operational
  M_Module[m]                      *M Module
)
endproc
```

Upon investigation of the H Module, it appears that the module is engaged in three parallel activities: the acceptance of parts, the execution of recipes, and the dispatch of parts. We define these activities by three processes in a parallel LOTOS composition. *Accept_Parts* accepts commands from the Cell/Line Controller, the parts from the transport system, and signals the *Execute_Recipe* process. The *Execute_Recipe* process chooses the recipe for the operation, and executes it. The *Dispatch_Parts* process sends the processed parts out, and returns *Operation_Status* to the Cell/Line Controller.

process H_Module[c,p,a,m] : **noexit** :=

```
hide s1, s2 in                     *gates s1, s2 not visible outside H_Module

( Accept_Parts[c,p,m,s1]           *accept parts
```

```
  |||                                          *while, in parallel
  Dispatch_Parts[c,p,m,s2]                     *dispatch parts
)
|[c,s1,s2]|                                    *while, in parallel
  Execute_Operation[a,m,s1,s2]                 *execute operations

endproc

where
process Accept_Parts[c,p,m,s1] : noexit :=

(
  c ? opr: Operation_Command                   *accept operation command
  ;p ! in_part(opr)                            *accept part for operation
  ;m ! ws_status(at(in_location,in_part(opr)))  *update world model to bookkeep location of part
  ;s1 ! in_part(opr) ! in_location             *tell Execute_Operation to process part and
                                               *tell its location
  ;stop                                        *terminate process
)

||| Accept_Parts[c,p,m,s1]                     *while accepting parts

endproc

process Execute_Operation[a,m,s1,s2] : noexit :=

  c ? opr: Operation_Command                   *accept operation command
  ;s1 ! in_part(opr) ! location: Location      *accept signal to process part and its location

;([opr Is STAMP] →                             *if operation is STAMP, then
(
  ;a ! Processing_Step_MOVE(location, stamp_loc)  *command Automation Module to move
                                               *part to stamp_loc
  ;m ! request_Status ! MOVE_Completed         *wait till move is completed
                                               *according to status in World Model
  ;m ! ws_status(at(location,no_part))         *update World Model, no part at location
  ;m ! ws_status(at(stamp_loc,in_part(opr)))   *update World Model, part at stamp_loc
  ;a ! Processing_Step_STAMP                   *Command Automation Module to STAMP

  (* etc. Command AM to move part to exit point,
  and bookkeep part movements and AM usage
  *)
```

```
  ;s2 ! out_part(opr) ! exit_loc          *send signal to dispatch part, and give location
  ;exit                                  *terminate process
[]                                       *or

[opr Is ASSEMBLY] →                      *if operation is ASSEMBLY, execute its recipe
  (* Specification of recipe for
  execution of ASSEMBLY operation *)
) >> Execute_Operation[a,m,s1,s2]

endproc
```

Similarly, the internal specification for *Dispatch_Parts*, *G_Module*, and M_Module can be developed.

In the above specification, we did not specify how a WSC processes recipes. For the sake of brevity, we hard-coded the sequence of processing steps to be executed in the specifications. The recipes in [BB86] are similar to the hard-coded example, but supplied by the user. In later work [Blo89], the recipes have been simplified so that routine status setting and checking to bookkeep the location of parts and the status of Automation Modules is done automatically. Thus, to describe the STAMP operation, the user would specify the following recipe:

```
Operation STAMP(in_part,out_part)
begin
     AM_Robot DO move(in_part.location, stamp_loc)
     AM_Stamper DO stamp
     AM_Robot DO move(stamp_loc, exit_loc)
end
```

We have illustrated how LOTOS can be used to develop the internal specifications from the external specifications. The complete specification of the WSC is documented in [BB86]. It consists of approximately forty pages of LOTOS specifications. Equivalent informal specifications would be approximately 120 pages.

5 Implementation

The internal specifications of the previous section specify the internal structure of the WSC. To build a WSC, the internal specifications have to be translated into a computer program.

We developed a structured method for translating internal LOTOS specifications into an implementation in C++ [Str86]. The method can be used to implement controllers in a routine, mechanical way, reducing the time required for such implementations. We applied these methods to implement a complete WSC kernel [BB86]. Figure 5 shows some of the characteristics of the implemented kernel.

We will first review the translation technique, and then give an example of its use.

- **68000 hardware, single board computer implementation**
- **C++/C programming language**
- **Object code 50KB**
- **Response time to Automation Module: 0 to 20 ms**
- **Source code 70 KB**
  - **1200 lines application independent general-purpose part**
  - **200 lines include files for C++ compatibility, independent of WSC**
  - **350 lines configuration dependent (Interface Module) code**
  - **500 lines of application dependent code: hardcoding recipes, World Model, etc., usually entered from a user interface**
- **Direct correspondence between LOTOS specifications and implementation**

Figure 5: WSC implementation information

5.1 Translation technique

LOTOS is based on several language concepts that have no counterpart in common programming languages. We therefore had to provide the following informal rules to translate these concepts into "C/C++":

1. **Translation of LOTOS processes:** Each process in LOTOS is a unit of related activity. It is implemented by a C++ task or a procedure, which is a unit of related activity in a programming language.

2. **Translation of operators ||, |||, and |[]|:** We had to provide mechanisms for translating parallel LOTOS processes. The unit of parallelism in a programming language is a "task". Multiple tasks can execute concurrently on a single computer because tasks can be temporarily suspended, making computer power available for other tasks. This is not possible with procedures. To translate the LOTOS construct "G ||| H", two tasks G and H are set up. Whenever an interaction for G is required, task G is switched in, and when an interaction for H is required, task H is switched in. Note that a switching overhead is incurred in saving and restoring system state each time a task starts executing. Hence, care must be taken in deciding which LOTOS processes become tasks, and which become procedures.

3. **Translation of LOTOS interactions and gates:** A convenient method of interaction between tasks in a programming language is through message passing. As explained in Section 3.1, LOTOS specifications show the preparedness of a process to interact. In the translation, this preparedness is indicated by either putting a message in a shared memory area, or by waiting for a message to be put there by another task. The "shared memory area" between

two tasks is called a *queue*. It is allowed to hold at most one message, to force tasks to complete an interaction before starting a new one.

A gate in LOTOS is translated by message communication queues. However, a gate provides bidirectional communication, whereas a queue is unidirectional. Hence two queues have to be set up for each gate.

The interaction operators "!" and "?" are implemented by sending and receiving values in messages. The process executing a "!" usually sends a message using a *put* operator because it wishes to establish a single value. The process executing a "?" receives the value using a *get* operator, and then verifies whether the value received is in the acceptable range. If the value is not in the correct range, it is not accepted, and the interaction does not complete.

4. **Translation of temporal operators:** The sequentiality operator ";" requires no special translation, since different statements in a programming language are executed sequentially. The LOTOS choice operator "[]" shows a non-deterministic selection between alternatives. In most cases, this has been replaced by deterministic choice with the *if-then* or *case* statements in C.

5. **Translation of other operators:** A translation rule has been developed for each operator in LOTOS [Wen88].

6. **Translation of LOTOS objects:** Objects describe data structures in LOTOS. These are translated into *structures* in C. Rules for operating on objects are translated into subroutines which manipulate the structures.

5.2 Translation example

To illustrate the translation technique, consider the example WSC of the previous sections. In the internal specifications, the process *Operation_Execution* consists of a parallel composition of processes *G_Module*, *H_Module*, and *M_Module*, where both G and H interact with M, but not with each other. We could set up tasks for G, H, and M. Further examination shows that the M module only reads and writes data in its world model, and can be implemented by an object-oriented data structure with *read* and *write* routines. This prevents unnecessary task switching overhead when read and write operations are requested from M by G and H.

The H module (see Figure 6) consists of three parallel tasks: *Accept_Parts* and *Dispatch_Parts* which communicate with *Execute_Recipe* on gates *s1* and *s2* respectively. Two communication queues, *s1* and *s2*, are set up. For gates requiring bidirectional communication, such as *c* which interacts with the Cell/Line Controller, two queues *c_in* and *c_out* are set up, to receive operation commands and send operation status respectively. Note that there are no queues corresponding to *m*, because the M module is implemented as a shared data structure, and not as a task. The C++ code for task *Accept_Parts* is shown:

```
Accept_Parts::Accept_Parts (qhead* c_in, qhead* p_in, qtail* s1) {
                                        *program Accept_Parts, task Accept_Parts
                                        *c_in, p_in: queues for receiving messages
                                        *s1: queue for sending message
Operation_Command opr;                  *opr holds an operation command
```

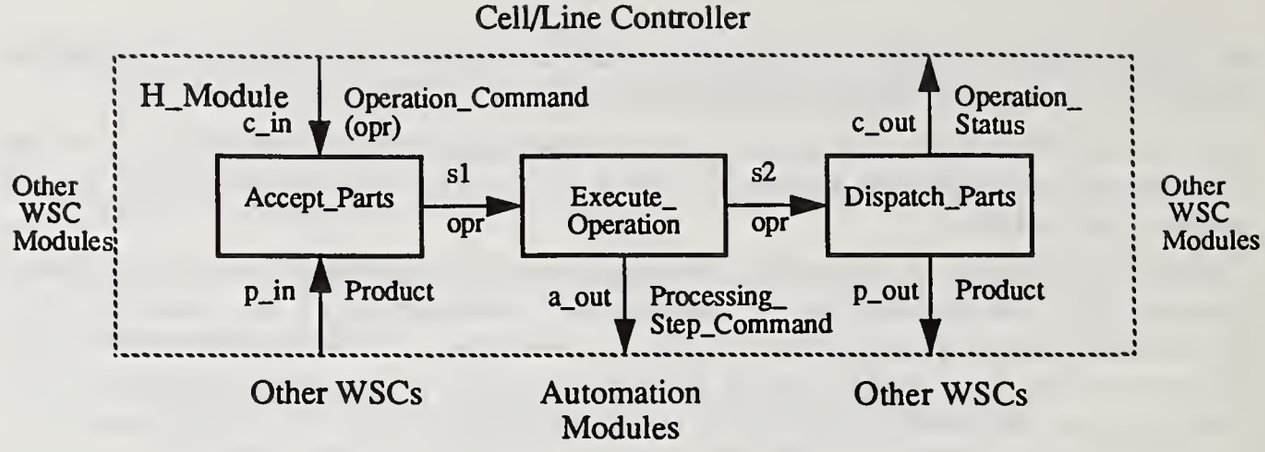


Figure 6: Tasks and queues in H_Module

```
for (;;) {                                          *carry out task for ever
    opr = c_in → get();                             *receive operation command
    in_product = p_in → get();                      *receive product
    m_module.write (in_location, in_product);       *update world model
    s1 → put (opr);                                 *send command
}}
```

6 Application

We developed a prototype WSC using the techniques described in this paper. It was used experimentally to solve an inductrial part calibration problem.

6.1 Tape head adjustment

An automated solution was required by a factory for the alignment of tape heads for magnetic tape drives, to reduce the cycle time for the alignment/calibration operation. We used this problem as a test case in our laboratory for the prototype version of the WSC. The WSC was used to control and co-ordinate a Robot, an Adjustment unit, and a Transport System [BHH+88]. A pseudo Cell/Line Controller provided commands to the WSC. The tape head adjustment set-up is shown in Figure 7. Here, the WSC is used to control the Alignment/Calibration station. In addition, there are three manual stations. The Transport System has separate control, and it moves parts between the various stations. The Cell/Line Controller provides transport commands to the Transport System, and operation commands to the WSC.

The steps in the tape head alignment operation were as follows: on receipt of a command from the Cell/Line Controller, the robot was commanded to pick up fingers (set-up) in preparation for part (tape drive) arrival on the transport system. When the part arrived, the WSC commanded the robot to pick it and place it in the workspace of the Adjustment Automation Module. The robot was then commanded by the WSC to change its fingers, pick up a measurement probe from storage,

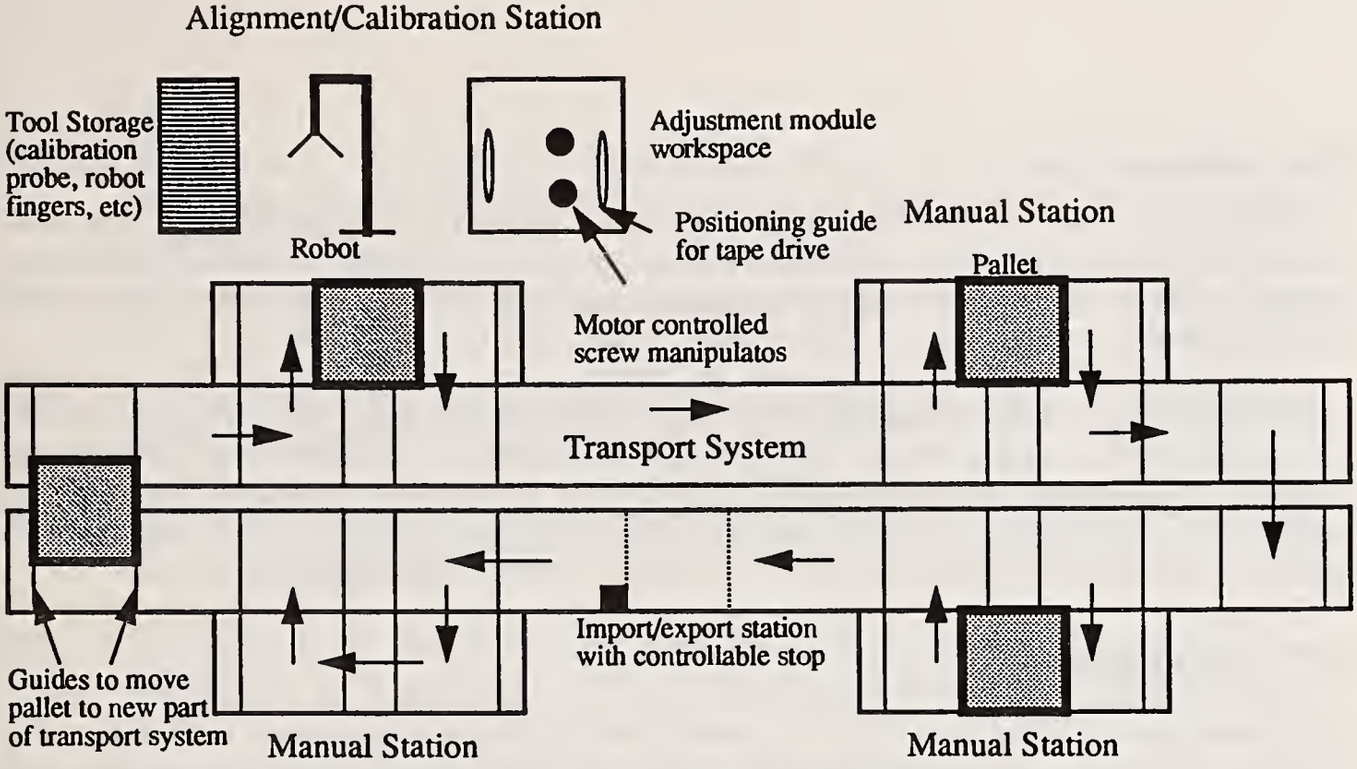


Figure 7: Workstation Controller demonstration system

and place it on the arm of the tape drive. The Adjustment unit was then commanded to measure the orientation of the tape drive, and based on the measurement obtained, to control a motor to change screw positions which would alter the tape head orientation. The adjustment process was repeated till the orientation was within an acceptable error margin, or a timeout occurred. When the adjustment was over, the robot was commanded to move the probe back, change its fingers, pick, move and place the tape drive on the transport system, and move to its home position. The transport system was signalled to move (exchange parts). The complexity of this example was considered representative of many electronic and light mechanical assembly applications.

The WSC was given recipes for three different commands: *Tape_Calibration*, *Part_Storage*, and *Part_Dispatch*. The *Part_Storage* recipe would receive a part from the Transport System, and have it moved by the robot to an internal storage location in the workstation's workspace. The *Part_Dispatch* recipe would command the robot to move a stored part from its internal storage location to the Transport System. The robot and the transport system were required to carry out each of these operations. The WSC was able to share these resources without interference, and would even carry out processing steps in parallel from different recipes. For example, the *Part_Dispatch* recipe commanded the transport system to leave with a part at the same time that the *Tape_Calibration* recipe commanded the robot to perform set-up steps.

6.2 Routing Controller

The WSC can also be used as a controller for a transport system. The WSC receives operation commands that request to transport a certain part from location A to B. The WSC executes a recipe that tells whether the part has to make a left turn, a right turn, or should go straight at nodes in the transport system. The WSC then commands Automation Modules that control the mechanics of the transport system at the nodes to move left, right, etc.

6.3 Industrial use

A WSC product has been developed on the basis of our prototype WSC. In addition to a WSC kernel, the product version has a substantial operator interface, and drivers for various Automation Modules. The operator interface includes a programming environment for recipes, to support the edit-compile-debug cycle.

The WSC has been in use in factories for printed circuit board assembly. It co-ordinates various component insertion machines, soldering stations, test and repair loops, etc. with recipes. It manages product change-over by downloading new programs to machines, and executing new recipes. It co-ordinates the transportation of products in the test and repair loop, and sends the product data from the test station to the repair station. The workstation controller has also been used in controlling the storage and retrieval of parts in a separate application.

7 Conclusion

The development of CIM controllers will benefit from the use of well-known system engineering techniques. However, before the actual development can start, one has to determine which manufacturing tasks a controller should execute and how it should interact with other controllers. We have discussed an approach that allows the definition of manufacturing tasks for a controller, its interactions with its environment, and that, via several steps, leads to a physical implementation of the controller.

We demonstrated the approach by reviewing the development of a Workstation Controler. This Workstation Controller coordinates Automation Modules for the execution of operations, and allows an easy reprogramming of the operations via recipes. The development has profited from feedback through experiments, and has led to a controller that is in actual use.

Acknowledgements

The Workstation Controller specifications were developed in co-operation with Sjoerd Sjoerdsma. Several colleagues at Philips Laboratories - Briarcliff participated in the tape alignment demonstration. The authors would like to thank Omer Bakkalbasi, Pieter Blonk, and Fletch Holmquist for reviewing this document, and providing feedback.

References

[AB87] Allen-Bradley. Vista 2000 cell controller. Product Brochure, 1987.

[AMBF82] J.S. Albus, C.R. McLean, A.J. Barbera, and M.L. Fitzgerald. An architecture for real-time sensory-interactive control of robots in a manufacturing facility. In *Fourth IFAC/IFIP Symposium on Information Control Problems in Manufacturing Technology*, 1982.

[BB86] F. Biemans and P. Blonk. On the formal specification and verification of CIM architectures using LOTOS. *Computers in Industry*, 7:491–504, 1986.

[BHH+88] F.P. Biemans, T.J. Harosia, F.N. Holmquist, C-C. Lee, A. van de Stadt, and R.G. Wendorf. Eaasy-0 system specification. Technical Report TR-88-048, Philips Laboratories - Briarcliff, 1988.

[Bie89] F. Biemans. *A Reference Model for Manufacturing Planning and Control.* PhD thesis, University of Twente, The Netherlands, 1989. ISBN 90-900 2961-3.

[Blo89] P. Blonk. A user friendly recipe language for workstation controllers. Technical Report TN-89-089, Philips Laboratories - Briarcliff, 1989.

[Bri86] E. Brinksma. Lotos: A formal description technique based on the temporal ordering of observational behavior. Technical Report ISO/TC97/SC21 N, International Organization for Standardization, 1986.

[BS86] F. Biemans and S. Sjoerdsma. Description and motivation of a workstation controller architecture. Technical Report 55/86 EN, Philips Centre for Manufacturing Technology, 1986.

[BV89] F. Biemans and C. Vissers. Reference model for manufacturing planning and control systems. *Journal of Manufacturing Systems*, 8(1), 1989.

[Soc88] Society of Manufacturing Engineers. *Achieving Integration Through Cell Controllers*, April 1988.

[Str86] B. Stroustrup. *The C++ Programming Language.* Addison-Wesley, 1986.

[Sys89] Philips I&E Industrial Automation Systems. Flexible automation systems tool FAST. Product Brochure, 1989.

[VSvS88] C. Vissers, G. Scollo, and M. van Sinderen. Architecture and specification style in formal descriptions of distributed systems. In *Proc. of. the 8th. IFIP WG 6.1 Int. Workshop on Protocol Specification, Testing and Verification.* North-Holland, 1988.

[Wen88] R.G. Wendorf. Implementation of manufacturing planning and control systems from lotos specifications. Technical Report TN-88-015, Philips Laboratories - Briarcliff, 1988.

ARCHITECTURE OF A FACILITY LEVEL CIM SYSTEM

G. HARHALAKIS, M.E. SSEMAKULA, AND A. JOHRI

Abstract

Computer Integrated Manufacturing (CIM) is commonly understood to be the integration of CAD and CAM. Instead, we propose that CIM can be regarded as a systems approach of linking together the various automation tools available today, so as to enable control of the entire manufacturing operation as well as related business functions. Unlike most of the recent research in CIM, we are not simply considering integration of various shopfloor-level elements of hardware and software, but rather address integration of the high-level manufacturing functions. The integration centers around select common entities in the system. A model for this integration at the facility level is presented along with the rules of interaction between the constituent modules. The model is based on the principle of database inter-operability and has been implemented using the 'Update Dependencies' language.

1 Introduction

The highly competitive nature of today's industry has forced manufacturers to actively explore the potential applications of computers in the manufacturing environment. The driving goal behind this search is the need to improve quality, and increase productivity, while maintaining a favourable cost structure. This has become increasingly more difficult with the general shortening in product life cycles, increased complexity of individual manufacturing processes, and a severe shortage of skilled labor.

In response to the pressures mentioned above, computers have been applied to a variety of functions within manufacturing, resulting in a proliferation of *'Computer Aided'* systems such as Computer Aided Design (CAD), Computer Aided Process Planning (CAPP), and Computer Aided Manufacturing (CAM), among others. In developing each of these individual systems however, little attention was given to how these systems would function in synergy. Consequently, most systems were incompatible with one another which resulted in the so called islands of automation. Without doubt, each of these computer aided systems helps to improve the efficiency of the function to which it is applied. It is now generally accepted however, that if the various application systems were interfaced, the performance of the resulting integrated system should be a dramatic improvement on

the capabilities of the individual systems. This has motivated the on-going search for a truly integrated computerized manufacturing system.

In this paper, we describe an architecture for a Computer Integrated Manufacturing (CIM) system based on the data commonality between selected modules of a manufacturing facility. The modules addressed in this research are Computer Aided Design (CAD), Computer Aided Process Planning (CAPP), and Manufacturing Resource Planning (MRP II). The data commonality is related to specific key entities such as parts, product structures, routings, and workcenters. By analysing the detail data content of each entity, rules are developed that govern the functioning of the overall system and interaction between modules, to ensure consistency and integrity between any data common to various modules.

Since each module in the system is assumed to have its own database of the entities it uses, it is important to maintain consistency between the various databases. The integration methodology adopted here, eliminates the need for data duplication, and controls consistency, by using the principle of database inter-operability. The inter-operability system monitors the individual modules integrated, and upon occurrence of an event affecting the state of any system database, executes a response according to preprogrammed expert rules, involving operations on the related systems' databases, to maintain data consistency between the modules involved. This required the development of a detailed rule base. The rule base was established on the basis of projected events found to be necessary following an action on a key entity in any module. The number of different actions possible during various database states, on different entities, led to a very large rulebase in order to specify the system completely, and to avoid inconsistencies and logical errors.

2 System Architecture

Integration of manufacturing functions can be achieved at various levels within the hierarchy of a manufacturing enterprise. Most contemporary research in CIM has concentrated on integration of the shopfloor level functions, resulting in applications such as CNC, AGVs, and automated inspection. We are proposing an approach to integration that focuses on the facility level of the enterprise and concentrates on the integration of information rather than hardware. We have developed the CIM architecture necessary for the integration of CAD, CAPP, and MRP II.

The integration is achieved by identifying the key entities on which each of these modules operates, and taking advantage of the commonality of data required for each entity in the various modules. By establishing an appropriate rule base incorporated into a Distributed Database Management System (DDBMS), we can control the flow of data between the various modules of the system.

The DDBMS approach was selected over that of a single large database accessed by the various modules because of the ease of implementing the former in an existing manufacturing environment. Use of a single database would require a 'ground-up' approach,

whereby the databases of existing systems would have to be rebuilt in the process of creating a single database. In addition, the design of a single database requires consideration of all the data it will eventually contain, making subsequent additions or modifications difficult. With the DDBMS approach, integration can be modular and phased, making it more acceptable for introduction in an existing manufacturing facility.

2.1 Functional Design

The model developed here is intended for a discrete-parts make-to-stock environment where CAD, CAPP, and MRP II modules are best utilized. Each of these modules is assigned specific functions and the overall integrated system is developed on the basis of these functions which are detailed below.

CAD, being the center of design activity, is the primary controller of product design information. The evaluation of design alternatives, creation of new product parts, and the modification of existing parts is performed within CAD, taking into account inputs from other departments [1]. Manufacturing and marketing are two of the major contributors to information regarding product designs. In addition, CAD initiates the Bills of Materials for any product assemblies. An important problem commonly encountered is that as a function, manufacturing succeeds design. Therefore any manufacturing problems encountered due to part design are relayed to CAD only after designs have been finalized. It is therefore necessary for CAPP, the originator of process plans in the system, to work in collaboration with CAD, during the design of the part, in order to reduce this problem significantly.

CAPP is responsible for developing the process plans in the system. It organizes the manufacturing activities to be performed on a part into specific operations, each assigned to a particular workcenter, and requiring tools, jigs, and fixtures; and also determines processing times. CAPP can also initiate parts in the system that are used on the shopfloor such as tooling, jigs, or fixtures (all generically referred to as tools in this paper), as well as their associated bills of materials. CAPP also maintains detailed workcenter files for use in generating process plans.

MRP II plays an executive role, planning for and monitoring the actual procurement and processing of items [2]. It can initiate non- product parts such as supply items, into the system. In addition, it records the process plans as generated by CAPP, and also the product structures of assembly parts. Workcenter data are maintained here, with MRP II having sole discretion as to their initiation and deletion in the system.

2.2 System Design

The system is designed to operate on the basis of entity data manipulation. The entities involved are:

- parts

- bills of materials
- process plans
- work centers

Parts are further subdivided into:
Product Parts – which are parts that go into constituting a final saleable product
Tool parts – which are parts used within the manufacturing enterprise such as tooling, jigs or fixtures
Supply parts – which are other consumable items (usually not requiring drawings), that do not go into the final product such as office supplies.
Table 1 shows the various entities, the system module that initiates each entity, and the modules where each entity is maintained.

Entity	Initiating Module	Maintaining Module
Product part	CAD	CAD, MRP II
Tool part	CAPP	CAD, MRP II
Supply part	MRP II	CAD, MRP II
Bill of Materials	CAD, CAPP	CAD,CAPP, MRP II
Process plan	CAPP	CAPP, MRP II
Workcenter	MRP II	CAPP, MRP II

Table 1: **System Entities**

2.2.1 Entity Data

Part Data: The data for part entities are maintained by CAD as well as MRP II. Part records are stored during the design process in CAD, and are maintained during the life cycle of the part. This is required for general part reference, and situations involving engineering changes to the part in question. In some cases, records are maintained even after the part is no longer active in the system, for future design reference. MRP II stores part data in the form of Part Master Records, which form part of the Part Master Record/Bill of Materials module.

If design modifications to a part become necessary, engineering change procedures are initiated. Based on the extent of engineering change, a decision has to be made whether to create a new part, or simply a new revision of the old part. If only a new revision is required, then the old part number remains active, and only the revision is updated. Because of this, CAD and MRP II systems typically store part data under two different records, one set containing data common to all revisions (called the part record), and the other set containing data specific to each revision of the part (called the revision record).

Table 2 shows the part data, whereas table 3 shows the revision data supported by the majority of CAD and MRP II systems.

Field No.	MRP II	CAD
1	Part Number	Part Number
2	Drawing Number	Drawing Number
3	Drawing Size	Drawing Size
4	Description	Description
5	BOM Unit of Measure	BOM Unit of Measure
6	Purchase/Inventory Unit of Measure	—
7	Unit of Measure Conversion factor	—
8	Source Code	Source Code
9	Standard Cost	—
10	Lead Time	—
11	Supersedes Part Number	Supersedes Part Number
12	Superseded by Part Number	Superseded by Part Number
13	Source	Source

Table 2: **Part Data Maintained by MRP II and CAD**

Field No.	MRP II	CAD
1	Part Number	Part Number
2	Revision Level	Revision Level
3	Effectivity Start Date	Effectivity Start Date
4	Effectivity End Date	Effectivity End Date
5	Status Code	Status Code
6	—	Drawing File Name

Table 3: **Revision Data Maintained by MRP II and CAD**

Product Structures: The proposed model incorporates the facility of maintaining single-level product structures (commonly called bill of materials, BOM). These provide information regarding components that make up a given part assembly. A complex product structure can have several levels, with each level representing parts that can be used as components of parts at the next higher level. In this way, a multilevel product structure can be created for a part, with the lowest levels representing purchased parts.

Bills of materials for individual parts are initiated by the module that created the part concerned. For example, BOMs for product parts are initiated in CAD, while BOMs

for tool parts are initiated in CAPP. The parts initiated in MRP II generally do not require BOMs and consequently, this module does not have the facility to establish BOMs in the system. MRP II is still allowed to maintain product structure data, since it is responsible for printing the shop packet and issuing it to the shopfloor. The handling of product structure data is similar to that of part master data. Details of the product structure data maintained in the system are shown in table 4.

Field No.	MRP II	CAD	CAPP
1	Parent Part Number	Parent Part Number	Parent Part Number
2	Parent Revision Level	Parent Revision Level	Parent Revision Level
3	Item Number	Item Number	Item Number
4	Component Part Number	Component Part Number	Component Part Number
5	Quantity per Assembly	Quantity per Assembly	Quantity per Assembly

Table 4: **Product Structure Data in MRP II, CAD and CAPP**

Process Planning Data: Process planning is the function within manufacturing responsible for the conversion of design specifications into actual manufacturing instructions for use on the shopfloor. The resulting process planning data typically include details such as operation description, required tools, jigs and fixtures, workcenter, and operation parameters such as speed and feed. Our model incorporates the facility of maintaining these instructions in the form of a routing or process plan, related to a given part revision. The detailed data content of routings common to generic CAPP and MRP II systems are shown in table 5. It is noted that process planning data is maintained in the CAPP and MRP II modules. It is possible to have alternate process plans for any part revision, as well as different revisions of a particular routing.

Workcenter Data: Workcenters are primarily initiated into the system from MRP II. MRP II users are responsible for establishing as well as phasing out workcenters in the system. In addition, CAPP also requires detailed workcenter information for generating process plans, therefore its workcenter files are integrated with the workcenter data in the MRP II module. The basic workcenter data fields supported by the majority of MRP II and CAPP systems are shown in table 6.

3 System Description

The integrated operation of the system depends on the interactions that take place between the various modules in accordance with certain events or scenarios of events that can occur. Within the proposed integrated system, these interactions are controlled by the

Field No.	MRP II	CAPP
1	Part Number	Part Number
2	Part Revision Level	Part Revision Level
3	Routing Alternative Number	Routing Alternative Number
4	Routing Revision Level	Routing Revision Level
5	Operation Number	Operation Number
6	Operation Description	Operation Description
7	Workcenter ID	Workcenter ID
8	Tool Number	Tool Number
9	Jig Number	Jig Number
10	Fixture Number	Fixture Number
11	Set Up Time	Set Up Time
12	—	Batch Machining Time
13	—	Handling Time
14	Run Time	Run Time
15	Resource Code	—
16	Begin Date	—
17	End Date	—
18	—	Feed
19	—	Speed
20	—	Depth of Cut
21	—	Number of Passes

Table 5: **Process Planning Data Maintained in MRP II and CAPP**

Field No.	MRP II	CAPP
1	Workcenter ID Number	Workcenter ID Number
2	Workcenter Description	Workcenter Description
3	Department	Department
4	Capacity	—
5	Resource Code	—
6	Rate Code	—
7	Dispatch Horizon	—
8	Effectivity Start Date	Effectivity Start Date
9	Effectivity End Date	Effectivity End Date
10	Workcenter Status Code	Workcenter Status Code
11	—	Horse Power
12	—	Speed Range
13	—	Feed Range
14	—	Work Envelope
15	—	Accuracy
16	—	Tool Change Time
17	—	Feed Change Time
18	—	Speed Change Time
19	—	Table Rotation Time
20	—	Tool Adjustment Time
21	—	Rapid Traverse Rate

Table 6: **Workcenter Data in MRP II, CAD and CAPP**

use of *status codes*, which are central to the operation of the system. Each entity in each module has a status code associated with it. Note that the status code associated with a given entity can differ in different modules. The following are the status codes used in the system.

Working: This status code is used with an entity during the preparation of the requisite data for the entity. It signifies that the information for the entity is as yet incomplete and therefore the entity data cannot be used. For example, a working status is associated with a design that has not been finalized.

Released: This status enables the entity to become active in the system. It is given to an entity when all its required data have been completed and verified.

Hold: This status is given to an entity when the entity is being reviewed for possible revision or replacement. Depending on the particular entity, this status might be necessitated by unsatisfactory performance, breakdown, or even routine review.

Obsolete: This code is given to entities that are no longer required in the system. Before any entity can be deleted from a module database, it must have an 'obsolete' status code. The MRP II module is unique in not using this status code because it processes obsolescence by use of 'effectivity start' and 'effectivity end' dates.

The actions that can be performed on each entity depend on the value of the status code. The following types of actions are generally possible:

- Entity creation
- Entity modification or revision
- Entity obsolescence
- Entity deletion

The following scenarios have been identified as descriptive of the operation of the integrated system proposed here.

1. Creation of new manufactured parts in CAD
2. Creation of new purchased parts in CAD
3. New revisions of manufactured parts in CAD
4. New revisions of purchased parts in CAD
5. Creation of new manufactured parts in CAPP
6. Creation of new purchased parts in CAD
7. New revisions of manufactured parts in CAPP
8. New revisions of purchased parts in CAPP
9. Creation of new supply parts in MRP II
10. New revisions of supply parts in MRP II

11. Making parts obsolete
12. Deleting parts
13. Creation of new process plans
14. New revisions of process plans
15. Deletion of process plans
16. Creation of new workcenters
17. Deletion of workcenters
18. Addition of new component relationships in CAD
19. Addition of new component relationships in CAPP
20. Deletion of component relationships
21. Substitution of components in relationships
22. Copying relationships between assemblies
23. Changing the required quantity of a component

Below, a couple of these scenarios will be described in detail to illustrate the performance of the system.

3.1 Creation of New Manufactured Parts in CAPP

As explained in section 2.1, CAPP can create tooling type parts in the system. CAPP first establishes a record of the part under a temporary part number (state 1 in figure 1). This is because CAD is the sole center for assigning permanent part numbers, so as to maintain consistency. The data content of the temporary record is as follows:

1. Temporary Part Number
2. Description
3. Unit of Measure
4. Source Code

A relation exists within CAD enabling it to know which new parts entered into the system require permanent part numbers. The creation of the temporary record in CAPP results in the update of the temporary part number in the CAD relation. On CAD establishing the new permanent part number (state 2 in figure 1), the following events occur.

- A temporary tuple record is created in CAPP, containing the permanent part number established by CAD, and the temporary part number initiated by CAPP. This enables CAPP to refer to the part with its permanent part number .
- The part record, under the temporary part number, is deleted from CAPP, as CAPP does not support part records.

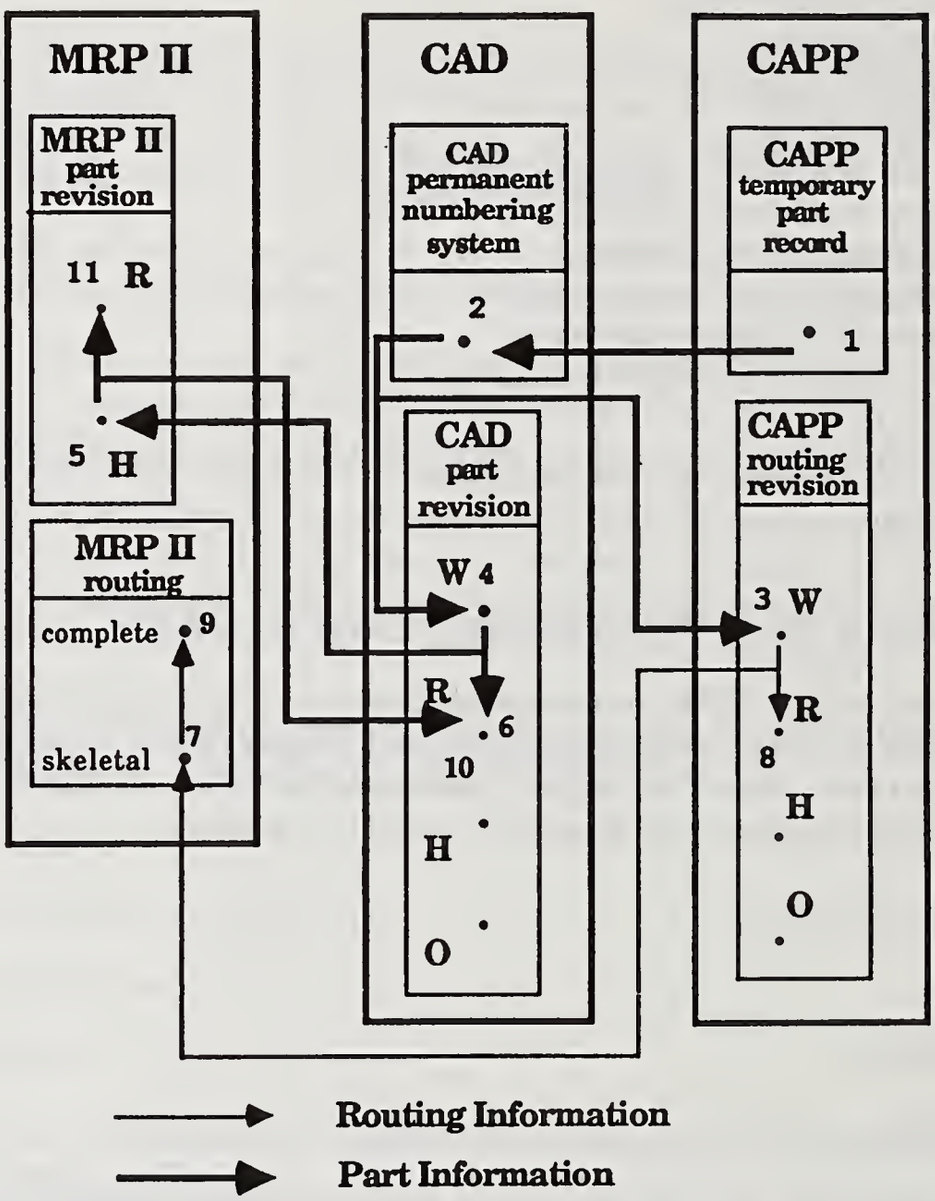


Figure 1: New Manufactured Part in CAPP

- Skeletal part and revision records are created within CAD with the status automatically set to 'working' (state 4 in figure 1). This is because the part design has not been finalized yet, and it is still being worked upon.

- In addition, a temporary record containing part data is created in CAPP, based on which the formal routing will be developed (state 3 in figure 1).

Once the design of the part is complete, CAD releases the part. A skeletal part master record and revision record are created for the part in MRP II. Those data fields maintained in MRP II, are initiated as 'unknown' within CAD, and are subsequently supplied by the MRP II users. In the case of the revision record, the status code is automatically set to hold 'h' (state 5 in figure 1), since many of the fields required by the MRP II system have to be completed before MRP II can consider the part active. MRP II also has to wait for CAPP to release the routings for the part. At this stage, the design is complete, whereas CAPP may still need to finilaze the routings. In addition, MRP II needs to have the part master record established before it can accept the part routings. On the success of these events, the part revision is given a 'released' status in CAD (state 6 in figure 1). MRP II can then start working on the part record of the part in its database. It fills in whatever information required is available, and also waits for CAPP to release the routings before it can proceed any further.

When CAPP finalizes the routing for the part, the routing is given a 'released' status in CAPP (state 8 in figure 1). Certain checks are carried out before the release can be successful. A check is made to make sure that the part for which the routing is being released, has a released status in CAD. If it does not, a message to this effect is generated in CAPP, and the release is not possible. In addition checks are made to ensure that the routing being released exists, with working status. On the success of these checks, the routing is given a released status and a skeletal routing is updated immediately to the MRP II routings module (state 7 in figure 1). At this point, the temporary part record based on which the routing was developed, is no longer required in CAPP and it is deleted.

Finally when MRP II completes all the data fields related to the part record, and the routings record (state 9 in figure 1), it gives the part revision a released status (state 11 in figure 1). The effectivity dates are downloaded to the CAD revision record (state 10 in figure 1).

3.2 New Revisions of Manufactured Parts in CAPP

As mentioned above, CAPP does not support part data. Despite this however, CAPP is given the facility to initiate new revisions of the parts it entered into the system. This is because the tooling type parts initiated by CAPP are used primarily by manufacturing, and therefore CAPP receives feedback regarding their performance. Any design change suggestions would therefore logically stem from CAPP users. CAPP initiates a new revision by inserting the following revision data record into the system (state 1 in figure 2).

1. Part Number

2. New Revision Level

3. Drawing Number

The insertion of the above record automatically creates a new revision record in CAD, with working status (state 3 in figure 2), subject to the condition that the part number entered by CAPP users exists, and the part was originally created by CAPP. It should be mentioned that since CAPP does not maintain part records, it cannot handle part revisions itself . Therefore, although CAPP can initiate the new revision in the system, the actual engineering change procedure occurs under the jurisdriction of CAD, which is the design center in the integrated system. Establishing the new revision level in CAD also results in the creation of a temporary part record in CAPP, based on which the routing for the part revision is created (state 2 in figure 2). This temporary record is automatically deleted when the routing is finalized and given a release status in CAPP.

CAD users then start to work on the design change to the part, with the help of CAPP users. CAPP users must be involved in this process of design change since the part originated in CAPP and is primarily used by the people in manufacturing. If found necessary, the current revision can be placed on hold on CAD, which automatically invokes a hold in MRP II. When the design of a new part revision is finalized, it is released in CAD. As in the case of creation of a new part, a skeletal revision record is created in MRP II, with the effectivity start date left unknown if not provided by the designer. In this case, the date is often specified by the CAD and CAPP users, to be as early as possible. This is because the part is used in the manufacturing operation and is not a component of a product part. MRP II is therefore not concerned about using up the current inventory before making the new revision effective. The status of the new revision is automatically set to hold in MRP II (state 6 in figure 2), as it has to prepare for the release of the new revision. The old revision is automatically given an obsolete status in CAD (state 5 in figure 2), since only one revision can be active at any given time. Upon the success of all these operations, the release of the new revision is successful in CAD (state 7 in figure 2). In the case of MRP II, obsolescence is handled with the help of effectivity dates. The effectivity start date of the new revision becomes the effectivity end date of the old revision, and from that date onwards the old revision is considered obsolete in MRP II.

MRP II then waits for the routing of the new revision from CAPP. CAPP has to wait for CAD to release the revision before it can give the routing a released status, assuming it is complete. When the routing is complete in CAPP, and the new CAD revision has a released status, it is released in CAPP (state 9 in figure 2). This results in a skeletal routing being created in MRP II (state 8 in figure 2). When MRP II completes all the data fields in the skeletal routing (state 10 in figure 2), it gives the new part revision a released status (state 12 in figure 2). On release in MRP II, the effectivity start date is automatically downloaded to the CAD revision record, in the event that it was left unknown at the time of release of the revision in CAD (state 11 in figure 2).

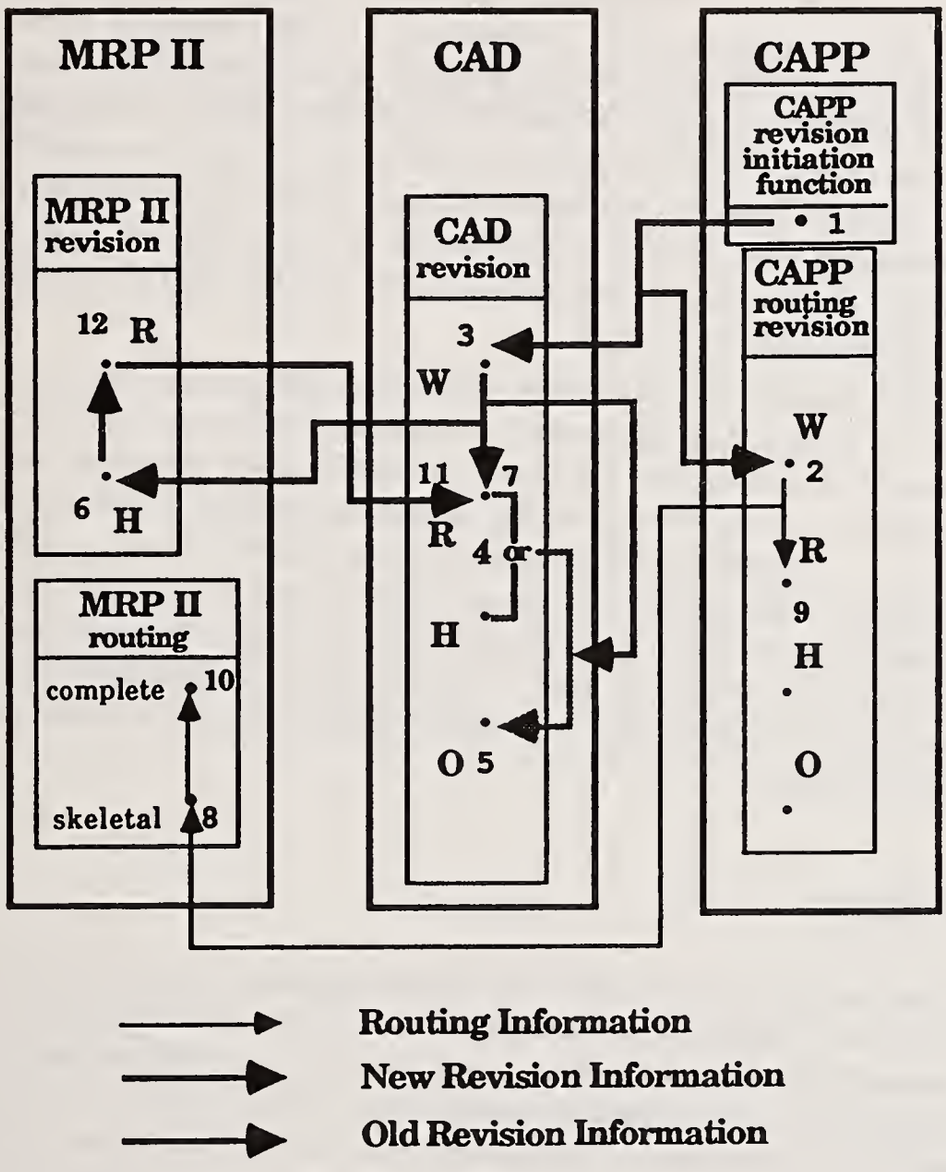


Figure 2: **Revision of Manufactured Part in CAPP**

4 System Implementation

The concept of utilizing individual application-system databases was adopted in this work in preferance to the traditional approach of having a large centralized database accessed by each application system. The approach used here is based on database interoperability. Database interoperability is the concatenation of the schemata of data of each of the databases of the application systems, along with a rule set constructed for each separate database, called update and retrieval dependencies. These update and retrieval dependencies control inter-database consistency through inter-database operation calls [1]. The Update Dependencies language has been developed specifically for implementing this database interoperability concept, and a first version of the interpreter has been implemented in Prolog [4]. The functional relationships between CAD, CAPP and MRP II have been specified in the language and tested.

The aim of this research was to provide a generic model for the integration of CAD, CAPP and MRP II, rather than integration of existing commercial packages. For the purposes of demonstration, test, and validation, certain routines of generic systems were recreated, and embedded in the interoperability system as specified by the Update Dependencies language. In an actual commercial implementation, this would not be necessary since existing routines in the application system could be utilized. In that situation, the interoperability system would be transparent to the users as they would interact with the commercial software instead of directly with the interpreter as is done here. Consequently, the volume of the source code specifying the interoperability system would be much smaller in an actual implementation.

5 Conclusion

The work presented here gives a new approach to the integration of manufacturing systems at the facility level. An integration methodology is presented which eliminates the need for data duplication and controls consistency by use of the database interoperability approach. The integration focuses on three specific modules, these being CAD, CAPP and MRP II and is based on the commonality of data between particular entities used in the various modules. A rule base was developed to govern the manipulation of the data associated with these entities through the use of status codes. The system was implemented by translation of the rule base into the Update Dependencies language developed at the University of Maryland.

References

[1] Bohse, M.E. 'Integration of Manufacturing Resource Planning And Computer Aided Design Through Database Interoperability' *MS Thesis* , Department of Mechanical Engineering, University of Maryland at College Park, (1987).

[2] Bray, Olin H. 'Computer Integrated Manufacturing, The Data Management Strategy' *Digital Press, CIM series* (1988).

[3] Harhalakis, G.; Ssemakula, M.E. and Johri, A. 'Functional Design of an Integrated CIM System at the Facility Level' Int. J. Computer Integrated Manufacturing, vol. 1, no. 4, (1988) pp 245 - 252.

[4] Mark, L. and Roussopoulos, N. 'Operational Specifications of Update Dependencies' *Technical Report*, Department of Computer Science, University of Maryland at College Park, (1987).

A COOPERATIVE SHOP-FLOOR CONTROL MODEL FOR COMPUTER-INTEGRATED MANUFACTURING

JAMES J. TING

Abstract:

A cooperative shop floor control model has been proposed in this paper. The control model is able to address the aspects of management, production, and transportation in a unified and asynchronous fashion. The model incorporates a pool control method with differed resource commitment and distributed job scheduling. With such control method, each module is independently and symmetrically controlled and the failure of module as well as module controller can be coped with without interfering the operations of other modules. The model also allows a simple and demonstratable correspondence between physical modules and control software objects to take the advantages of object-oriented design and programming approaches.

1.Introduction:

A common control model found in many computer integrated manufacturing (CIM) architectures is *supervisory control*, also called hierarchical or command-feedback control [COH88] [KEO87] [JON86]. Two major factors contributing to the popularity of the supervisory model are its similarity with traditional production management and control hierarchies and its conformity to the classical problem solving paradigm, top-down decomposition. The similarity with traditional management and control hierarchies makes the supervisory control model easy to comprehend for management people; the conformity to classical problem-solving method makes the model easy to apply for engineers. Therefore, the model is often the choice in upgrading a conventional factory into a CIM facility.

However, as the local controllers and communication networks on the shop floor become increasingly intelligent and powerful [BEN89] [SAN89], the traditional supervisory model tends to inefficiently utilize such local intelligence and power in a modern CIM system [DUF86] [HAT85]. A different approach, *cooperative control*, also called heterarchical or negotiation-based control, has been proposed to take advantage of the modern computer control technology. In the cooperative model, the local controller possesses more control knowledge and shares more control responsibility than its counterpart in the supervisory model. Furthermore, these local controllers are equally accessible to each other, have equal right of access to resources, operate independently, and strictly conform to some overall system rules [HAT85]. With such a control model, there is no need for a supervisory level and the exchange of local information through the supervisor. This results in a simpler and more efficient CIM architecture. In addition, the cooperative model allows a higher level of control distribution which in turn results in better flexibility, operatability, and modifiability.

Despite the promises of the cooperative control model, many important issues remain to be solved:

- what is the role of supervisors in cooperative control,
- how to efficiently store part data and exchange information,
- how scheduling can be distributed,
- how to achieve global production optimality with distributed scheduling techniques,
- how the cooperative model performs in large scale flexible manufacturing systems,

- what philosophy should be used in the design of software so that objectives such as reduced complexity, high fault-tolerance and low development cost are achieved.

Most of the existing research on cooperative control only provides primitive control protocols and fails to address most of these important issues. To remedy the shortcomings of the existing cooperative control models, an *alternative cooperative control model* is proposed in this paper to address the above issues.

2.The shop-floor control problem

The control of a manufacturing system can be decomposed into four major layers: corporate, factory, shop floor, and device layers; in some CIM reference models, the shop floor layer may be further divided into shop, cell, and station layers [JON86] and the device layer into machine, sequencing, and servo control layers [KEO87]. The corporate layer consists of several (possibly geographically separate) facilities/factories that are coordinated to produce a single product or product line. The factory layer is comprised of all engineering, business, and production operations (often) at one geographical location. The shop floor layer consists of loosely coupled manufacturing modules which are physically located in one area/shop and logically related in one production process. The device/operator layer corresponds to the manufacturing devices within each module which are closely coupled by the requirements for real-time synchronization, proprietary devices, etc. or human operators who are managed as one group. These layers are respectively responsible for strategic planning, operational planning, inter-module production control, and inter-module device motion control or operator dispatching. The control problem at the shop floor layer is the main concern of this research.

2.1 Context

To accomplish manufacturing integration, it is essential to identify not only all manufacturing components but also their relationships. This section describes the context of shop floor control; Figure 1 illustrates the relationships of shop floor control with other CIM components.

The shop floor control function interfaces with factory planning, device control, purchasing and distribution, and personnel management functional entities in a manufacturing enterprise. Shop floor control, factory planning and device control are all part of CIM application functions which carry out the primary missions of a manufacturing plant. The factory planning function determines the master production schedule to be used as a base line for shop floor control while receiving back the production results and actual shop-floor production capacity for further operational planning. After the shop floor control function decides which module will perform what jobs, the device control functional entities within each module will receive the job notices from the module controller and perform necessary operations. The devices control entities also report any status changes of the module to shop floor control entities for more accurate resources allocation among modules.

Besides the interfaces with primary manufacturing functions, the shop floor control is also related with external functional influences. As defined in [COH88], the external functional influences are the entities in a manufacturing enterprise which are separate from production functions of a manufacturing plant but affect them by sending inputs. The external functional influences which affect shop floor control are raw materials purchasing, finished products distribution, and personnel management. The purchasing and distribution functions affect the actual stock level of raw materials available for real-time allocation to shop floor production modules and the actual storage space for storing finished products. Shop floor control, in turn, determines the amount of finished products available for distribution. On the aspect of human resources, the personnel management affects the availability of manpower on shop floor and receives actual personnel performance/attendance records from shop floor control function for evaluation and accounting purposes.

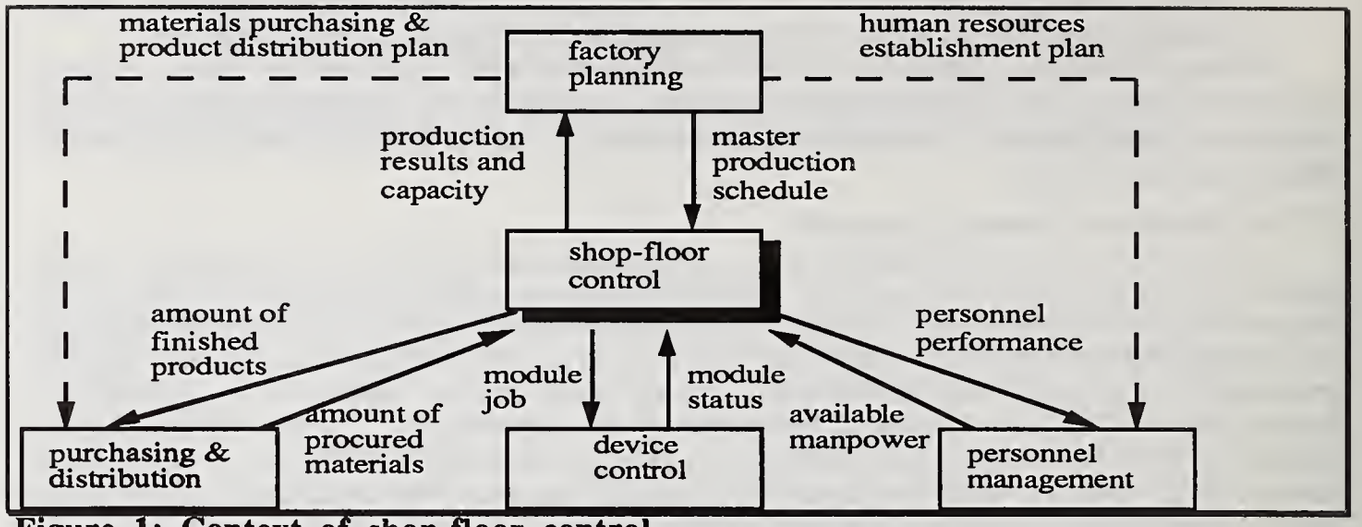


Figure 1: Context of shop-floor control

2.2 Scope

The scope of shop-floor production control, besides intra-module communications, is primarily module-level job scheduling, progress monitoring, and status reporting. While both shop-floor control and factory planning involve scheduling, they are different in granularity, objective, and method. The shop floor scheduling deals with immediate and detailed resources assignment based on a master schedule and real time shop floor status, and its objective is to reduce system disruption caused by changes which were not anticipated during planning. On the other hand, the factory layer scheduling deals with plan-ahead and coarser resources allocation based on long-term production goals, and its objective is to optimize productivity and balance the production and inventory costs. Therefore, the shop floor scheduling system is more likely to use distributed and dynamic scheduling techniques for their flexibility and quick response to local changes while the factory scheduling system tends to use centralized and static scheduling techniques for their efficiency in decision-making process and ability of providing optimal solutions.

Shop floor and device controls are also similar in that they both involve real-time control. However, the shop floor control is less time critical and not restricted to proprietary devices and synchronous methods. This is because that the shop floor layer aggregates manufacturing devices into coherent modules, and encapsulates the part of control problem involving very fast response time, proprietary device interfaces, and real-time synchronization within each module. With the modularization, the shop floor layer only deals with inter-module job control and leaves the intra-module operation (or motion) control to the device layer. The modularization also relieves shop floor control from the restriction to proprietary devices and permits standards for shop floor control to be adopted more easily. Finally, the modularization allows asynchronous methods to be used for shop floor control since necessary real-time synchronization actions are encapsulated within each module.

In summary, the shop floor control problem involves immediate and detailed production scheduling, progress monitoring, and status reporting at the module level. The objective of shop-floor control is to provide better system flexibility, operatability, and maintainability so that disruption caused by unexpected changes can be greatly reduced. Standard and distributed control methods are better suited to achieve this objective than non-standard and non-distributed ones. The focus of this paper is on providing a distributed shop floor control model based on standard communication networks.

3.Existing shop-floor control models

While there are variations in the definition of a supervisory model, they generally agree in that "each control module decomposes the current input command from its supervisor into procedures to be executed at that level, subcommands to be used to one or more subordinate modules and status feedback sent to the supervisor" [JON86]. This command-feedback relationship among manufacturing modules is very similar to that in traditional production management and control hierarchies; it also conform to the classical problem solving paradigm, top-down decomposition.

Although many supervisory control models claim to be distributed, they are at best network-based and not truly distributed models [LEL83] [COU88]. In a system based on such models, the lower level components passively wait for the commands from their supervisory components to initiate their production activities. The supervisory and subordinate components may be on separate computers and the commands may be sent and received through a communication network, whereas the control relationship between these components is similar to that between main routine and subroutines in a centralized system. Furthermore, the supervisory component needs to possess some global knowledge about the world it controls: which components constitute the world, what they can do, and how they are doing. Such a global knowledge requirement also excludes the supervisory model from being truly distributed since no global knowledge should be assumed as a prior to the proper operation of the components in a distributed system according to criteria described in [LEL83] [COU88].

With the elimination of the supervisory relationship, cooperative control models have been able to offer prospects of reduced complexity, improved operatability, and increased expandability and maintainability over supervisory models [HAT85] [DUF86]. In a system based on such a cooperative control model, the supervisory control function is distributed to all subordinate components which cooperatively perform the control function. These components have equal access to resources, operate independently, require no global knowledge as a prerequisit to their proper operation, and all strictly conform to the cooperation rules [HAT85]. The unified and symmetrical relationship between components reduces the complexity of control structure. In addition, the independent and symmetrical operation mode of the components makes the cooperative control system more resilient to failures as well as more expandable and maintainable. Some of the heterarchical control models have been proposed in literature [SHA88] [DUF86] [LEW87] and they can be categorized as controller driven, intelligent part driven, or data flow respectively.

3.1.Controller driven

The controller-driven model is based on a bidding mechanism similar to the contract net protocol [DAV83]. A new job entering the shop floor control network may go to any of the shop floor controllers. The controller which receives the job order, called manager of the job, will broadcast a task announcement into the network. Every controller including the manager may bid for the task within a predetermined period of time, called a bidding interval. The manager collects all bids and evaluates them after the deadline for bid submission. According to certain criteria, the best bid is selected by the manager, and the task is awarded to the best bidder. When the module finishes the task, the module controller will check to see if there are any remaining operations to be done. If all operations of the job have been completed, the corresponding workpiece is sent to the storage area; otherwise, the controller becomes the new manager of the remaining job and starts the task assignment process again. Shaw [SHA87] [SHA88a] describes such a the controller-driven protocol.

3.2.Intelligent part driven

The intelligent part driven model is also based on a bidding mechanism similar to that of the controller driven model, but this model defines the intelligent part software, instead of work module controller, as the task manager who in charge of the bidding process. An intelligent part software can be viewed as a dynamic information storage and processing unit which maintains its

own quality control history, performance measures and due dates, and variable process plan. For each physical part type, there is a corresponding intelligent part software. When a physical part (batch) enters the input station, its part software gets invoked on one of the controllers. The part software will broadcast a task announcement, collect bids from the controllers, evaluate the bids, and choose the best bidder to award the task to. The part software will then request the transportation system to move its physical counterpart to the the award winning work-module. The part software is also capable of immediately analyzing the results of part inspection and initiating corrective actions if necessary. Two variations of this type of protocol are defined by Duffie and Piper [DUF86] and Maley [MAL88a] respectively.

3.3.Data-flow

Like the controller-driven model, the data-flow model describes production tasks as plain data, not intelligent part programs; but unlike the controller-driven model, the data-flow model defines a specialized control component, called Module 0 or M_0, for introducing jobs into and removing them from the communication network. A job is an indexed sequence of all the tasks to be performed on a batch of parts. M_0, generates messages describing jobs on hand, places the current task pointer to the first task(s) of each job, and then broadcasts the message into the network. After all tasks of a job have been carried out, M_0 removes the job from the network. M_0 also has to assure that there are always a fixed number of jobs circulating in the system in order to make the model's control policy valid.

The claiming mechanism used in the data-flow model for job scheduling is also different from the other approaches' bidding mechanism. The task claiming process starts with each controller listening to the job messages, saving the jobs whose current tasks can be performed by the work-module in an internal FIFO queue, and discarding the others. When a work-module becomes or almost becomes idle, its controller dequeues the first job from the internal job queue and claims the job by broadcasting a claim message. If there is more than one controller simultaneously claiming the same task, they all withdraw their claim messages, independently and respectively wait for a random length of time, and then re-claim the task.[1] The successful controller requests materials needed from the materials handling system; at the same time, the other controllers remove the claimed job from their internal queues. Upon finishing a task, the controller will return the material, advance the job's current task pointer, and re-broadcast the message. The controller may re-broadcast the message without performing the claimed task if the requested materials are not available. The data-flow model was first developed by Lewis et al. [1982]. The rational behind this model is that by increasing the average number of occupied work-modules, the manufacturing system will 1) increase the throughput and 2) decrease the mean production lead time [LEW 1987].

3.4.Summary

The existing cooperative control models are able to make more efficient use of local controllers and communication networks and provide the controlled manufacturing system with better operatability than supervisory models. The cooperative control models distribute supervisory control functions to subordinate controllers and make the local controllers share more control responsibilities than their counterparts. Hence, the increasing power and intelligence of local controller and communication network can be better utilized in a cooperative control models. The distribution of supervisory control functions also permit the control functions to continue with some local controller failure. Therefore, the underlying manufacturing system can continue with degraded operations in response to machine or controller failure. However, the existing cooperative model provide only a partial solution to shop-floor control problem and fail to address the issues discussed in Introduction section.

4.An alternative cooperative control model

[1] This is similar to the conflict resolution strategy used in the communication protocol CSMA/CD.

To remedy the shortcomings of the existing cooperative control models, a *cooperative control model* is proposed in this paper to address the above issues. The domain of the model is the shop-floor control layer of a flexible and integrated manufacturing system. At the shop-floor control layer, the manufacturing system can be viewed as a network of work, storage, transport, and management modules. The work, storage, and transport modules are responsible for and capable of coordination of their own production schedules through a distributed dynamic scheduling method defined by the control model; the management module takes the role of monitoring and reporting production progresses of the other modules and managing shop floor personnel. This new model also distributes module-specific data to local module controllers while the shared and part-specific data are stored in a data/software storage module. The model tackles the global production optimality problem by implementing a "pull" control strategy with local optimization in the job scheduling method. The model addresses the software design problem by taking an object-oriented approach. Finally the model performance for large scale FMSs will be analyzed by a simulation tool which can be used for evaluating the performance of any proposed cooperative controlled manufacturing system.

4.1 Assumptions of the underlying system

At the shop floor layer, a flexible and integrated manufacturing system is viewed as a collection of manufacturing modules which are connected by a standard communication network and an automatic transportation network (cf. Figure 2). The shop floor modules can be either management, transport, or production modules; the production modules can be further classified as work, or storage modules. Each module has one control computer which is responsible for the shop-floor-layer control of the module, inter and intra module communications, and data/software storage and retrieval. Production modules also have local buffers for storing raw materials needed for the next job and/or finished parts produced from the last job. The work modules can be in any layout, cellular, functional, or hybrid, as long as they are formed to encapsulate proprietary equipment and its control or to group devices which require real-time synchronization. An example of such a module is a transfer line controlled by Septor's System 90 [VAS87].

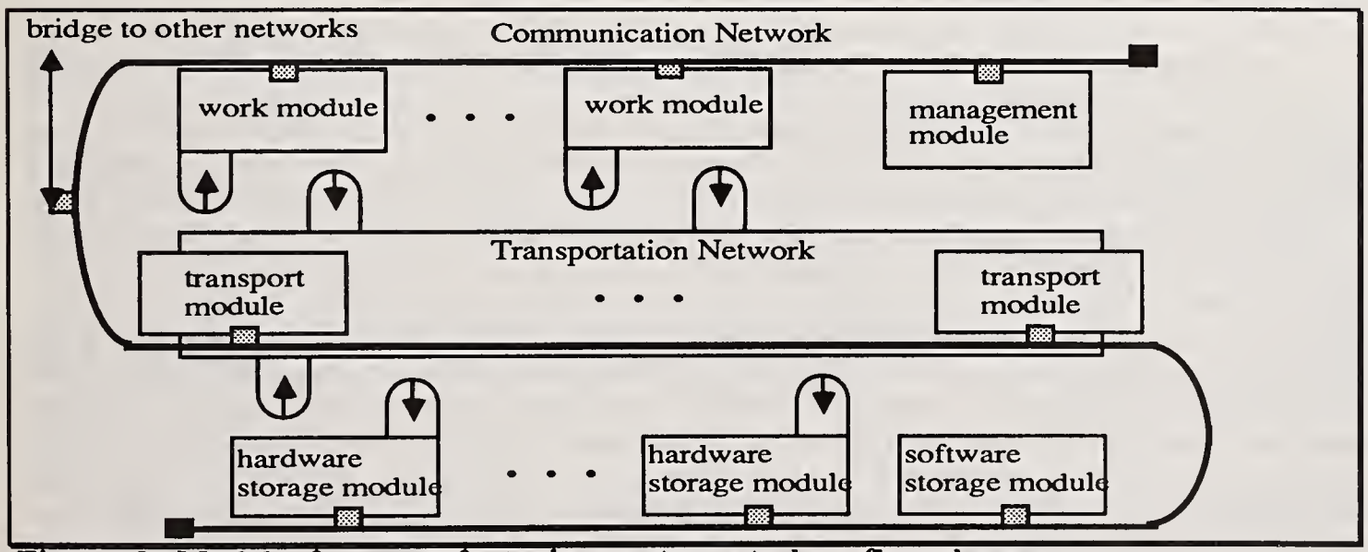


Figure 2. Model of a manufacturing system at shop floor layer

The shop floor transportation network is a key component for manufacturing flexibility and itself has to be flexible. To make the network flexible, the network operating functions, such as path selection, collision prevention, and job scheduling, are assumed to be distributed to the transport modules running on the network. As a result, these transport modules are autonomous, i.e. knowing which modules they can reach and how to get there, and independent, i.e. scheduling

their own production jobs. The transport modules are also assumed to be quick response and small capacity vehicles so that they do not require multiple parts to make up a full load and could transport single parts [MAL88a] [MCG86]. Furthermore, the failure of a transport module will not completely block the access to any of the production modules.

While human operators may drive or walk on the shop floor, the model assumes they are moved by the same transport modules used for carrying materials. The single transportation means allows unified road design and better traffic control.

Another key component for manufacturing flexibility and system integration is the shop-floor communication network. The communication network connects all module control computers (or module controllers) to provide a quick and accurate way of exchanging information on the shop floor. This network is assumed to be a plant-wide Local Area Network (LAN). As oppose to other communication networks, LANs have the following characteristics: [IEE86]

- optimized for moderate geographic area such as one or a few close-by buildings,
- providing moderate to high data rate, low delay, and low error rate, and
- usually owned by a single organization.

While there may be several different types of LANs interconnected in a manufacturing facility, this shop-floor control LAN is assumed to be a single LAN or several similar LANs connected by repeaters so that they function as a single LAN.

To make the communication network comply with standard LANs described in Stallings [STA87], further assumptions on the underlying LAN operating system are made as follows: it provides asynchronous communication between controllers; it guarantees the delivery of messages[2] while imposing an unpredictable transmission delay; and it does not necessarily follow FIFO policy on the transmission of messages from multiple nodes, and the FIFO policy holds on the messages from a single node. An example of non-FIFO transmission network is the CSMA/CD based LAN [STA87].

4.2.The control model

4.2.1 General Framework

Pull control with deferred commitment: The proposed model incorporates a pull control method with deferred commitment. The pull control method is based on the principle of Kanban system, which was first developed and used at Toyota [MON81] [SUG77]. With this method, production jobs are introduced to the orders of end-products. The production modules which can produce the products will record the jobs and broadcast sub-jobs to procure the parts needed to perform the jobs if the parts are not locally available. The same part procurement process will propagate through the production system until some production modules have all required parts to perform the (sub-)jobs. Such production modules may very likely to be raw material storage modules at the beginning of the production stream. These modules than start a job claiming process to determine who wins the job. The winner of this claiming process needs to broadcast transport jobs to have the parts moved over if they are not in local buffers. With all necessary parts on hand, the winning module then performs the job and broadcasts a job-done message when the jobs are finished to notify downstream modules about the availability of their inquired parts. The transport jobs are claimed by transport modules in the same way as production jobs by production modules.

This pull control method is different from traditional Kanban system in its deferred resource commitment. In the Kanban system, a kanban is a card used to authorize either a production or a transport job as the production or transport job notice does in the above pull control method. When the Kanban system is implemented in a transfer-line setting [MON81], each module is pre-assigned a small number of production kanbans and is limited to perform the pre-assigned

---

[2] Unless there is a hardware failure which jeopardizes the proper function of the network.

production functions; while in a job-shop setting [GRA88], the first idle module can select a production kanban from the central decision board on which all kanbans are kept. Unlike the Kanban system, this pull method does not commit production jobs or parts to particular modules or operations until all required resources are available for immediate performance of the jobs. The allocation of resources to the jobs is done by a job claiming process which yields an locally optimal solution. Such deferred commitment approach brings about more flexibility and quicker response to unexpected changes. Such a pull control method is illustrated in Figure 3.

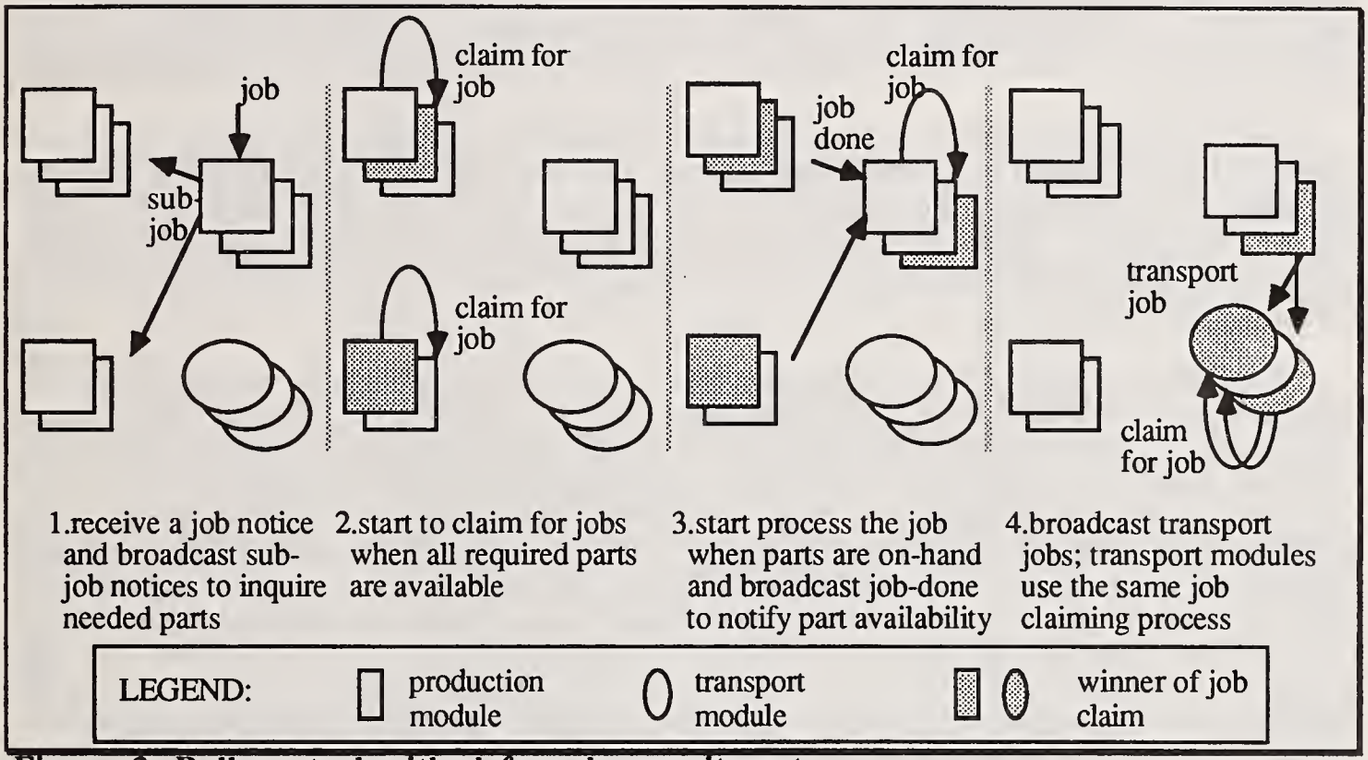


Figure 3. Pull control with deferred commitment

Failure handling: Both manufacturing modules and controllers may fail although the failure rate of manufacturing modules, which consist of mostly mechanical devices, is usually higher than that of controllers, which are generally electronic computers. The module failure is easier to cope with by simply modifying the capacity list in the controller to reflect the disability of the module. With the modified capacity list, the controller will automatically skip all the jobs which cannot be performed by the module. If the module failed after jobs have been won and parts have been transported to local buffers, the controller will re-announce the unstarted jobs and make the already shipped parts available to public again. The started but unfinished jobs will stay in the module and will continue with manual control by operators once the module is repaired. The repaired module may joint other modules to claim for the re-announced jobs which are still open at the time.

The controller failure is harder to cope with since it may result in some information loss. The information loss caused by controller failure can happen in one of the following situations:

1. a job is announced when no controller of capable modules is in normal operation at the time
2. a job has been announced and yet to be claimed when all controller of capable modules for the job go down
3. the claiming process for a job has been started and the controller which has broadcasted the best bid fails before it completes the process

4.a job has been won by a module whose controller subsequently failed before it can finish the job
5.a job-done message is broadcasted when no controller which is capable of performing the subsequent job is in normal operation

To recover the lost information in situations 1, 2, and 4, the above generic pull control model is modified as follows: a controller which is waiting for parts to perform a job will periodically re-announce sub-jobs of the job if the controller is not hearing any production activity for the sub-jobs (or sub-jobs of the sub-jobs of ... of the sub-jobs) from upperstream module controllers. The success of the modified model relies on some implementation which allows quick detection of the parent-child relation among jobs. One possible implementation is to use simple job ids and attach parent job ids to the front of each job id, e.g. 2.3.5 represents the fifth sub-job of the third sub-job of the second job.

Situation 3 is handled in a distributed job scheduling/claiming method described later. Situation 5 can be handle in a similar way as above. The controller which broadcasts a job-done message will periodically re-broadcast the message until the claiming process for the job's parent job has been started. With this addition to the base model, the part availability information conveyed by the job-done message will not be lost due to temporarily the absence of modules picking up the information.

Flexible lot sizing: While large lot size will reduce the possibility for parallel production and decrease the flexibility for failure handling, small lot size will increase the frequency of decision making and module setups. This control model allows flexible lot sizing to provide both the production efficiency of large lot size and the failure handling flexibility of small lot size.

The flexible lot sizing is done by allowing production modules to claim for only a small portion of a job order each time and can continue claim for the same job when it almost finishes currently on-hand work until the whole job is claimed. The job claim will include the estimated time for setup besides the process and transportation times. By considering the setup time, a module which is performing the same job tends to win the claim and to cumulate the small lots into a large lot. Should the module fail, it immediately stops claiming for more jobs and the remaining unclaimed part of the job can be performed by other modules without having to be retracted from the failed module. In addition, a large job can be naturally performed in parallel if there are more than one capable module and all claiming for only a small portion of the job each time.

To implement the flexible lot sizing method, the underlying communication network has to accommodate the increased communication traffic due to increased frequency of job claiming, the underlying transportation network needs to incorporate the increased transportation traffic as a result of smaller transport lots, and there should be no difficult in splitting production orders into small lots.

Role of supervisor: The management module in the proposed control model plays the role of shop floor supervisor. The management module consists of managers, foremen, and operators. Since the production scheduling work is done by all the modules cooperatively, the management module is responsible for only progress monitoring, status reporting, and shop floor personnel management. Progress monitoring is done by passively listening to the broadcasted job win and done messages. Based on these messages, the management module can maintain a work history of all under-supervised production and transport modules. From the history, information about product quality, production lead time, and module idle time can be derived and used to determine the schedule for preventive maintenance of each module. The information is also reported to operational planning systems at the factory layer for planning future production.

When a module's maintenance is due, the management module sends a maintenance job notice to the module. The maintenance job will be handled as a normal production job by the module except only one module will claim for the job. The module will request for resources, including tools and operators, claim for the job once all the resources are available, then request for

transportation of these resources to local buffers, and finally perform the maintenance job when all resources have arrived. The maintenance job usually does not have higher priority than normal production jobs so that the production jobs which are claimed and won before the maintenance job will be finished before maintenance to ensure non-blocking of the down stream production by the module. Consequently, the time of actual maintenance won't be necessarily the same as original scheduled, and won't be too much different either since the production jobs are all fairly small.

When a module breaks down unexpectedly, it sends repair jobs to request for operator services from the management module and other resources as required from different modules. While the repair job notice is the same as the resource request for maintenance jobs, it has priority over all other types of jobs: the management module will first assign operators to repair job and the transport module will first ship repair resources.

Local optimality: To ensure that the choice of the winner for a job claim is at least locally optimal, each module can only claim for one job at a time. If multiple claims are allowed, one module might win two jobs during the same claiming period. However, the bids included in the winning claims are based on the module capacity before any job has been won. If both jobs are performed at this module, one of the jobs has to be delayed and the claim information becomes false. The choice of the winner for the job was based on false information and could not guarantee to be the best.

With the restriction of one claim at a time, a module's bid for a job reflects its true capability when the job is actually conducted. Therefore, the best bidder for a job will yield an locally optimal performance among all possible modules for this particular job. The choice of best bidder for each job is achieved in this control model by a scheduling algorithm which selects the best bidder for a job during each claiming period. Since the claiming process is through a communication network, the time for claiming a job will be very short compared with the time for actually performing the job. Consequently, even a module can claim for one job at a time, it can claim for many jobs and has all opportunities to win one before it finishes the last job and becomes idle.

Non-blocking pallet/fixture return: In an automated manufacturing system, pallets and fixtures are needed for loading and fixing parts on vehicles and machines. After the process is done, the parts will be unloaded from the pallets and fixtures. If the unloading station is in different module from the loading station, the empty pallets and fixtures need to be shipped back to loading station and reused on other parts. It is very desirable that the shipment of the empty pallets and fixtures should not block that of materials and parts.

The return of empty pallets and fixtures can be incorporated in this model by letting the storage module issue a production job solicitating empty pallets and fixtures for each used pallet or fixture. To avoid blocking normal traffic, the empty pallet/fixture return job can be treated as low priority jobs by the transport modules and are shipped only when the transport modules have nothing else to do; or the pallet/fixture return jobs can be issued at the end of each day of work when all other jobs have been finished.

4.2.2.Distributed and dynamic scheduling

A distributed and dynamic scheduling method is used in the model's job claiming process. This scheduling method is different from the bidding mechanisms of existing control models in that it is more distributed in terms of imposing no decision center, requiring no prior knowledge about global state, and allowing various heuristic rules to be used in job sequencing and dispatching. The method is different from existing distributed election algorithms [RAY88] in that it tolerates controller failure and can be used not only on ring-topology networks but also on token-bus and CSMA/CD networks.

The scheduling method is based on the Transmission Axiom: in a LAN or a group of repeater-connected LANs, the transmission of a message will take over the whole message-sending

channel at least until the message is propagated through the network(s). With the Transmission Axiom, it can be easily proved that broadcasted messages are heard in the same sequence by all active nodes on the network. Since all claims for jobs are broadcasted in the control model, every active node (or module) can expect to receive the same sequence of claims for each job. By examining the sequence of claims, each module knows from whom it can expect endorsement[3], to whom it should send its endorsement, or if it wins the job claim according the following rules:

- each module sends its endorsement to the first module that has broadcasted a better claim for the same job than the local claim once it has received all expected endorsements.
- each module can expect to receive an endorsement from the module who's claim is worse than the local claim but is the best among the claims received before the local claim was received, or the module who's claim is worse than the local claim and received after the local claim and before a claim which is better than local claim .
- a module can broadcast a win message for a job claim to terminate the claiming process if it receives no better claim while having received all expected endorsements.

The distributed and dynamic scheduling method is textually symmetrical, i.e. the same algorithm is used by all controllers while each controller has a unique reference name and may behave differently according to different messages received. The algorithm is as follows.

For each module controller i
initialize the open job set, $J \leftarrow \{\}$
BEGIN
1. when $\exists$ an open job J_j
1.1. $J \leftarrow J \cup \{J_j\}$
1.2. local bid, $b_i(J_j) \leftarrow$ NULL
1.3. start the local claiming process for the job, $C(J_j) \leftarrow ()$
1.4. expected direct endorser list, $E(J_j) \leftarrow ()$
1.5. better bidder list, $B(J_j) \leftarrow ()$
1.6. LocalBidReceived $\leftarrow$ FALSE
2. when job bidding is permitted and $J \neq \{\}$
2.1. select a job J_{next} from J
2.2. prepare a local bid $b_i(J_{next})$
2.3. if $\exists$ a bid $b_k(J_{next}) \in C(J_{next}) \wedge b_i(J_{next}) <_{bid} b_k(J_{next})$, then
2.3.1. terminate the local claiming process $C(J_{next})$
2.4. else
2.4.1. broadcast local bid message $m(b_i(J_{next}))$
3. when the incoming message buffer is not empty
3.1. dequeue a message m(•)
3.2. if it is a bid message $m(b_l(J_j))$, $J_j \in J$, then
3.2.1. $C(J_j) \leftarrow C(J_j) \ll b_l(J_j)$
3.2.2. if $b_i(J_j) \neq$ NULL $\wedge$ $b_i(J_j) <_{bid} b_l(J_j)$, then
3.2.2.1. $B(J_j) \leftarrow B(J_j) \ll l$
3.2.3. if l should be a direct endorser, then
3.2.3.1. $E(J_j) \leftarrow E(J_j) \ll l$
3.3. else if it is a win message $m(w_l(J_j))$, $J_j \in J$, then

---

[3] A module's endorsement is the act of supporting another module in selection of the winner for a job claiming process or a message sent to the other module to indicate the endorsement.

3.3.1. terminate the local claiming process $C(J_j)$

3.4. else if it is an endorsement message $m(e_l(J_j))$, $J_j \in J$, then

3.4.1. remove l from $E(J_j)$

3.4. else if it is a controller failure message $m(f_l(J_j))$, $J_j \in J$, then

3.4.1. remove l from $E(J_j)$

3.4.2 remove $b_l(J_j)$ from $C(J_j)$

3.4.3 modify $E(J_j)$

4. when the incoming message buffer is empty

4.1. if (LocalBidReceive = TRUE $\wedge$ $C(J_j) \neq (b_l(J_j)) \wedge B(J_j) = () \wedge E(J_j) = ()) \vee C(J_j) = (b_l(J_j), b_l(J_j))$, then

4.1.1. broadcast a win message $m(w_i(J_j))$

4.1.2. terminate the local claiming process $C(J_j)$

4.2. else if $B(J_j) \neq () \wedge E(J_j) = ()$, then

4.2.1. BetterBidder $\leftarrow$ GetFirstEntry($B(J_j)$)

4.2.2. MessageSent $\leftarrow$ send an endorsement message $m(e_i(J_j))$ to BetterBidder

4.2.3. if MessageSent = FALSE, then

4.2.3.1 broadcast a controller failure message $m(f_{BetterBidder}(J_j))$

4.2.3.2 remove $b_{BetterBidder}(J_j)$ from $C(J_j)$

4.2.3.3 modify $E(J_j)$ to include new endorsers

4.2.3.4. Go To step 4.1

4.2.4. else

4.2.4.1. terminate the local claiming process $C(J_j)$

END

In the normal case, the procedure can be completed with n messages and the message complexity of the above algorithm is only O(n), where n is the number of modules involved in the claiming process. In the case of controller failure, the sending of endorsements cannot be completed due to the failure of receiving controllers; the endorsers have to

- broadcast a controller failure message
- modify its direct endorser list
- follow the normal endorsement procedure

The above failure handling steps make the algorithm capable of tolerating any controller failure and guarantee a successful job assignment within one run of the claiming process. The extra messages needed for failure handling are 2 for each endorser of a failed controller. In the worst case that all bids were received in a reverse preference order and the first half bidders failed before endorsement, the total extra messages will be no more than $(n^2 - n)/2$. The message complexity of this algorithm becomes $O(n^2)$.

Some desirable characteristics of this algorithm are: it is textually symmetrical, it needs no prio global knowledge, and it can incorporate various job dispatching rules in step 2.1 and module selection rules in the bid comparison function "$<_{bid}$" (cf. step 3.2.2 of the algorithm).

4.2.3.Combining pull with push control

A complete pull control system has potential problems if some sub-assemblies which contain critical parts have long production-lead times or if the size of a company is much smaller than the suppliers of raw materials and it is impossible to impose Just In Time (JIT) delivery dates to the suppliers. For a manufacturing process with critical sub-assembly parts which have long production-lead times, the pull control method propagates the long sub-assembly times into long delivery times for end-products. If the manufacturing system works to orders, the long delivery times means long waiting times for customers and, hence, less competitiveness in the marketplace.

Or if the source of raw materials is not controllable, the pull production control method will results in an unpredictable delivery time for end-products. This is, too, unacceptable for work-to-order type of manufacturing systems. To remedy the problems, the production of critical sub-assembly parts with long delivery times or the procurement of raw materials with uncontrollable sources need to be planned ahead of time, i.e. they will not be driven by immediate needs of downstream production but by pre-planned inventory levels which is part of push control.

This control model is able to combine pull and push control methods by inserting stock points at the end of the sub-assemblies which contains critical parts with long production-lead times. These stock points are hardware storage modules in the terminology of this control model. These storage modules receive production orders directly form the operational planning system instead of from downstream modules. Therefore, some sub-assemblies may progress ahead of the immediate needs by end-product assembly. But, the sub-assembly processes still controlled by the same pull method and the whole control process consists of segments of pull controlled processes as opposed to a single pull process. An example of such hybrid production control is illustrated in Figure 4.

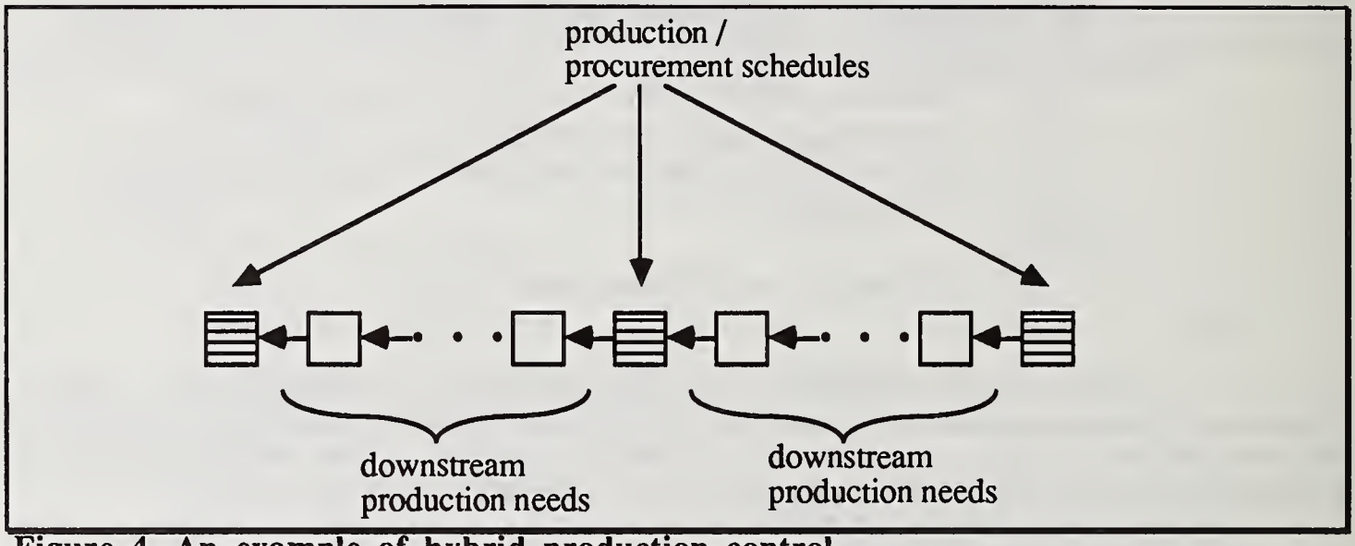


Figure 4. An example of hybrid production control

To deal with the uncontrollable resource of raw materials, an inventory control function is needed by the storage module of raw materials. The inventory control function keeps tracking the module's internal inventory level and re-orders raw materials from the supplier according to some pre-determined criteria or pre-planned procurement schedules.

5.An object-oriented control software design

A generic software model for shop floor control is designed based on an Object-Oriented Design (OOD) method. The principle of OOD is that the key to software quality lies in the structuring of the solution to a problem in such a way as to reflect the structure of the problem itself [BOO83]. Since the shop floor consists of clearly identifiable manufacturing objects such as machines, operators, parts, and work orders, there should be a simple and demonstrable correspondence between physical components on the shop floor and abstract components in the control software. The application of OOD in the domain of shop floor control should be very natural and even more beneficial than that in the domain of general computing [HAR89].

Although the concept of objects in software development has been around for a long time, the methodology for creating an object-oriented design did not exist until early 80's [ABB83] [BOO83]. Both Abbott's and Booch's methods contend that OOD begins with a natural language

description of the solution strategy for the software realization of a real-world problem. Following the informal strategy, four steps are defined to further specify the software [BOO83] [BOO86]:

- identify objects and their attributes
- identify operations which are performed or required by the objects
- establish interfaces/visibility between the objects
- decide on detailed design issues that will provide an implementation description for objects

The work of Abbott and Booch has been refined by EVB to provide a more rigorous although still informal step-by-step method for OOD [EVB86]. A somewhat similar alternative to the above method is given in [LOR86] to explicitly address some important concepts of OOD such as messages and inheritance. A good introduction to these two design methods with simple examples can be found in [PRE87]. Wirfs-Brock and Wilkerson argue that the data-driven approach, same as the above methods, makes the structure a part of object definition and may easily lead to a definition of operations which reflects the structure [WIR89]. They propose a different approach, responsibility driven approach, which first indentifies the objects' responsibility instead of objects' structure (the other objects known by the object). They argues that the data-driven approach makes the structure a part of object definition and may easily lead to a definition of operations which reflects the structure.

5.1.Basic terminology for OOD

This section provides a definition for the terminology used in the description of the control software model in the remaining section. The definition is somewhat different from that commonly found in OOP literature since OOD describes the software system at a higher level of abstraction than OOP which works on implementation details.

- *object*: is an entity which has states and whose behavior is characterized by the operations it performs and suffers
- *actor object*: an object that suffers no operations but only performs operations on other objects
- *server object*: an object that only suffers operations but performs no operations on other objects
- *agent object*: an object that serves some objects by performing operations on other objects
- *class*: a general description of one or many similar objects
- *instance*: a specific description of a particular object (or an object which is not a class)
- *meta class*: a class which is the generalization of some other classes
- *primitive class*: a class which is not a meta class
- *inheritance*: the automatic possession of some state values and operations of one class from some other class(es)
- *message*: a piece of information sent to an object to initiate operations of and/or request other information from the object with no hint or concern as to what the receiver should do to accommodate the sender's wish.
- *protocol*: a standardized set of messages and associated rules prescribing the object's behavior at the receipt of each message.

An object often corresponds or is analogous to a real world component and describes either the general property of a class of such components or a specific instance of such components. Objects are distinguished by their states and operations and fall into one of the tree types: actor, agent, and server. In the shop floor control problem, actor objects correspond to module controllers which require production operations of device controllers, service operations of human operators, and data collection/processing operations of the part and job entities; the device controllers in turn require production operations of the machine or other manufacturing equipment and are modeled as agent objects while machine, human operators, parts, and jobs carry out these operations and are modeled as server objects. The actor objects are not required by other objects to perform any operations. Instead, they participate in a cooperative control process through

passing messages. The messages have no control over the behavior of the receiving objects. But the actor objects all agree on the same protocol to keep the system's overall behavior in accordance.

5.2.Shop-floor control objects

Unlike intelligent-part driven approach to shop floor control (cf. section 3.2), the proposed control model describes module controllers as intelligent actor objects while the supporting entities, i.e. parts, programs, tools, work orders/jobs, etc., are designed as non-intelligent server objects. These non-intelligent objects will simply collect, process, and provide the data which describes the production status/information of their counterparts in the real world. The intelligent control objects contain the control knowledge and impose much of the difficulty in the design of shop floor control software. This research focuses on the design of these intelligent control objects.

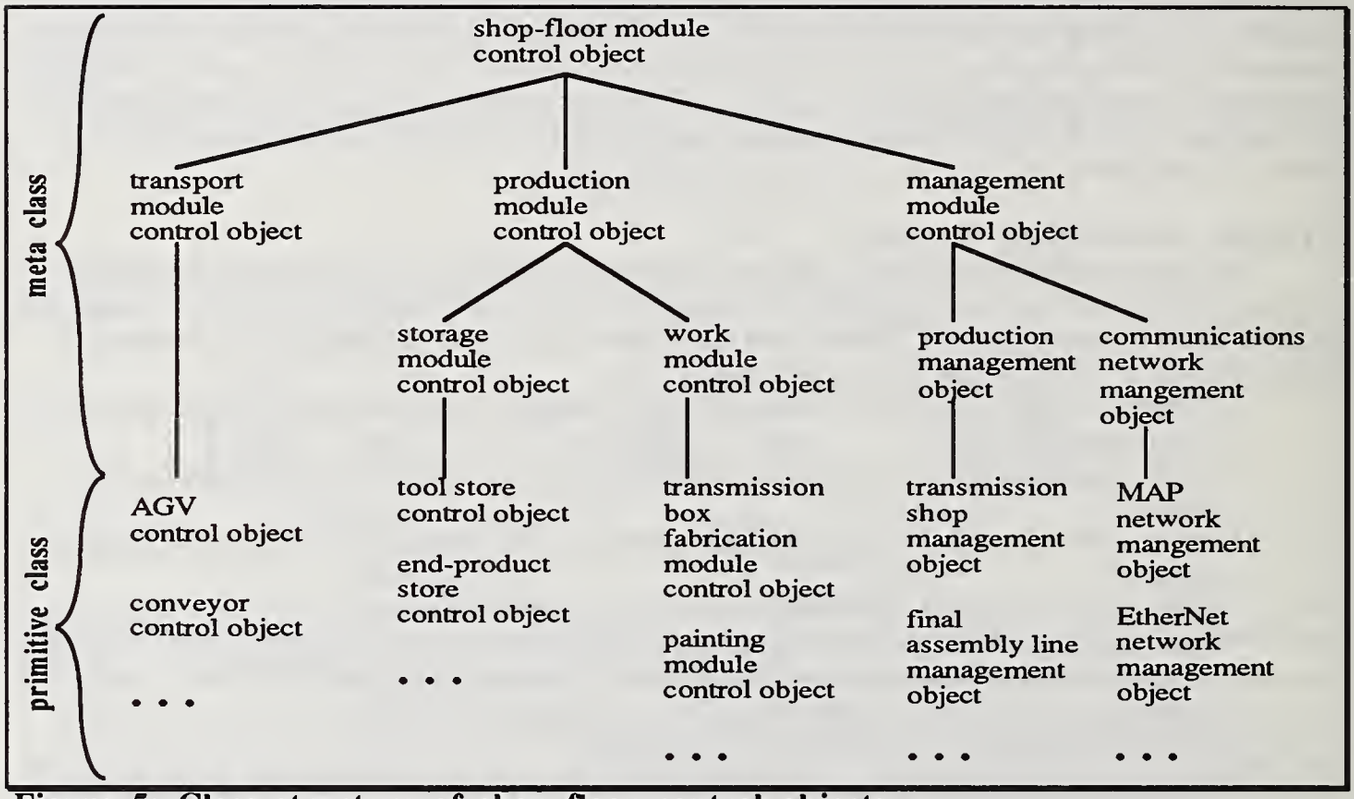


Figure 5: Class structure of shop floor control objects

The shop floor control objects are identified and they are structured in terms of inheritance relationship as shown in Figure 5. Since every shop floor module has states representing its processing status, and its behavior is characterized by the operations it can perform or it requires of other module/device to perform, the module control software is logically identified as an object in the control software model. But there may be too many different modules in discrete manufacturing, it is very cumbersome to list all instances or even the primitive classes of the control objects. Therefore, only the meta classes are fully illustrated.

The first level of control object meta class consists of work, storage, transport, production management, and communication management control objects. The work control object corresponds to the module which fabricate, assembly, disassembly, inspect, or construct product out of raw materials and/or parts. The storage control object represents the control software of a module which stores, retrieves, loads, and unloads supporting hardware or software entities such as parts, tools, programs, and files. The transport control objects is responsible for the control of all types of transportation devices such as AGVs and conveyors. The production management object deals with the dispatching of operators, recording and analyzing the performance data of

equipment and shop floor personnel, and providing aggregated information to higher level manufacturing control objects. Finally, the communications network management control object corresponds to the operating system of a shop floor communications network.

The second level meta classes are production and management module control objects. The production module control object is a generalization of work and storage control objects. The work and storage modules are very similar if one considers the storage module is a type of work module which has a very large local buffer for storing required parts and/or tools and treats the storage, retrieval, loading, and unloading operations equivalent to the work module operations. The work module is restricted to the ability of producing certain type of products while the storage module also has limitation on the types of parts/tools it can store and retrieve. The management module control object is the generalization of production and communications network management control objects. Both management control objects are responsible for allocating resources and recording, analyzing, and reporting resources' performance data. Their main differences come from the different types of resources they managed.

5.2.State variables and operations of control objects

This section describes the data variables/structures and operations to be incorporated in the first level meta-class control objects except the communications network management object. The exception is because that the proposed control model assumes that existing communications networks are used and there is no need to re-design the network management control objects.

For the production module control objects, the following data structure and variables are needed to capture the states of production jobs and module capability. The production control objects can be distinguished into storage or work control objects by different jobs specified in the capacity list, and storage jobs do not involve as frequent equipment setups as work jobs do.

- capacity list: a list of jobs the module can perform, the resources required for performing these jobs, and estimated times needed to perform the jobs if all resources are locally available
- message queue: a sequence of messages received by the module
- open job queue: a sequence of jobs which are announced to be performed but do not have the required resources available on the shop floor yet
- ready job queue: a set of jobs which are ready to be claimed
- bid list: a list of bids, including local bid, for the ready job
- assigned job queue: a set of jobs assigned to the module with all required resources for performing the jobs been shipped to the local buffer
- job claiming permit: a variable indicating if the object can start a local claiming process for a ready job
- module status: a variable indicating if the module is in the state of idle, processing, breakdown, or maintenance.

The transport module control object has the same data structures and variables plus an additional map-list which includes the list of locations the transport module can reach and the distance between any two reachable locations. The production management object also has the same data structures for the service jobs which to be performed by operators but needs no job claiming permit and module status variables. It needs an additional database management system to record and analyze the large amount of performance data.

The operations to be performed by each control object are given as follows:

<u>Work module control object:</u>

- buffer and process the incoming messages
- record and update the information about open jobs
- claim for an open manufacture job with all required parts available on the shop floor
- perform the manufacture jobs assigned to the module
- perform module-maintenance jobs

- retract assigned manufacture jobs in case of manufacturing equipment failure
- request for operator services
- answer queries
- buffer and send outgoing messages

Storage module control object:

- buffer and process the incoming messages
- record and update the information about open storage/retrieval and loading/unloading jobs
- claim for an open storage/retrieval and loading/unloading job with all required parts available on the shop floor
- perform the storage/retrieval and loading/unloading jobs assigned to the module
- solicit for empty pallets and fixtures
- perform the module-maintenance job
- retract assigned storage/retrieval and loading/unloading jobs in case of manufacturing equipment failure
- request for operator services
- answer queries
- buffer and send outgoing messages

Transport module control object:

- buffer and process the incoming messages
- record and update the information about open transport jobs
- claim for an open transport job with all required parts available on the shop floor
- perform the transport jobs assigned to the module
- perform the module-maintenance job
- retract assigned transport jobs in case of manufacturing equipment failure
- request for operator services
- answer queries
- buffer and send outgoing messages

Production management object:

- buffer and process the incoming messages
- record and update the information about open operator-service jobs
- claim for an open job with all required parts available on the shop floor
- perform the operator-service jobs
- record performance data from non-management modules
- perform security measures
- answer queries
- buffer and send outgoing messages

5.3.Interfaces between control objects

The sending of a message is the only way of communicating in an asynchronous system such as the proposed control system. Therefore, the interfaces between control objects is defined by the messages which are passed and received by the objects. The messages act as stimuli to initiate the local operations of receiving objects and have the following general format:

(type, receiver(s), sender, job-info)

where the type of a message can be:

for production module control object: JOB, CLAIM, ENDORSEMENT, WIN, DONE, MAINTENANCE

for transport module control object: T-JOB, T-CLAIM, T-ENDORSEMENT, T-WIN, T-DONE, MAINTENANCE

for production management object: EMERGENCY-REPAIR or REGULAR-SERVICE-REQUEST

The job-info attribute of a message contains a description about the job and has the following format:

(job-id, status, quantity, produced product name, due date, bid for the job, parent job, assigned module, remaining process time until end-product, remaining # of operations until end-product)

where the status of a job can take on one of the values specified as follows:

NEW, AVAILABLE-FOR-CLAIM, A-CLAIM, ENDORSEMENT, CLAIMED, ASSIGNED, DONE, MOVED, QUERY, FILE-TRANSFER

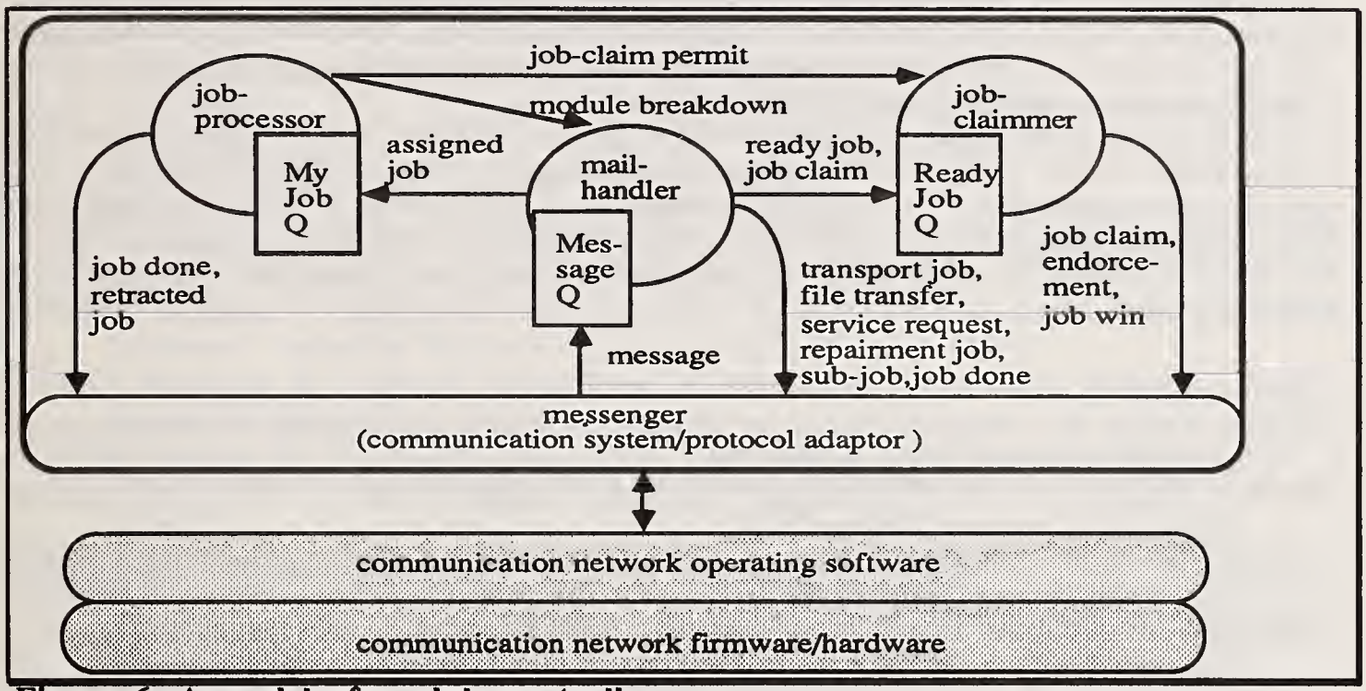


Figure 6. A model of module controller

The rules of using these messages have been informally described in section 4.2. Figure 6 illustrates the messages sent and received by and the job information flows within a production module control object. As also shown in the Figure, the control object design includes a separate communication interface entity, called messenger, so that the majority of control software is independent from the difference of communication protocols; the change of communication network only affect the interface entity.

6.Conclusion

A cooperative control model has been proposed in this paper to address the issues outlined in the Introduction section. The issue about the role of supervisors in cooperative control is addressed in the model by including a production management module which consists of shop floor managers, foremen, and operators. The controller of such a management module is responsible for monitoring, analyzing and reporting the performance data about shop floor personnel as well as equipment. The issue about the efficient storage and exchange of part data is addressed by including software storage modules to store part data as well as other shared shop floor production information. The data or programs stored in the software module are treated as other physical resources and can be requested by and transfered to a module controller immediately before the requiring job is started. The data or programs can be erased after use to vacant the local buffer for future needs. The distributed scheduling issue is resolved by designing a truly distributed scheduling method. The distributed scheduling method can accommodate the failure of manufacturing equipment as well as control computers without interrupting the global job claiming (scheduling) process. The global production optimality issue is tackled by incorporating a pull control method. The pull method allows a global control of work in process inventory levels and

locally optimal allocation of resources. Finally, the control software design issue is dealt with by introducing the Object-Oriented Design approach. The OOD approach can be naturally applied to the design of control software based on the proposed control model. This is because, the control model provides a simple correspondence between control objects and physical modules.

The performance issue of the control model in a large scale FMS, however, is not addressed in this paper. Although the communication delay is expected to be comparatively much smaller than manufacturing delay for a normal production job, it is not clear how many controllers can be connected to a plant-wide LAN without introducing a bottleneck to the production process and how different types of LANs perform with the proposed control protocol. To answer such questions, a simulation study is undergoing to measure the model's performance by explicitly modeling the flows of messages as well as that of physical entities.

A limitation of the proposed control model is that the efficiency of its distributed scheduling method relies on the efficiency of message broadcasting function provided by the underlying communications network. The scheduling method will not be very efficient if the time for broadcasting a message is much longer than single point message passing. Another limitation of this model is its inability of providing global production solutions beyond the work in process inventory control.

7.References

[ABB83] Abbott, R.J. "Program Design by Information English Descriptions," *Communications of ACM*, November 1983, pp.882-894.

[BEN89] Benassi, F., "As Technology Changes, So Does Roles of PLCs," *Managing Automation*, January 1989, pp.28-32.

[BOO83] Booch, G. *Software Engineering with Ada*, The Benjamin/Cummings Publishing Co. Inc., 1983, Second Edition, 1986

[BOO86] ---------. "Object-Oriented Development", *IEEE Transactions on Software Engineering*, February 1986, pp. 211-221.

[COH88] Cohen, Gideon, "Main Components of a Computer-Integrated Manufacturing Reference Model", *CIM Review*, Winter 1988, pp. 28-36.

[COU88] Coulouries, G.F., and J. Dollimore. *Distributed Systems: Concepts and Design*, Addison-Wesley Publishing Company, Inc., 1988, 366 pp.

[DAV83] Davis, R., and R.G. Smith, "Negotiation as a Metaphor for Distributed Problem Solving", *Artificial Intelligence 20*, vol. 1, 1983, pp. 63-109.

[DUF86] Duffie, N.A., and R.S. Piper, "Nonhierarchical Control of Manufacturing Systems," *Journal of Manufacturing Systems*, vol. 5, no. 2, 1986, pp. 137-139.

[EVB86] *Object-Oriented Design Handbook*, EVB Software Engineering, Inc., Rockville MD, 1986.

[GRA88] Gravel, M., and W.L. Price. "Using the Kanban in a job shop environment," *International Journal of Production Researches*, vol. 26, no. 6, 1988, pp. 1105-1118.

[HAR89] Harrison, W.H., J.J. Shilling, and P.F. Sweeney. "Good News, Bad News: Experience Building a Software Development Environment Using the Object-Oriented Paradigm," *OOPSLA'89 Proceedings*, New Orleans, LA, October 1989, pp. 85-94.

[HAT85] Hatvany, J., "Intelligence and Cooperation in Heterarchical Manufacturing Systems," *Robotics and Computer Integrated Manufacturing*, vol. 2, no. 2, 1985, pp.101-104.

[IEE86] IEEE Computer Society. *IEEE Standard 802.1: Overview, Interworking, and Systems Management*, August 1986.

[JON86] Jones, A.T., and C.R. McLean, "A Proposed Hierarchical Control Model for Automated Manufacturing Systems," *Journal of Manufacturing Systems*, vol. 5, no. 1, 1986, pp. 15-25.

[KEO87] Keogh, J.P., "Advanced Production Control Systems," in *ESPRIT CIM*, B. Hirsch and M. Actis-Dato (eds), North-Holland, 1987, pp. 221-234.

[LEL83] LeLann, G. "Motivations, Objectives, and Characterization of Distributed Systems," in *Distributed Systems - Architecture and Implementation*, B.W. Lampson, M. Paul, and H.J. Siegert (eds.), Springer-Verlag, Second Edition, 1983

[LEW82] Lewis, W.C., M.M. Barash, J.J. Solberg, "Single Queue Management of a Job Shop as Implemented by a Data Flow Architecture", *Proceedings of the 23rd International Machine Tool Design and Research Conference*, Manchester, September 1982, pp. 415-420.

[LEW87] -------, "Computer Integrated Manufacturing System Control: A Data Flow Approach", *Journal of Manufacturing Systems*, vol. 6, no. 3, 1987, pp. 177-191.

[MAL88a] Maley, J.G. "Managing the Flow of Intelligent Parts", *Robotics and Computer-Integrated Manufacturing*, vol. 4, no. 3/4, 1988, pp. 525-530.

[MAL88b] Maley, J.G., S. Ruiz-Mier, and J.J. Solberg. "Dynamic Control in Automated Manufacturing: a Knowledge Integrated Approach," *International Journal of Production Researches*, vol. 26, no. 11, 1988, pp. 1739-1748.

[MCG86] McGilem, C.D., J.J. Solberg, J.M.A. Tanchoco, A. Midha, and C.L. Moodie. "Towards an Intelligent Factory Transport System," *ORSA/TIMS meeting*, Miami, FL, October 1986; Purdue University, W. Lafayette, IN 47907.

[MON81] Monden, Y. "Smoothed Production lets Toyota adapt to demand changes and reduce inventory," *Industrial Engineering*, vol. 13, no. 8, 1981, pp. 42-51.

[PRE87] Pressman, R. *Software Engineering: A Practitioner's Approach*, McGrow-Hill Co., Second Edition, 1987.

[RAY88] Raynal, M. "Distributed Algorithms: Their Nature & the Problems Encountered," in *Parallel and Distributed Algorithms,* M. Cosnard et al. (eds), North-Holland, 1988, pp. 179-185.

[SAN89] Santo, Brian, "Industrial Electronics," *IEEE Spectrum*, January 1989, pp. 53-55.

[SHA88] Shaw, M.J., and A. Whinston, "A Distributed Knowledge-Based Approach to Flexible Automation: The Contract-Net FrameWork", *Int'l Journal of Flexible Manufacturing Systems*, 1988.

[STA87] Stallings, W. *Handbook of Computer Communications standard: Local Network*, Macmillan Publishing Co., New York, 1987, 244 pp.

[SUG77] Sugimori, Y., K. Kusunoki, F. Cho, and S. Uchikawa. "Toyota production system and Kandan system materialization of just-in-time and respect-for-human system," *International Journal of Production Researches*, vol. 15, no. 6, 1977, pp. 553-564.

[VAS87] Vasilash, G.S., "Rule-based Breakthrough: for Transfer Line Control," *Production*, May 1987, pp. 34-37.

[WIR89] Wirfs-Brock, R., and B. Wilkerson. "Object-Oriented Design: A Responsibility Approach," *OOPSLA'89 Proceedings*, New Orleans, LA, Octorber 1989, pp. 71-75.

The Importance of Decompositions

in

CIM Control Architectures

by

Wayne J. Davis
Professor of General Engineering
University of Illinois @ Urbana-Champaign
Urbana, Illinois

S. Daniel Thompson
Assistant Professor of General Engineering
University of Illinois @ Urbana-Champaign
Urbana, Illinois

Larry R. White
Assistant Professor of Operations Management
Weatherhead School of Management
Case Western University
Cleveland, Ohio

Abstract

This paper provides a conceptual overview of the role that decomposition approaches can play in defining control hierarchies for computer-integrated manufacturing. This paper focuses upon the production planning problem and defines a multi-level hierarchy to address aggregate, intermediate and detailed production planning. Two basic decomposition approaches are discussed, temporal decomposition and disaggregation. The paper provides a brief survey of the previous applications of these approaches in the literature. It continues to define improved strategies for hierarchical interaction. Considerable attention is focused upon the determination of a suitable planning horizon to be considered in the production planning problem. Finally, the paper sketches the interaction of the production planning hierarchy with other corporate functions, including strategic corporate planning, marketing and purchasing.

Introduction

Today, there has been a renewed interest in manufacturing research, spurred first by increased global competition and a need for improved manufacturing efficiencies, and second, by advances in computing capabilities. To address these needs, manufacturers are increasingly turning to the implementation of computer-integrated manufacturing (CIM). To date, there is little theoretical foundation to guide the overall implementation of a CIM hierarchy. In many cases, the manufacturer is confronted with a variety of vendors, each addressing a particular concern. The hope is that by integrating the individual subsystems an overall efficient hierarchy will evolve. In reality, however, the individual packages are often incapable of communicating with each other as no interfacing standards exists. For example, the manufacturer is incapable of implementing the communication interfaces among the material requirements planning subsystem, cell controller subsystem, and the computer-aided design subsystems. In this sense, the so-called islands of automation develop.

The complexity of the modern large-scale manufacturing system, necessarily limits the scope of the problem to be addressed by a given vendor. The vendor then provides for research and product development in its predefined area of expertise. It is doubtful that any vendor will ever provide an overall solution, and if he did, it is unlikely that any manufacturer would totally commit to a single vendor for a complete solution of its manufacturing problem. Vendors certainly provide one regime for manufacturing research. The other major contributor is academia. Here again, the complexity of the manufacturing problem has forced individual researchers to focus upon a selected subproblem. The emergence of Manufacturing Research Centers has provided an incentive for integration, yet a complete definition of the manufacturing problem facing a given industry is not forthcoming. The conclusion that can be drawn is that for the foreseeable future research in computer-integrated manufacturing will continue to focus upon subproblems.

It should be noted, however, that currently several architectures have been developed for the prototypic implementation. Both *The International Journal of Computer Integrated Manufacturing* and *The IEEE Transactions of Systems, Man and Cybernetics* have recently devoted special issues to the topic. Research monographs such as Williams [13] have also addressed the topic. This material primarily provides an overview of models that have been developed. It does not provide an extensive theoretical basis for design, nor does it provide guidance toward future implementations. Jones et al. [12] provides a more extensive discussion of these issues.

What is needed is a complete definition of the overall manufacturing problem facing one or more industries. The definition of this overall problem would first allow each industry to understand the fundamental properties that distinguish it from other industries. For example, how does the manufacturing of discrete parts differ from the manufacturing problem faced by a large integrated steel

mill? In formulating the overall problem and documenting the differences, one could move toward a more global definition of the manufacturing problem that faces a variety of industries. This approach may not be feasible, however, due to the complexity of the overall manufacturing problem for any industry. Davis and Jones [7] adopted an alternative approach. They realized that the definition of the overall manufacturing problem is an extremely difficult task and began a mathematical definition of the subproblems to be addressed by each function in the CIM hierarchy. In particular, their paper focused upon the definition of the controllers for the individual processes and their interactions with the production scheduler. To provide this interface between process control and production scheduling, they defined an additional hierarchical level termed the process coordinator to supervise the implementation of each assigned processing task for the given manufacturing process. In this manner, the production scheduler, which is defined in greater detail in Davis and Jones [6], coordinates the implementation of the processing tasks at specific processes while the process coordinator provides for the process planning to implement the processing task and assures the processing plan is correctly implemented at a given process.

In performing the above definitions, Davis and Jones [7] have attempted not only to generalize and enhance the defined functionality of each hierarchical entity, they have also focused on interactions among the hierarchical entities. In their development, they have employed the basic principles of mathematical decomposition theory and distributed control. In this approach, for example, they have demonstrated that the production scheduling problem itself can be decomposed into several subproblems representing the scheduling that occurs at several hierarchical levels in a shop floor architecture. Given a typical standard shop floor architecture as defined by the Automated Manufacturing Research Facility at the National Institute of Standards and Technology, production scheduling can occur at factory, shop, cell and station levels. This scheduling must be orchestrated in a manner to permit a considered job to visit the appropriate stations such that all requisite processing tasks will be completed in a manner that both guarantees that the processes will be visited in the proper order and optimizes the overall production flow within the factory. To this end, the scheduling at each entity in the shop floor hierarchical architecture must be coordinated using the principles of mathematical decomposition theory. Davis [8] goes further to demonstrate that the interactions among the hierarchical entities must not only provide for coordination in the solution of the subproblems associated with the decomposition of the overall scheduling problem, but they must also provide controlling input to the subordinate scheduling elements to guide them in the implementation of their assigned production tasks. In this manner, the decision-making and control functions at every hierarchical level are intrinsically linked and cannot be separated.

The validity of the proposed decomposition of the scheduling problem has been substantiated by recent success in the specification of a generic controller which can address both the decision-making and control functions at any hierarchical

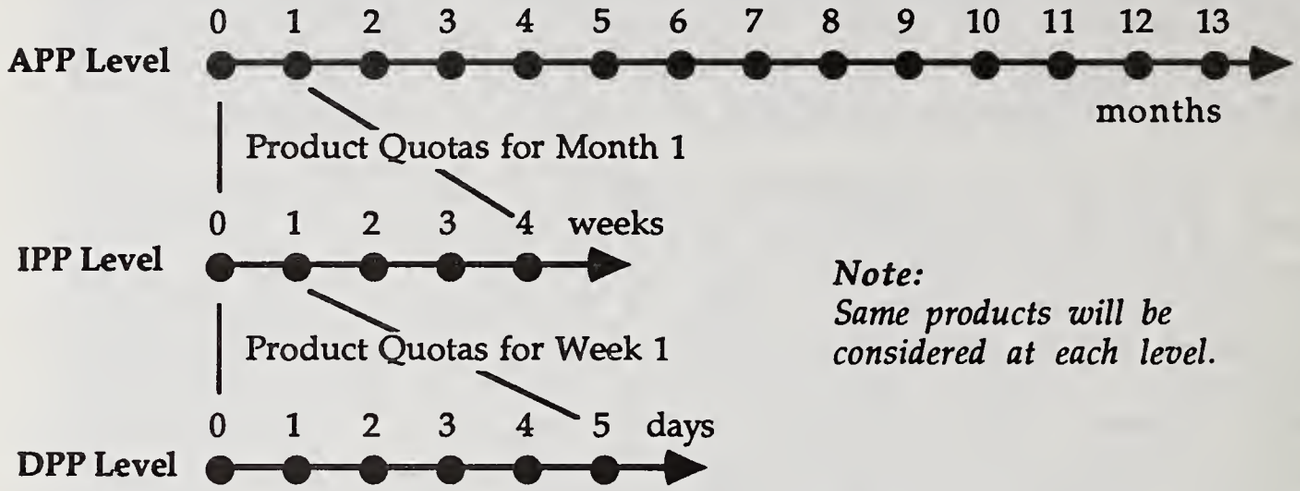
scheduling level. In addition, the defined controller is sufficiently robust to permit its employment both as process coordinator and controller. The specification of this controller will be reported shortly. In addition, the planning functions of the controllers are described in a manner that will allow them to be integrated with the design function to address issues including the design for manufacturability.

The above defined decomposition is spatial in nature as it provides the definition of a class of scheduling and planning problems across a collection of manufacturing entities comprising the factory architecture. The remainder of this paper will concentrate on alternative forms of decomposition, termed temporal decomposition and disaggregation, which decompose the production planning problem over an extended planning horizon into more detailed production planning problems associated with near term production. Specifically, the production planning problem to be addressed will begin with a forecasted customer demand and a collection of issued jobs with assigned due dates which will then be scheduled for production by entities within the factory architecture. The presentation will begin with the definition of the fundamental decision constraints that will be considered at the various hierarchical levels comprising the production planning system. Several existing models will be discussed within the context of the proposed decomposition. Next, we look at the more fundamental assumptions employed in the decomposition, which include the definition of a planning horizon. Finally, we demonstrate how the production planning problem is intrinsically linked to the other manufacturing functions including production scheduling, purchasing, marketing, product distribution, and strategic corporate finance.

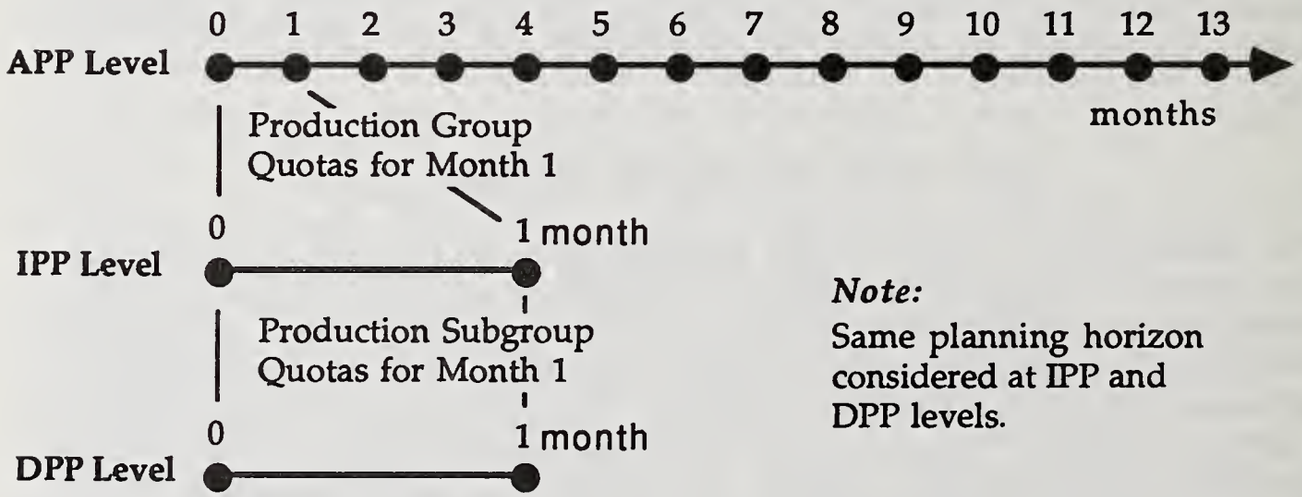
The Production Planning Problem

In defining the schema for the decomposition of the overall production planning problem, the typical first step is to establish an extended planning horizon over which production planning will occur. Let this planning horizon be denoted by T. Typically, a planning horizon of a year or more would be considered to allow for any seasonality, though this assumption will be questioned later. At the highest level of our planning hierarchy, the planning horizon would typically be subdivided into course planning periods of typically one month or more duration. At the intermediate levels, a shorter planning period would be adopted, say one week. Finally, at the detailed production planning level, a short term planning period of one day or less could be considered. This temporal decomposition is demonstrated in Figure 1(a), where the planning horizon of 1 year has been decomposed into 13 planning months, each of which in turn is further decomposed into 4 planning weeks, each of 5 work day duration.

In addition to decomposing the planning horizon into smaller time intervals, the product classifications by each level of the hierarchical production planner are also typically disaggregated as one moves down the hierarchy. A typical product



(a) Typical Temporal Decomposition Scheme



(b) Typical Product Disaggregation Scheme

Figure 1--Decomposition Schemes for Production Planning Hierarchy

disaggregation scheme is pictured in Figure 1(b). At the highest or aggregate production planning (APP) level, usually a gross aggregate representation of product groups are considered. At the intermediate production planning (IPP) level, the aggregate product groups are subdivided into product subgroups which in turn will be subdivided into individual products at the detailed production planning (DPP) level. As illustrated in Figure 1(b), the planning horizon at the IPP and the DPP levels are typically the same under a product disaggregation schema.

Under the product disaggregation schema, the APP problem begins by assuming that there exists a set of demand forecasts for product group j in the planning period t or $\{D_j^t \mid j=1,...,J;\ t=1,...,T\}$. Given this forecasted demand scenario, the goal of the APP problem is to establish the production quotas, X_j^t, for each product group j in each planning period t. It is the nature of the production planning problem that the production quotas generally will not be exactly equal to forecasted demands. To this end, we will allow flexibility through the introduction of both inventory for product group j in period t, I_j^t, and backorder, B_j^t, in the material balance constraints given as

$$I_j^{t-1} + X_j^t = S_j^t + I_j^t \quad \text{for } j=1,...,J;\ t=1,...,T \tag{1}$$

$$S_j^t + B_j^t = D_j^t + B_j^{t-1} \quad \text{for } j=1,...,J;\ t=1,...,T \tag{2}$$

where S_j^t represents the projected sales of product group j in period t. In this formulation, we will presume that there are limitations to the inventory that can be maintained in each period t given as by the functional constraints

$$I_j^t \le I_j^{max} \quad \text{for } t=1,...,T \text{ and } j=1,...,J \tag{3}$$

Here we are assuming that the primary coupling of the inventory between periods is through the material balance constraint (1). We will also assume that limitations exists in the production capacity which constrain the production quotas X_j^t through the production constraints

$$\sum_{j=1}^{J} a_{jm} X_j^t \le C_m^t \quad \text{for } m=1,...,M \text{ and } t=1,...,T \tag{4}$$

In constraint (4), we are assuming that there are M distinct production capacities to be considered, denoted by C_m^t (m=1,...,M). The coefficient a_{jm} specifies the amount of capacity m that the production of one unit of X_j^t will consume. As noted, the capacity C_m^t is assumed to vary with time as indicated by the superscript t. This capacity can be modified by several other production inputs. The first input to be considered is the level of personnel in period t, denoted by the variable P^t. P^t is assumed to be governed by the personnel balance constraint

$$P^{t-1} + H^t - F^t = P^t \quad \text{for } t=1,...,T \tag{5}$$

where H^t is the number of personnel hired in period t and F^t is the number fired. Constraint (5) is sometimes included within the APP problem. However, in other instances, P^t may be determined by other manufacturing functions such as human resource development. This is especially true today when manufacturing companies are faced with an increased need to provide specialized training. The second production input is the capitalization level in period t. In most

formulations of the APP problem, capitalization has been ignored. Nevertheless, it is an important consideration as it not only provides for the purchase of material inputs, but it also can reflect the purchase of additional production capacity. The capitalization is often excluded from direct consideration in the APP level since its determination is not made by the APP, but rather by corporate finance in fulfilling their strategic planning function. This fact again accentuates the previously stated concerns pertaining to the definition of individual functions without developing an adequate interface to the other functions. It must be recognized that there are indeed exogenous decisions which provide direct impact upon the APP problem, and ultimately the interfaces with these exogenous functions must be specified.

To define the aggregate production planning problem the next step is to specify the objective function to be employed in the optimization. Typically, one would attempt to maximize the total profit that would emerge from the assignment of values for X_j^t.

$$P = \sum_{t=1}^{T} \left[\sum_{j=1}^{J} \left(s_j^t S_j^t - x_j^t X_j^t - i_j^t I_j^t - b_j^t B_j^t \right) - p^t P^t - h^t H^t - f^t F^t \right] (1+i)^{-t} \qquad (6)$$

where

s_j^t is the selling price per unit of product group j in period t
x_j^t is the direct production costs per unit of product group j in period t
i_j^t is the inventory holding cost per unit of product group j in period t
b_j^t is the backordering penalty cost arising per unit of product group j in period t
p^t is the individual labor cost in period t
h^t is the hiring cost for each person hired in period t
f^t is the firing cost for each person fired in period t
$(1+i)^{-t}$ is the discounting factor used to compute the present worth of the cash flow in period t given the interest rate i.

The objective function as defined above could be revised to consider additional costing detail. For example, we may decide to delineate between production costs associated with regular versus overtime production. We may further include additional inventory costs which allow for shrinkage and account for acquiring additional storage space when a given level of inventory is exceeded. For our discussion purposes here, a simplified formulation will be adequate as we desire to focus upon the decomposition of the overall production planning problem into a collection of subproblems.

The above APP problem is now nearly specified. However, one important set of constraints remains, namely the boundary conditions at the beginning and the end of planning horizon, periods 0 and T, respectively. To this end, we must at a minimum specify the values for I_j^t (j=1,...,J) and P^t at periods t = 0 and T. With respect to the initial conditions, we will assume that they are known and can be specified. Therefore, we will include the additional set of boundary condition constraints

$$\{I_1^0,\ldots,I_J^0;P^0\} \in \text{I.C.}(0) \qquad (7)$$

On the other hand, the final boundary conditions are not known with certainty, but nevertheless they must be specified as

$$\{I_1^T,\ldots,I_J^T;P^T\} \in \text{F.C.}(T) \qquad (8)$$

It is at this juncture that a difficulty arises. The production planning problem is being constrained by a collection of boundary conditions at the end of period T which we cannot precisely specify. Moreover, the specification equation (8) may influence the values assigned to all the decision variables, including the production quotas for the current planning period (t=1), X_j^1 for j=1,...,J. If these production quotas are not influenced by equation (8), then we may question the utility of including planning period T in the considered planning horizon.

The difficultly of specifying the boundary conditions is but one level of imprecision that exists in the production planning problem. We will return to the discussion of the planning horizon later. Other forms of imprecision also arise from the aggregation of the individual product type into product groups. In developing this aggregation, the individual products are assigned to a given product group based upon their similarities. Nevertheless, no specific product within the group will likely have the exact production requirements as specified for the product group in constraint (4) nor will it have the same selling price s_j^t as included in the objective function (6). Instead, the values of constants included in equations (4) and (6) are likely to represent weighted average values arising from the consideration of all products within the product group.

Another form of imprecision arises from the forecasted demand stream for the aggregate product groups. That is, the values for D_j^t used in the formulation are themselves random variables. Given the levels of imprecision, the optimality of any given aggregate production plan is impossible to demonstrate. Furthermore, it appears to be impossible to reconcile the imprecision. We could abandon the use of aggregate product groups by considering production quotas for the individual products over the planning horizon. However, one would then be faced with forecasting the demand for the individual products over the planning horizon, which is typically much more difficult and less precise than forecasting the demand for the aggregate product groups. Thus, the imprecision in demand forecasting can never be eliminated. In addition, the number of decision variables as well as the number of constraints will be significantly increased by considering the individual products, making the resulting problem much more difficult to solve. For a large corporation, the size of the resulting problem would likely be prohibitive.

To overcome the computational complexities, both a temporal decomposition scheme to break down the extended planning horizon into smaller subintervals and a product disaggregation scheme to break down the aggregate product group into individual products is required. Unfortunately, to date, little work has addressed the simultaneous consideration of both requirements. In this presentation, we will attempt to implement both.

As defined in aggregate terms, there is really no need to decompose the mathematical programming problem as specified by equations and constraints (1) through (8). However, if we want to consider additional detail in the production planning in one or more periods, then the inclusion of the detailed constraints as described above, can lead quickly to an intractable program. As discussed earlier and depicted in Figure 1, there are two basic approaches toward the definition of a production planning hierarchy. Product disaggregation schemes have been adopted by several authors, including Bitran et al. [3,4,5] and Axsater et al. [1,2] to permit refinement of production quotas for product groups into individual products over a fixed planning period at all hierarchical planning levels. Other authors such as Gershwin [9,10,11] have employed a temporal decomposition to investigate the production requirements of specific products over successively shorter planning periods in their formulation of the planning hierarchy. Neither approach in itself represents a complete solution. Specifically, although the product disaggregation schemes do ultimately specify the detailed production requirements upon a shortened planning horizon, we are still faced with the further decomposition of the remaining planning horizon into smaller periods until eventually specific jobs with associated due dates can be issued for production scheduling. On the other hand, temporal decomposition schemes, which consider individual products at each decision level, also suffer from the fact that there is typically not sufficient data to support the planning for the individual products over an extended planning horizon which the APP typically must address. As noted above, even if detailed production planning problems could be formulated over the extended horizon, the resulting problem would likely be impossible to solve.

The decomposition scheme to be described here contains elements of both a product disaggregation and a temporal decomposition. With respect to the disaggregation concept, as we consider the lower hierarchical elements within the hierarchical production planning problem, we will address the production planning with more refined product groups until we ultimately specify the manufacturing of a specific product for a specific customer with an associated due date when the order must be complete. The temporal decomposition arises from the fact, that each sublevel of the production planning hierarchy considers successively smaller planning periods over a shorter planning horizon. For example, as described above, the APP problem could consider a planning horizon of T months. The IPP level, could subsequently consider a planning horizon of 4 work weeks and provide more detailed production quotas for each of the weeks. The DPP level would then employ a planning horizon of one week and perform DPP for each of the work days comprising the considered work week.

Theoretically, at the IPP level, production planning could be performed for every month considered at the aggregate production planning level. In practice, however, this approach is seldom adopted. Instead, most authors have adopted a model in which production planning is performed for the first month of the planning horizon only. Therefore, the APP solves its problem to establish values

for all decision variables in each of the T months which it considers. It then provides the values of these decision variables for the first month $\{X_j^1, S_j^1, I_j^1, B_j^1$ and $P^1 \mid j=1,...,J\}$ to the intermediate planning level. Assuming that each aggregate product group j is comprised of intermediate product groups k for $k=1,...,K_j$ the IPP would then establish production quotas, sales and associated inventory levels associated with weeks w of month 1 or

$\left\{X_{jk}^{1,w}, S_{jk}^{1,w}, I_{jk}^{1,w}$ and $B_{jk}^{1,w} \mid j=1,...,J;\ k=1,...,K_j$ and $w=1,...,4\right\}$ such that

$$X_j^1 = \sum_{k=1}^{K_j} \sum_{w=1}^{4} X_{jk}^{1,w} \tag{9}$$

$$S_j^1 = \sum_{k=1}^{K_j} \sum_{w=1}^{4} S_{jk}^{1,w} \tag{10}$$

$$I_j^1 = \sum_{k=1}^{K_j} \sum_{w=1}^{4} I_{jk}^{1,w} \tag{11}$$

$$B_j^1 = \sum_{k=1}^{K_j} \sum_{w=1}^{4} B_{jk}^{1,w} \tag{12}$$

In establishing the values for its decision variables, the IPP will use the same basic constraints sets as the APP. However, they will be formulated with additional detailed information to permit the proper specification of the associated decision variables within a given product subgroup. In particular, the demand stream will more likely employ booked orders rather than forecasted demand due to the shortness of the planning horizon. The production constraints, corresponding to constraint (4) at the APP level, will also provide the additional production details associated with the more specific intermediate product subgroups.

Additional constraints will also be introduced, including material requirements planning. Specifically, the intermediate production quotas will be established to insure that the necessary material inputs are available. In this manner, the IPP problem is intrinsically tied to the purchasing function for the CIM hierarchy. Given the global manufacturing environment that exists today, this could require that intermediate production quotas be established several weeks in advance. For example, an engine manufacturer may employ components that are supplied by a South American vendor. An electronics manufacturer may use solid state components from a Far East supplier. The physical distance required in transporting these goods requires that IPP be performed over an extended horizon. It is this fact that renders many of the existing decomposition schemes appearing in the literature intractable based upon their currently selected planning horizons. In addition, the APP considers only an aggregate approximation of the true production

planning problem. There may be additional benefits that can be gained by considering a more detailed optimization over intermediate horizon. For example, one month may be too short of a planning horizon to develop an understanding of the variations that arise in throughput due to fluctuations in product demand. To this end, rather providing a detailed development of the production planning problems at the lower DPP level of the planning hierarchy, we will focus upon the assumptions being implied by existing decompositions and the potential for enhancing the hierarchical interactions among the APP and the IPP levels. The similarity of the interaction of the IPP with the DPP will easily permit an expansion of this discussion in the future.

Current Hierarchical Schema

In Figure 2, the minimal practical schema for hierarchical interaction has been presented which is derived from existing models for either product disaggregation or temporal decomposition. As depicted, typically the APP solves its problem for the next T months and passes the values for the decision variables associated with the first month to the intermediate level. At this point, the APP is idled until the first month's aggregate production quotas are realized. Meanwhile, the IPP attempts to meet the specification of the first month's production quotas by treating them as goals as indicated by equations (9) through (12). Existing models for the most part do not provide for any feedback during the first month from the IPP to the APP until the end of the month, as the IPP is expected to satisfy equations (9) through (12) as planned. It is only at the end of the month, that the IPP's success in meeting the production goals is known and returned to the APP as $\{X_j^1, S_j^1, I_j^1, B_j^1$ and $P^1 \mid j=1,...,J\}$ (note a bold character is indicating feedback information). It is important to recall that the aggregate production quotas were developed using only approximate constraints and that the manufacturing processes are themselves stochastic entities. Further, as discussed in the previous paragraph, the manufacturing systems response is also tied to that of the external vendors whose delivery dates also represent stochastic entities. Therefore, it is unlikely that the aggregate production quotas will ever be satisfied exactly as planned.

Therefore, during the first week of the first month of the planning horizon considered by the IPP, there is considerable uncertainty in the manner in which the production quotas for that month will be satisfied. As the IPP moves further into the first month planning horizon, say the third or fourth week, the IPP can make much better estimations upon the success in satisfying these goals. As noted above, the APP typically does not see any of these estimates until the entire one month planning horizon for the IPP has been completed. Then using the concept of a rolling planning horizon, the APP employs the realized production quotas for the first month and resolves the APP problem to obtain production quotas for months 2 through T+1, assuming that a constant planning horizon of T months is being employed at the APP level. Upon solution of its new problem, the production

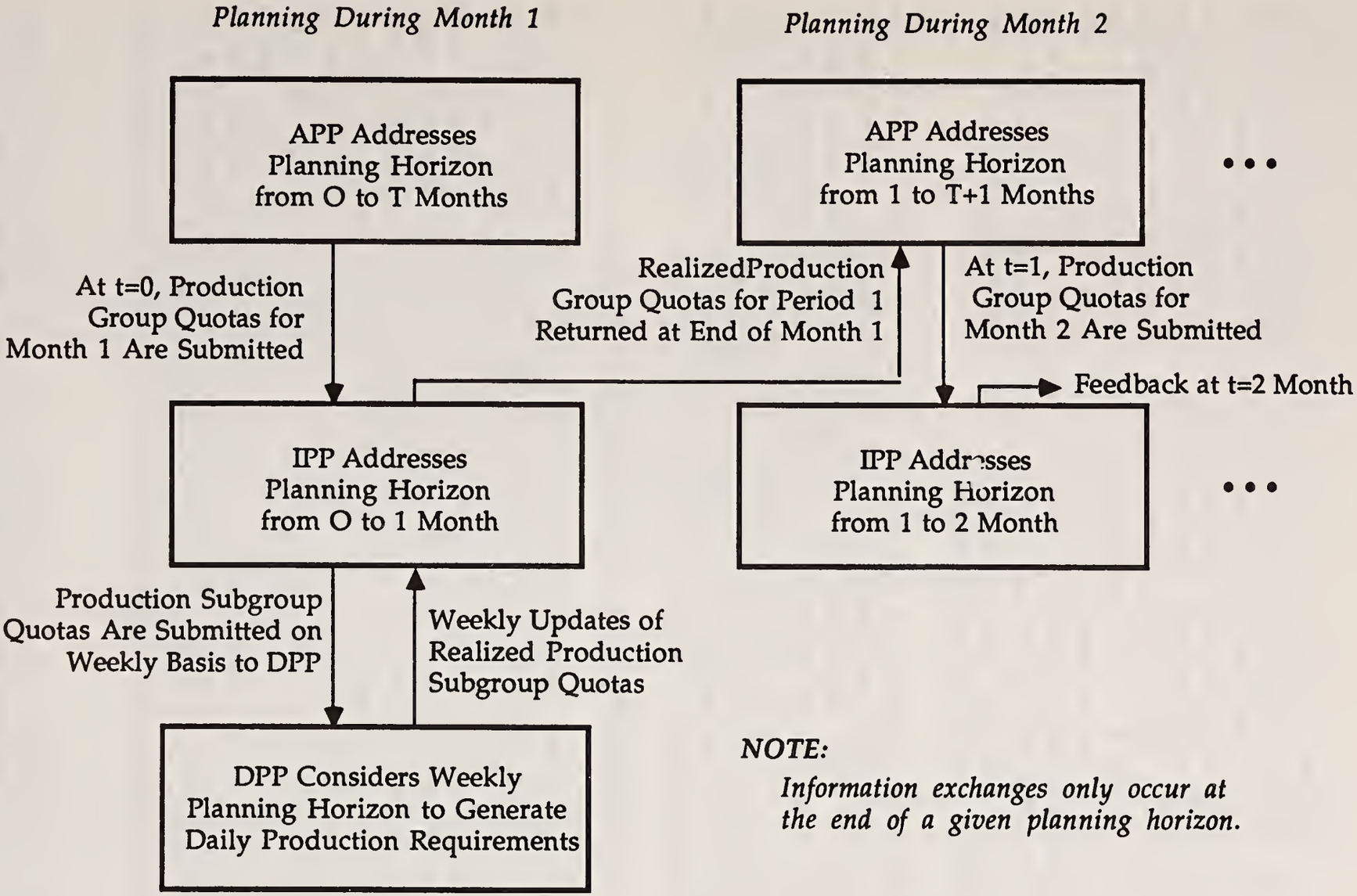


Figure 2--Minimal Hierarchical Interaction Scheme

quotas for month 2 are again passed to the IPP for implementation and the process is repeated.

Under the minimal interaction schema, the IPP's interaction with the DPP is nearly identical. That is, the IPP passes the first week's subgroup quotas to the DPP and waits for the DPP to respond with the implementation of its requests at the end of the first week. The IPP then solves its problem for weeks 2 through 4 and passes the second week's quotas to the DPP. The IPP then again waits for the DPP feedback on the successful completion of the second week's quotas before resolving the quotas for weeks 3 and 4. After the third week, the IPP passes the fourth weeks quotas to the DPP and upon completion of the fourth week responds to the APP with the realized production quotas for the first month. The APP then responds with the production quotas for the second month where upon the IPP again solves it problem for the new four week planning horizon and submits the first week's subgroup quotas to the DPP for implementation.

The reader will notice that the IPP has a variable planning horizon as currently formulated, with the planning horizon varying from one to four weeks. Given that the IPP must coordinate his decision-making with the purchasing to secure material inputs for the planned production, the one week planning horizon at week four in the planning cycle is somewhat troubling. Even more troubling is the fact that the IPP, and also purchasing, never see the next month's production quotas until the current month is complete. Therefore, there is a complete discontinuity of planning at the end of each month at the IPP level.

A New Hierarchical Coordination Schema

The above proposed hierarchical interaction is perhaps the simplest that can be proposed, and as noted above, has been adopted to provide for minimal coordination among the hierarchical planning elements. Unfortunately, it does not reflect the true operations of the planning hierarchy where considerably more interaction among the hierarchical levels will likely exist. To this end, more elaborate coordination schemes are under development. One proposed scheme begins with the same decomposition of time into smaller planning periods at each sublevel of the planning hierarchy (see Figure 3). Again the APP begins by considering the planning horizon over the next T months. However, the APP would consider the planning for periods t = 2,...,T+1 even as the production quotas for the current period 1 are being implemented by the IPP. Instead of performing intermediate planning for period 1 only, the IPP considers aggregate production for both periods 1 and 2. Therefore, to establish boundary conditions for the IPP, the APP must specify group quotas for both periods 1 and 2[1] . Upon releasing the quotas

[1] Limiting the passing of quotas for periods 1 and 2 has been made to facilitate discussion only. In implementing the proposed coordination schema, quotas for any number of periods can be passed from the APP to the IPP for use in its planning.

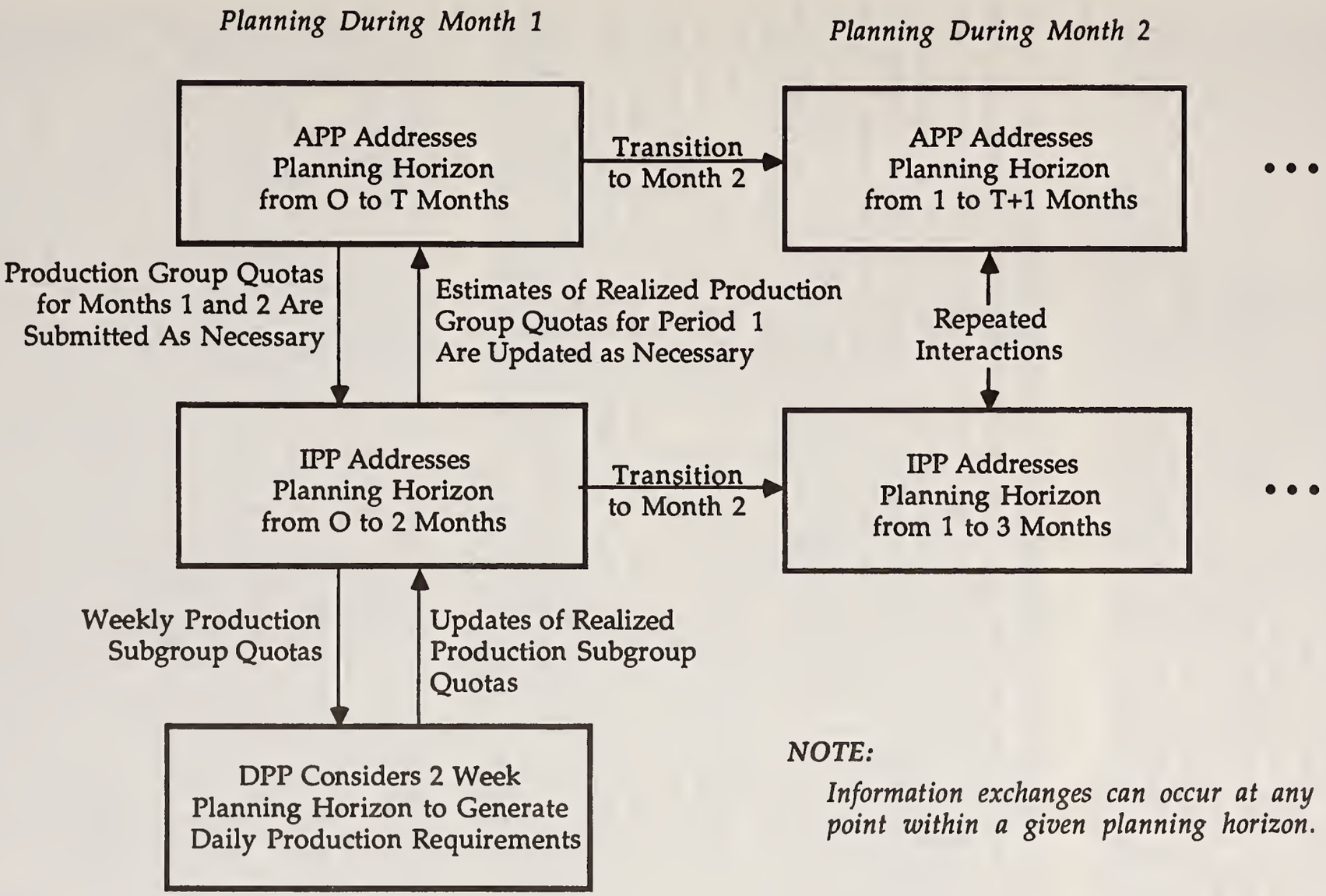


Figure 3--Improved Hierarchical Interaction Scheme

for period 1 to the IPP, the refined planning for period 1 is solely in the hands of the IPP. The quotas for the end of period 2, however, are still flexible and will be considered by both the APP and the IPP during the first month. In its intermediate planning, the IPP will continuously update the manner in which it expects to meet the current period's quotas for period 1. This in turn affects the expected initial conditions used by the APP in solving its problem in period 2. As stated above, during the first weeks of the IPP's one month planning horizon there are considerable uncertainties in the eventual realization of the first month's production quotas. However, in the later weeks of the first month, much better estimates can be made.

The proposed coordination between the two levels of decision-making is as follows. With each update of the expected outcome of this month's planning–that is, the expected values that will be realized for $\{X_j^1, S_j^1, I_j^1$ and $B_j^1 \mid j=1,...,J\}$ by the IPP–the newly updated values are returned to the APP. The APP then resolves its problem for periods t=2,...,T+1 to obtain a revised set of goals for the IPP at period 2 or $\{X_j^2, S_j^2, I_j^2$ and $B_j^2 \mid j=1,...,J\}$. The IPP then resolves its problem for periods 1 and 2 using the most updated measurements of system performance to update its decision. The updating procedure could also be triggered by the APP. Suppose that a new customer demand is realized which forces a change in the second month's production quotas $\{X_j^2, S_j^2, I_j^2$ and $B_j^2 \mid j=1,...,J\}$. The APP would then pass the updated values to the IPP, which would then resolve its problem to provide any updates to this period's quotas $\{X_j^1, S_j^1, I_j^1$ and $B_j^1 \mid j=1,...,J\}$. These updated values could then be returned to the APP who may again update $\{X_j^2, S_j^2, I_j^2$ and $B_j^2 \mid j=1,...,J\}$. The updated values would then be returned to the IPP for its action. We note here that only IPP can change the values for the current period's outcome $\{X_j^1, S_j^1, I_j^1$ and $B_j^1 \mid j=1,...,J\}$, and only the APP can change the next period's goals, $\{X_j^2, S_j^2, I_j^2$ and $B_j^2 \mid j=1,...,J\}$. This interaction is repeated throughout the one month planning period whenever updates are necessary.

The described interaction provides for a nearly continuous interaction between the APP and the IPP. As the IPP approaches the end of the first planning period, the outcome of this month's production is nearly known with certainty. Therefore, the production quotas for the next month will have stabilized. At the end of the first month, the IPP will already know the quotas that it will receive for the next month as it has been considering them in its planning throughout the current month. When the next month production quotas are released to the IPP for implementation, the APP will also submit tentative production quotas for month 3 which it and the IPP will refine during the month 2. During the second month, the planning horizon for the APP will include months 3 through T+2. In addition to providing an overlap in planning periods between the APP and the IPP, the proposed interaction scheme provides an extended horizon for the IPP to consider. Under the minimal interaction schema, during the fourth week, the IPP would only have a single week planning horizon to consider. As noted earlier, material requirements planning is typically implemented at the IPP level. The single week

planning period during the fourth week is likely to be inadequate to provide for a proper interaction with purchasing to secure input materials from outside vendors. Further, under the original configuration neither the IPP nor purchasing would have information concerning the next month's quotas. Under the proposed interaction, tentative production quotas for the next month would be known from the outset. At a minimum, the IPP would consider a planning horizon of five weeks which is more likely to provide for adequate lead time to interact with purchasing to fulfill the task associated with material requirements planning. If this period is still inadequate, then the APP could submit tentative production quotas for two or more months in advance.

A similar decomposition scheme could be developed between the IPP and the DPP. Specifically, the IPP could issue subgroup production quotas for the current week to the DPP for implementation and tentative quotas for the next week to facilitate planning. The interaction scheme would allow the DPP to update subgroup production quotas for the current week as the week evolves while the IPP would update the tentative quotas for the next week. In this manner, the DPP would never consider a planning horizon of less than one week in its determination of which jobs and their associated due dates are to be issued to the production scheduler for manufacturing.

Additional Production Planning Considerations

The Planning Horizon

In our previous discussion, we noted that the specification of a planning horizon for the APP required us to specify boundary conditions for inventories and personnel levels at the end of period T [see equation (8)]. We also noted that, in general, the specification of these boundary conditions influenced the optimal assignment of values to the decision variables, including the production goals which the IPP considers, namely $\{X_j^1, S_j^1, I_j^1$ and $B_j^1 \mid j=1,...,J\}$ and $\{X_j^2, S_j^2, I_j^2$ and $B_j^2 \mid j=1,...,J\}$. If these boundary conditions do not influence the optimal assignment of goals, then we might question the consideration of month T in the planning horizon. In fact, if this is the case, we could argue that planning in period T is decoupled from the planning for the current period. Theoretically, the planning in any given period is never really decoupled from the specification of the boundary conditions during a later period. That is, using the principles of sensitivity analysis in mathematical programming, we can investigate the behavior in the production quotas for a given period as a function of the specified boundary conditions. If we can demonstrate that the consideration of boundary conditions within a meaningful range of ending inventories and backorders produces no changes in the values of $\{X_j^1, S_j^1, I_j^1$ and $B_j^1 \mid j=1,...,J\}$ and $\{X_j^2, S_j^2, I_j^2$ and $B_j^2 \mid j=1,...,J\}$, then we might conclude that planning in period T is decoupled from the planning in periods 1 and 2. Here again, however, we must qualify our comments. Specifically, there are indeed other factors pertaining to factors in the later periods that can influence the development of production quotas for periods 1 and 2. For example, if we modify the production

capacity constraints, constraint (4), in the later periods, this modification may also lead to changes in the production quotas for periods 1 and 2. As we noted above, the production capacities C_m^t for m=1,...,M are indeed functions of other variables including the personnel levels, P^t, and the capitalization levels in period t. These latter variables are also not likely to be under the control of the APP, but rather they are established by other manufacturing functions. Therefore, it appears to be prudent that variations in the production capacities, C_m^t (m=1,...M and t=1,...,T), shoulld also be considered in determining a planning horizon.

The issue then arises to what minimum period can the planning horizon be reduced. This is indeed a difficult issue to address. A major consideration is the level of certainty with which we can specify the APP's problem. Specifically, if we know the boundary conditions at any period, the production capacities at all intervening period and the associated product demands, then the planning horizon can be set to the period where the boundary conditions are known. Given the level of uncertainties associated with the APP problem, such a scenario is not likely to occur. There are other special cases that also might be considered. In the situation, where we are faced with excess production capacity for the foreseeable future, then the production plan would likely produce to meet each period's demand, with little inventory being carried from period to period. In this case, the planning horizon is typically very short, say one or two periods. At the other extreme, if backorders are completely retained and product demand eclipses our production capacity for the foreseeable future, then the planning horizon would again be very short as we would simply attempt to fulfill the most profitable demand each period. Unfortunately, neither of these two latter scenarios represents an ideal corporate situation. In the first case, corporate planning should be considering the reduction of production capacity while the latter situation would likely spur an expansion of production capacity. Neither of these decisions can be addressed by the APP as defined. Indeed, other corporate functions would be involved.

It is difficult to establish an APP planning horizon without the coordinated interaction with other manufacturing functions. The APP may employ sensitivity analysis to seek the influences of boundary conditions and production capacities upon the proposed production quotas given the forecasted product demand, but in so doing, the APP is actually attempting to estimate the consequences that will be derived from future interactions with the other manufacturing functions. To this end, we must recognize that the overall production planning function, as defined here, is actually a subproblem within a much larger corporate planning effort. To this end, we will now focus upon these additional interactions.

Interaction with Additional Corporate Functions

In the past, a constant planning horizon has been justified to address seasonality issues. The presumption was that the manufacturing system was likely to experience a repeatable demand cycle. The length of this demand cycle would then determine the planning horizon. Assuming this repeatable demand, then one

could argue that the initial conditions for this period would likely represent the ideal boundary conditions for the end of the demand (or planning) cycle. Perhaps this assumption was appropriate when product lifecycles were several years in duration. However, with today's short product lifecycles, the repeatability of a product demand cycle is unlikely. To this end, APP must be more aggressive in seeking the proper planning horizon and boundary conditions. This decision, however, cannot be made independently of the other manufacturing functions including marketing and strategic corporate planning. That is, given the present products to be marketed, it is marketing that generates the demand stream which the APP attempts to satisfy. Thus, the interaction between marketing and the APP is obvious. Specifically, given the current forecasted demand stream from marketing, the APP can project the manner in which it will optimally satisfy that demand. Given this information the marketing function may modify its marketing strategy. For example, if there is excess production capacity, marketing may consider lowering the price for certain products to generate additional demand. If there is excess demand, the price for selected products might be raised to increase the profitability for the demand that can be satisfied. If cyclic demand periods do exist, marketing might also adopt strategies to shift demand to periods with excess production capacity. The consequences of any of these actions upon the APP function are obvious.

Similarly, corporate strategic planning also influences the APP problem. In particular, we noted in equation (4) that production capacities are limited based upon the current capitalization levels. Strategic planning can obviously increase the production capacity to satisfy unfulfilled demand by providing additional capital. The APP function obviously provides inputs to where such capital should be applied. Alternatively, strategic planning may also reduce capacity for the production of products which are no longer profitable. Strategic planning also controls when new products will be introduced for production and when production will cease for existing products. Not only do these decisions influence which products the APP will consider and the associated manufacturing capacities to fulfill forecasted demand, but they also influence the marketing function in its sales effort. Today, a year is an extended period for the consideration of a planning horizon for the APP. During this period, production capacities as well as the products to be produced are likely to be changed. It is unlikely that any production cycle will repeat. Rather, it is the combined efforts of strategic planning, marketing and production planning that must define the planning horizon and the boundary conditions that should be satisfied. That is, the production planning problem itself is a subproblem of a larger corporate planning problem that must be considered. Therefore, it is essential that our focus upon the APP problem also addresses the larger corporate planning problem to provide an adequate interface into the other corporate functions with which it must interact.

CONCLUSION

This paper has provided a broad overview of the production planning problem. The focus of the discussion, however, was directed toward the role that decomposition approaches will play in defining a CIM hierarchy rather than the discussion of the production planning problem per se. The production problem itself represents an ideal example to demonstrate the need for decomposition. It requires us to consider a temporal decomposition of an extended planning horizon into smaller planning periods which eventually will lead to the specification of specific products to be scheduled for production with associated due dates. The production planning problem also allows us to discuss another approach to hierarchical decision-making involving the principles of disaggregation among the decision variables at the various hierarchical levels. Both of these approaches have already been separately applied in the literature This paper shows how both approaches can be simultaneously applied to address the production planning problem. The reader is referred to Davis and Jones [6] for yet another approach to hierarchical decomposition, namely spatial decomposition as it is applied to the production scheduling function within a CIM hierarchy. Further, Davis and Jones [7] provides a more fundamental (and mathematical) discussion of how the individual functions interact to define a CIM hierarchy.

This paper has, however, extended the functional discussion in Davis and Jones [7] to consider additional functions. In discussing the interaction of the APP with the IPP, we developed a hierarchical coordination scheme which provided for an continuous update of the decision-making for both problems. In addition, we provided for a more extended planning horizon at the IPP level to permit it to effectively interact with the corporate purchasing function to implement the tasks of materials requirement planning. This extended planning horizon at the IPP level also provided for a continuity of production quotas as the IPP moves from one planning period to the next.

We next focused the discussion upon determination of the planning horizon for the APP. We briefly highlighted the role that the boundary conditions and the production capacities play in establishing the planning horizon. To this end, we expanded our discussion to sketch the interaction of the APP with the other corporate functions including the strategic planning and marketing. It was shown that in today's manufacturing environment with its short product lifecycles, these three functions must act in unison to establish both the planning horizon and the boundary conditions that the production planning problem will address.

The scope of this paper has been conceptual in basis. Theoretical and mathematical proofs do exist for many of the concepts introduced here. However, this paper has elected to provide a larger conceptual overview of the interaction of the CIM functions rather than providing detailed mathematical discussions. Future papers and research will address the mathematical issues. This paper has attempted to demonstrate the complexity of the mathematical decision-making and control

problem being addressed by the CIM hierarchy, the role that the principles of decompositions can provide in addressing the overall problem, and finally the need to establish improved coordination mechanisms to provide for the interactions of the individual manufacturing functions.

References

[1] Axsater, S., "On the Design of the Aggregate Model in a Hierarchical Production Planning System," **Engineering and Process Economics**, Vol. 4, 1979, pp. 84-97.

[2] Axsater, S. and Jonsson, H., "Aggregation and Disaggregation in Hierarchical Production Planning," **European Journal of Operations Research**, Vol. 17, No. 3, 1984, pp. 338-350.

[3] Bitran, G. R. and Hax, A., "On the Design of Hierarchical Production Planning Systems," **Decision Sciences**, Vol. 8, 1977, pp. 28-55.

[4] Bitran, G., Haas, E. and Hax, A., "Hierarchical Production Planning: a single stage system," **Operations Research**, Vol. 29, 1981, pp. 717-743.

[5] Bitran, G., Haas, E. and Hax, A., "Hierarchical Production Planning: a two stage system," **Operations Research**, Vol. 30, 1982, pp. 232-251.

[6] Davis, W. and Jones, A., "A Real-time Production Scheduler for a Stochastic Manufacturing Environment," **International Journal of Computer Integrated Manufacturing**, Vol. 1, No. 2, 101-112, 1988.

[7] Davis, W.J. and Jones, A.T., "A Functional Approach to Designing Hierarchies for CIM," **IEEE Trans. on Systems, Man, and Cybernetics**, Vol. 19, No. 2, 1989, pp. 164-174.

[8] Davis, W. J., "Evolving Coordination Schemes in Real-Time Production Scheduling," **Engineering Costs and Production Economics**, Vol. 17, 1989, pp. 111-124.

[9] Gershwin, S. B., "A Hierarchical Framework for Discrete Event Scheduling in Manufacturing Systems," Technical Paper LIDS-P-1682, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1987.

[10] Gershwin, S. B., Caramanis, M. and Murray, P., "Simulation Experience with a Hierarchical Scheduling Policy of a Simple Manufacturing System," **Proc. of the 27th IEEE Conf. on Decision and Control**, Austin, Texas, 1988.

[11] Gershwin, S. B., "Hierarchical Flow Control: A Framework for Scheduling and Planning Discrete Events in Manufacturing Systems," **IEEE Proceedings Special Issue on Discrete Event Systems**, 1989.

[12] Jones, A. T., Barkmeyer, E., and Davis, W. J., "Issues in the Design and Implementation of a System Architecture for Computer Integrated Manufacturing," **The Intl. J. of Computer Integrated Manufacturing** (A special issue on CIM architecture), Vol. 2, No. 3, 1989.

[13] Williams, T. J., **The Use of Digital Computers in Process Control**, Instrument Society of America, Research Triangle Park, NC, 1984.

DEVELOPING A CIM ARCHITECTURE FOR EDUCATIONAL, RESEARCH, AND TECHNOLOGY TRANSFER ACTIVITIES

RONALD WOOLSEY
BRUCE DALLMAN
RAYMOND KAPPERMAN
WILLIAM FORAKER
SUSAN LESKO
ROGER VICROY
LARRY HEATH

ABSTRACT

The following paper describes the mission, funding, development, and implementation of the Computer Integrated Manufacturing (CIM) facilities at Indiana State University. In 1985, faculty from the School of Technology developed a proposal to the Indiana Commission for Higher Education to implement a four year degree program in Computer Integrated Manufacturing. The program was approved and funded. This paper details the implementation of the facilities to support the educational, research, and technology transfer missions of the CIM program. Also discussed are the functional requirements of the system and the architecture as it has developed.

1. Program development

During 1984 and 1985, the School of Technology at Indiana State University proposed and received University and State approval for a baccalaureate degree program in Computer Integrated Manufacturing (CIM) to complement its other undergraduate and graduate curricula in manufacturing, mechanical, and related technologies. This program proposal was a continuation of a long history of commitment to the broad area of manufacturing for which programs had been originally established in 1962.

The philosophy of technology programs at Indiana State University has, since their beginnings, been to prepare graduates to assume technical and managerial positions in industrial organizations. To insure that graduates meet the needs of industry, input from industrial organizations is constantly solicited for curriculum and laboratory development. This philosophy has led to an approach in laboratory implementation utilizing production class equipment and facilities.

1.1 Organizational requirements and goals

From the beginning, it was quite clear that development of the CIM system with production class equipment would translate into the most expensive venture attempted by the School of Technology. Therefore, careful attention was devoted to the specification of hardware, software, and communication architecture in order to help ensure a system that would meet the requirements of multiple missions, including:

1. Support the educational mission of the programs in the School of Technology, School of Business, and other units on campus (i.e., Computer Science, Physical Sciences, Conference and Non-credit Programs);

2. Support research, development, and dissemination activities of the faculty, staff, and students; and

3. Support of the economic development mission of the University through direct technical and business assistance to regional industrial companies.

1.1.1 Educational mission

Support of the educational mission required that the CIM system serve as a resource to CIM and related technology courses as well as to other campus units including the School of Business. Instructional topics utilizing all or part of the system include: CIM architecture; control systems; robotics; plant layout and materials handling including automated systems; production planning and inventory control; materials requirements planning (MRP); Computer Aided Design (CAD) which includes two dimensional drafting and three dimensional modeling, with Finite Element Analysis (FEA); manufacturing simulation; cell design including simulation modeling; and Computer Aided Manufacturing (CAM) with process planning, code generation, and graphic simulation.

1.1.2 Research mission

The research activities of faculty required the CIM system to be adaptable to a wide range of configurations and functions to serve university personnel in their research and technology transfer activities. The primary emphasis of research activities of CIM related faculty is the application of existing computer-based manufacturing technologies to the problems of small to medium sized manufacturing organizations. A representative sample of these activities includes work cell design and development, analysis of designs, implementation of CAD, prototyping, concept drawings, and issues relating to the integration of design, manufacturing, and business functions.

1.1.3 Economic development mission

To support the economic development activities, the CIM system was designed to serve as a demonstration, application, and training platform for both hard and soft manufacturing technologies. This technology transfer mission is further facilitated through the operation of the Technology Services Center. The Center provides an interface between faculty expertise in the School of Technology and regional industries. Companies frequently contact the Center with questions concerning manufacturing processes, technologies, and industrial management issues. In cases requiring business assistance, the Center also serves as a link to our School of Business which operates a Small Business Development Center (SBDC). The Center's outreach mission specializes in helping small manufacturing companies implement existing methods and technologies to improve their competitive position.

1.2 Development expertise

During the process of developing the architecture for and implementation of the CIM platform, many sources of expertise were employed. Faculty and staff expertise was the primary source used throughout the process. Additional sources employed included an industrial advisory committee, a CIM consultant from Electronic Data Systems (EDS), the results of an industrial needs assessment conducted statewide with manufacturing industries, vendors directly involved in supplying equipment to the system, and various specialized information sources. Those sources included the Automated Manufacturing Research Facility (AMRF), the Air Force Manufacturing Center at Wright-Patterson Air Force Base, the Manufacturing Technology Information Analysis Center (MTIAC), the Society of Manufacturing Engineers (SME), and the National Association of Industrial Technology (NAIT).

2. Proposed system requirements and resulting capabilities

2.1 System requirements

The major system requirements for the total CIM system fell into three main functional areas. Those are 1) Computer Aided Design (CAD); 2) Computer Aided Manufacturing (CAM); and 3) Business/ Management functions. While not a separate requirement, each of the three main functions were to be fully integrated with the capability to have automatic file updating to a common database when changes are made to part geometry, available materials, schedules, etc. The criteria used to determine the vendor for the CIM system were 1) system capabilities and degree of integration of functions and 2) degree to which the vendor would enter into a long term partnership for the CIM program and laboratory support. Vendors which were finalists in the evaluation included Control Data Corporation, Digital Equipment Corporation, International Business Machines, and Prime Computer.

2.2 System capabilities

2.2.1 Enterprise level system

After a lengthy review process, Prime Computer was selected as the vendor. As purchased, the enterprise level system centers around a Prime 4450 supermini-computer, which is a 5.8 MIP machine with 32 megabytes of internal memory. External storage is on three 817 megabyte drives with communication conducted over 32 asynchronous lines and the University network. Figure 1 lists the software residing on this computer including the CAD system (MEDUSA), CAM functions (GNC, CNC programming) and business functions (MAN-FACT II), as well as other software available to the CIM system.

2.2.2 Computer aided design (CAD)

The CIM system's computer aided design capabilities center around the MEDUSA software package. The Medusa package features two dimensional drafting, three dimensional solids modeling, CNC code generation -- GNC with five axis capabilities, properties analysis, IGES and DXF translators, variational geometry, and interactive shaded viewing. MEDUSA is accessible on eight dedicated workstations (2-Tektronix 4225's and 2-4211's, 4-Prime

PW150's,) in addition to a laboratory which houses 26 micro-based workstations, emulating Tektronix 4107 terminals. Separate from, but in addition to the Prime capabilities, two Silicon Graphics Personal Iris workstations complete the high end engineering and simulation tasks. In addition, two micro-based CAD packages, Cadkey and Autocad, and numerous other support software complement this configuration with alternative design and drafting capabilities.

2.2.3 Computer aided manufacturing (CAM)

The computer aided manufacturing capabilities of the CIM system are housed in two separate laboratories. The machining laboratory focuses on CNC machine operation, on-line, and off-line programming. The CIM laboratory focuses on the integration of operations and includes five major cells/systems. Those cells are: 1) a machining workcell; 2) a processing workcell; 3) an assembly workcell; 4) a materials storage and retrieval system; and 5) an automated material handling system including an automated guided vehicle (AGV). These two laboratories have the following equipment in place and operational with varying degrees of integration into the total communications architecture.

Puma Unimate 762 industrial robot
GMF S110R industrial robot
GMF V210 vision system
Intelledex industrial robot
IBM 7535 industrial robot
IBM 7545 industrial robot
Bosch TS 2 modular transfer system
Republic Storage System's multi-cell ASRS system
Apogee Robotics AGV with a roller top load/unload system
Bridgeport Interact 412 Machining Center with the
    Heidenhain TNC 151 Controller
Okuma LB10 Turning Center with the OSP5000L-G Controller
Okuma MC-3V Machining Center with the OSP5000M-G Controller
ESAB Plasma CSX Shape Cutter with the Compu-Path 500 Controller

Laboratory Control Computers:
    IBM PS/2 PS/2 model 80
    IBM PS/2 Model 60
    IBM 7552 Industrial Control Computer/Artic Card
    Allen Bradley 1785-LT Programmable Logic Controller 5/15A
(See Figure 1 for application software list)

2.2.4 CIM business/management capabilities

The functions provided for shop floor management in the operation of the CIM system are provided by MAN-FACT II. This software package, residing on the Prime 4450 provides the following:

Purchase order entry for master scheduling
Personnel data
Bills of material

Figure 1. System Software Overview.

Parts master listing
Phantom bill of material
Engineering change order control/reporting
Manufacturing routing information
Work order releases
Production planning and control
Capacity requirements planning
Inventory management
Simulation capabilities with multiple business applications

Specifically for use by School of Business personnel, the MAN-FACT II software has the following business applications integrated with the total CIM system.

Personnel planning and forecasting
Financial management
Accounting procedures -- accounts payable, receivable, etc.
Cost accounting
Marketing/sales
Transfer pricing
Forecasting
Financial modeling
General ledger
Fixed assets
Payroll
Inventory turns analysis
Beginning/ending value analysis

3. System design overview

3.1 Communication requirements

With the identification of system requirements and capabilities, it became apparent that adding additional communication traffic to the existing university network would decrease the throughput rates significantly. The obvious solution is to segment the CIM system from the university network by use of a network bridge. This provides for integration of the CIM equipment while still allowing access to other campus computing resources.

The complexity of the communication architecture was also effected by the physical location of the 4 main facilities. Those facilities are: the Design laboratory, the CIM laboratory, the Machining laboratory, and Business/Management terminals which are all housed in separate buildings on campus. The Design laboratory encompasses the design and development of part geometry. The Machining laboratory focuses on CNC machine operation, on-line, and off-line programming. The CIM laboratory provides the platform for the integration of all CIM functions and has evolved as the control and demonstration center for all computer integrated manufacturing processes. Business/Management functions are assessable through any computer or terminal tied to the campus network.

3.2 OSI design model

Computer integrated manufacturing environments require specialized computers to accomplish a variety of concurrent tasks. Requirements of this magnitude dictate the type of computing power needed for each application. The diversity of hardware available from vendors in the CIM market requires organization of communication parameters to allow interconnectivity between protocols. The system configuration at ISU has evolved as a multi-vendor solution and therefore dictates the use of a formal communications structure. To provide this structure, the Open Systems Interconnection (OSI) model has been followed and is explained below. (See Figure 2)

The function of each of the layers in the OSI model are summarized below [INTEL85].

The **Physical Layer** describes the physical media over which the bit stream is to be transmitted. This layer specifies type of cable (coax, twisted pair, etc.), connectors, signal levels, bit rate, data encoding method, modulation method, and method for detecting collisions in contention networks. In short, this layer describes the physical media over which the bit stream is transmitted and the method of transmission, i.e., baseband or broadband.

The **Data Link** describes rules for transmitting on the channel (made up of the encoder/decoder, transceiver cable, and transmission medium). Such items as the format of the information (frame) and procedures for gaining control of the channel (access method), transmitting the frame and releasing the physical media are specified by the Data Link Layer.

The **Network Layer** controls switching between links in a multihop network. The Network Layer is not necessary for a single Local Area Network (LAN) system because all stations connected onto a LAN share the same channel. This Layer is critical in gateway, communication server, and dialup-communication applications.

The **Transport Layer** ensures end-to-end message integrity and provides for the required quality of service for exchanged information. For example, end-to-end acknowledgements and flow control are performed by the Transport Layer.

The **Session Layer** establishes and terminates logical connections between network entities. This Layer is also responsible for the mapping of logical names into network addresses.

The **Presentation Layer** provides for any necessary translation, format conversion, or code conversion to put the information into a recognizable form.

The **Application Layer** provides network based services to the end user. Examples of network services are distributed databases and electronic mail. The Application Layer is not to be confused with the end user application itself.

With the OSI model in use, the Physical and Datalink layers can vary without effecting the types of protocols used at individual workstations. For example, layers 1 and 2 of a network

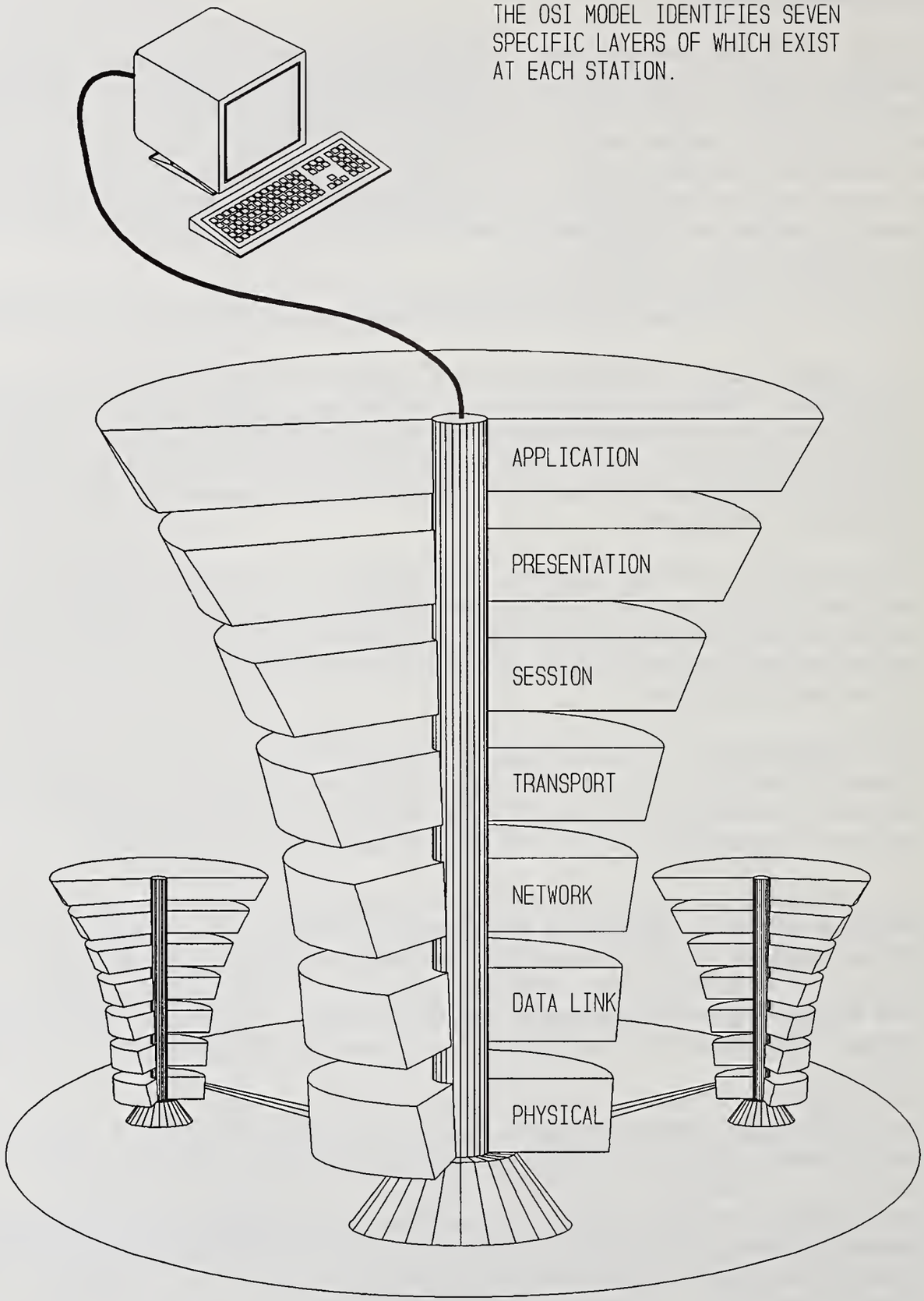


Figure 2. OSI Model.

can be changed from carrier sense multi-access/collision detect (CSMA/CD) based (e.g., IEEE 802.3) or token ring based (e.g., IEEE 802.5)(and the associated hardware), without affecting layers 3 through 7.

3.3 Design considerations

The first two layers of the model are typically implemented through one of the following methods. [BATES87]

Ethernet: CSMA/CD, Baseband, Coax
Arcnet: Token bus, Baseband, Coax
MAP/TOP: Token bus, Broadband, Coax
IBM: Token ring, Baseband, Twisted pair

Careful consideration must be taken when analyzing the benefits of each system. As the characteristics of the CIM system were established, the patterns of data flow became more clearly defined. For the ISU application, ethernet was chosen as the solution for the lower layers in the OSI model. This allowed the physical characteristics to be defined and the network structure to be developed. Due to the variety of computing needs required by and the inherent traffic that would be created by the CIM system, a segmented structure was established to insure the highest throughput levels to each network. Figure 3 diagrams the major network connections and features, however none of the local area networks are detailed. Figure 7 represents the entire structure and figures 4 through 6 provide specific details of each of the CIM system network segments.

Two contributing factors dictating the use of a segmented backbone (trunk-line) are the physical distance between segments and the requirements of the system's specialized computers. Individual computers were placed on separate segments which were bridged to minimize traffic throughout the entire backbone while allowing optimum throughput rates for the CIM system and other independent university resources.

The interconnection through bridges will allow the traffic concentrated on each segment to be isolated. This isolation will also allow the simultaneous operation of a central tape drive to backup individual servers on a scheduled basis with no reduction in performance on other segments. Although, the system being backed-up will notice a slight reduction in performance, maximum utilization is still obtained throughout the network.

3.4 Communication methods

The use of a multi-port asynchronous device (port-selector) allows communication with any of the mainframe/supermini computers on campus via asynchronous lines from any terminal or PC within each facility. An alternative method is provided through asynchronous servers located on each of the networks. A Netware (Novell) to Transmission Control Protocol/Internet Protocol (TCP/IP) gateway is used to communicate between the different protocols. This also allows the general sharing of databases which exist on any of the systems located on the network. Such sharing can be accomplished through File Transfer Protocol (FTP) which provides for bi-directional communications between host, file servers and/or personal computers. Also available is the Remote Terminal Protocol (TELNET) which provides terminal emulation for access to TCP/IP hosts.

Figure 3. ISU Computer Network.

4. CIM system architecture

4.1 Communications with the Prime 4450

The selected computer for the CIM environment, a Prime 4450, is attached to the backbone on a separate segment which allows for a direct link with all computers needed to function in the CIM system (Figure 7). Through the use of TCP/IP, the PRIME 4450 communicates with eight graphic workstations for design purposes. Two Unix based super-workstations are available via this connection to share part geometry/data for finite element analysis and simulation of individual workcells. A third means of connection is the multi-port asynchronous communication lines which allows classroom and laboratory areas to have primary access to operating software. This multiple connection arrangement is important to provide a flexible teaching environment for classes and/or student projects. Further, this arrangement allows additional facilities to be readily added to meet the need for expansion to other academic units which elect to participate in the CIM program.

4.2 Design laboratory

The Design laboratory (Figure 4), is concerned with instruction in the development of part geometry and evaluation of design integrity. Located in this laboratory is a 386 file server utilizing Novell software connected via thinnet to 26 286/AT based workstations. In addition, communication with the CIM host computer is accomplished through asynchronous lines to the port-selector.

Each workstation is a 80286 based microcomputer with an 80287 coprocessor, VGA graphics with flat tension mask (FTM) monitor, digitizing tablet, local printer, and high capacity drives. These individual stations use Tektronix emulation software to take advantage of the asynchronous connection for a variety of purposes. For example, access to the Prime Medusa 2D, 3D, Geometric Numerical Control (GNC), and Finite Element Analysis (FEA) software can be accomplished with considerable cost savings as compared to the use of specialized terminals.

Other duties the file server performs include basic data file collection and transfer, on-line immediate information storage, security and record keeping. Also, an IBM XT computer in the laboratory is dedicated to tape backup, maintenance, system status, and mainframe plot facility access.

This server directly controls three plotters which may be queued from any campus network. Through the use of a separate plot spooler (IBM XT) and a Value Added Process (VAP), the network allows for the operation of a fourth plotter with control software in the foreground and an inkjet printer in continuous operation in the background. These are also accessible from any campus network.

The aforementioned capabilities allow many types of plot data to be directed to any of the plotters on campus. Through a special application program, virtually all graphic images displayed on the 286/AT workstations can be plotted on any plotter that will support the Hewlett Packard Graphics Language (HPGL).

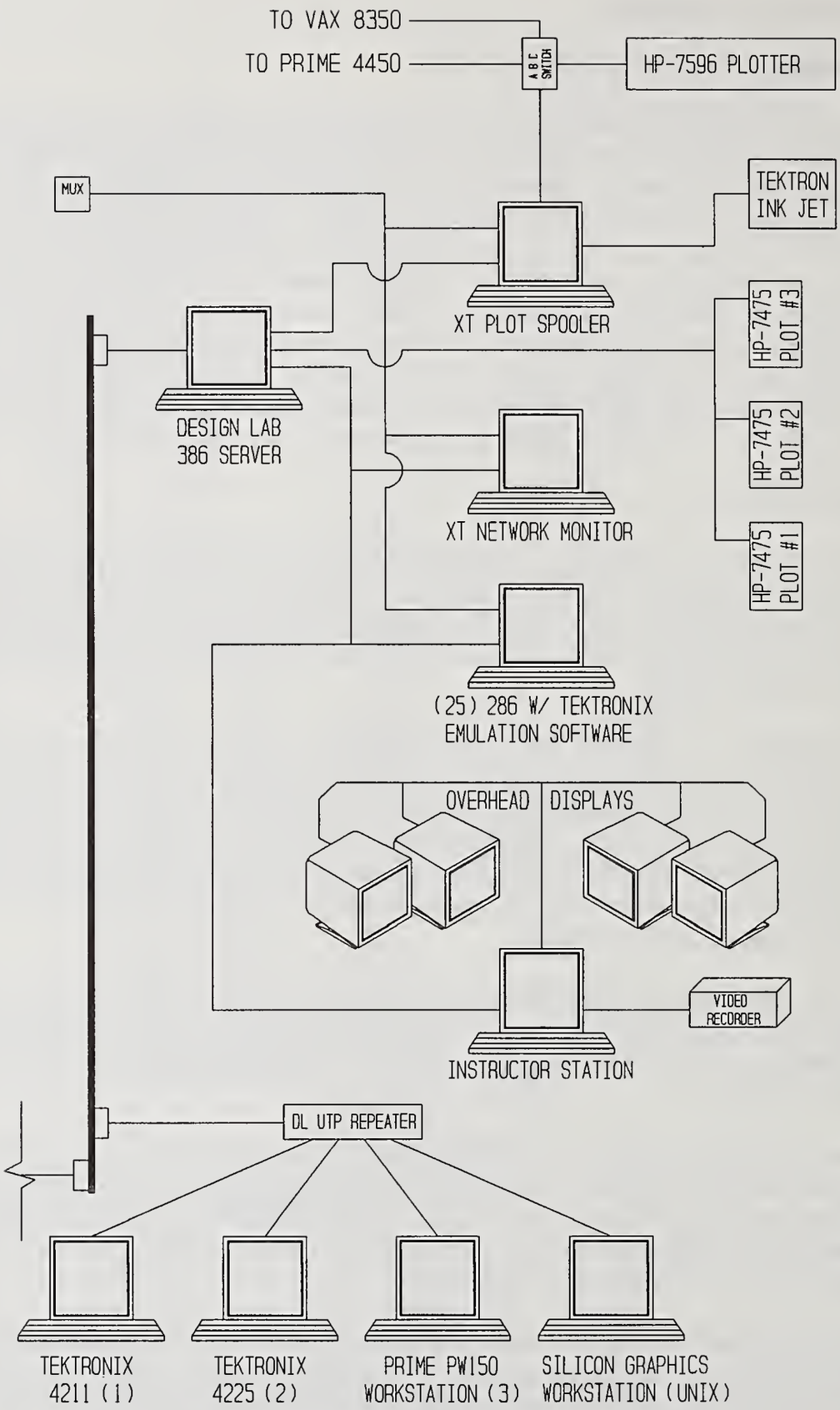


Figure 4. Design Laboratory Network.

A unique feature of this laboratory is the instructor's station which includes another 286/AT workstation, a video graphics converter (VGA to NTSC), and four 25 inch monitors arranged overhead around the laboratory for simultaneous viewing. This allows the instructor to perform real-time demonstrations at the station, while recording on video tape for review at a later date.

4.3 CIM laboratory

The CIM laboratory (Figure 5) is the central location for manufacturing operations. A PS/2 model 60 serves as the local communications coordinator and mass storage device. This server is connected to the CIM backbone segment for network communications and to most devices in the laboratory for local functions. It provides the link between the shop floor controllers and other campus networks via the backbone.

Local network devices include an IBM 7552 which is used as the main communications controller for all shop floor equipment. Additional devices connected to the server include and PS/2 model 80, seven XT class computers used as multi-purpose terminals/processors, and an additional XT class computer that serves as a laboratory status display and shop floor data manager. Also located on the CIM lan is a separate cluster of AT and XT class computers which provide additional computing, printing, and plotting resources. Local printing and plotting resources are accessed similar to the procedure used in the design lab.

4.3.1 Shop floor control

To effectively operate shop floor activities in a CIM environment, it is necessary to establish an internal protocol for run-time communication that can be used for real time monitoring and control. Allen Bradley's Advisor PC running on the PS/2 model 80 and the IBM 7552 acting as a communications controller comprise the CIM laboratory shop floor control system. This system provides real time control of all laboratory functions. Figure 5 indicates the devices connected to the 7552 which include two IBM assembly robots, an Intelledex assembly robot, one Puma/Unimate and one GMF continuous path robot, a Bridgeport CNC machining center, a GMF vision system controller, the controller for the AGV, and several shop floor data collection devices. The shop floor control system routes data files, instructions, and other required data to the shop floor and also receives production status information. The Advisor program and its data files are stored on the server.

An additional task completed by the PS/2 model 80 is control of the ASRS system. As seen in figure 5, the ASRS is controlled by an XT computer directly connected to the PS/2 model 80. Information on raw materials, work in process inventory, and finished goods is stored in the inventory database for use by the shop floor control system and the business/management software.

4.3.2 Control area workstations

A major function of the seven XT class computers is to provide local off-line programming stations for CNC equipment, robots, and other automated equipment on the shop floor. It is anticipated that all manufacturing programs will be written off-line and subsequently

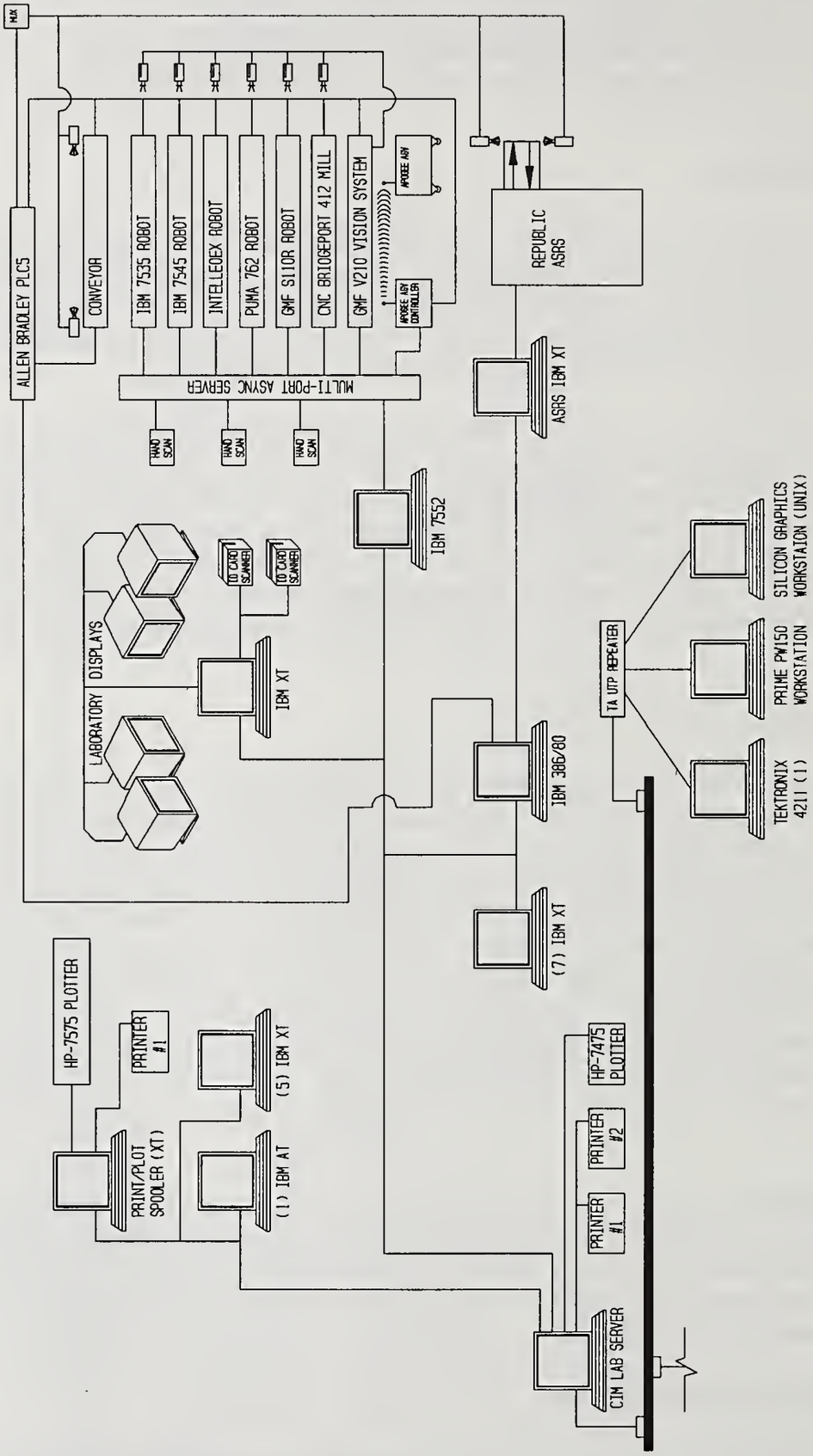


Figure 5. CIM Laboratory Network.

downloaded to their respective devices for final calibration and debugging. After programs have been qualified for production use, they are then transferred back to their original device and are stored on the file server for automatic download to the target devices as necessary. The programming languages for all of the devices will be available on the network. Off-line programming will usually be accomplished in the control area of the automated facility, but can be done on other network devices that have the requisite computing and graphics capability.

These machines will also serve as terminals for the 4450 to access the Man-Fact II software. Production scheduling, inventory control, and other manufacturing management and business functions will be performed on these machines. These computers will also be used for quality routings, personnel reports, wordprocessing, spreadsheets, and other miscellaneous functions.

4.3.3 Additional production area systems

When a product is manufactured, a sequence of steps is required using many different protocols. The Internal Protocol for Product Development (IPPD) in the CIM laboratory provides the capability to concurrently manufacture multiple custom products. The IPPD is being developed at ISU for the CIM laboratory to facilitate communications between different industrial devices within the existing architecture.

An XT class computer will be used as a controller for monitors and bar code scanners distributed throughout the laboratory. This system provides data input for shop operations and monitoring of the production status of the laboratory.

The AGV receives its instructions from a radio frequency (RF) controller linked to the 7552. Those instructions are based upon routing information and production operation requirements.

To provide for off-line preparation of tooling and fixturing for a product or product families, each manufacturing cell has two stations for smart carts. One cart provides for quick change tooling. The other provides customized product fixturing. When production schedules require fixturing and/or tooling changes, the existing carts are replaced with a new carts having the proper configuration. When plugged into a tooling station, each cart receives plant air, 110v power, digital control, and communication capability for actuation and identification of cart components.

4.3.4 Shop floor part tracking

Part tracking on the shop floor is accomplished by bar code scanning. All ASRS totes have attached bar codes. Each bar code is identified in the inventory database so that the system can track the location of all parts. Scanners are located throughout the laboratory at key points for part tracking. Each scanner communicates with the Allen Bradley Programmable Logic Controller (PLC-5) through the Serial Adapter Module (SAM) via Allen Bradley's DF1 protocol, which is a modified ANSI X3.28/D1/F1 specification.

4.3.5 CIM laboratory digital control

The Allen Bradley PLC-5 is the center of the digital I/O control. The PLC-5 is directly connected to the PS/2 model 80 and receives its program from the Advisor. The PLC-5 controls all CIM lab I/O including robotic cycle initialization and status, machine tool status, load and unload station control, and conveyor movement.

The assembly cell conveyor system is a particularly complex control problem as it is a combination of many independent devices. In order for these devices to function as a coordinated unit, the PLC sends and receives signals of various voltages, both AC and DC, based on the logic control program.

The Advisor tracks changes in all I/O status and provides graphic display. The PLC-5 also has the capability to communicate via Allen Bradley's Data Highway proprietary network.

4.4 Machining laboratory

The machining laboratory is detailed in figure 6. A file server is connected to the network and controls the activities of the laboratory. Four XT class machines are used for off-line CNC programming, graphic verification of tool paths and material removal, and program debugging. As programs are completed and debugged, they are routed through the network to the appropriate device. Programs can also be generated and downloaded to these devices from other locations on the CIM network.

5. Summary

Indiana State University has developed a computer integrated manufacturing system that serves multiple missions. The system is modular in design to serve the educational needs of faculty and students. The distributed architecture was also designed to serve the outreach mission as many of the research and technology transfer activities of the faculty are targeted toward the needs of small to medium sized manufacturing organizations.

While preserving distributed and microcomputer based capabilities, the CIM system also provides high end CAD, manufacturing planning, and information integration functions typical of larger systems. This has resulted in an architecture with some hardware and software redundancy and communication characteristics that would not be found in a more focused implementation. However, the system described in this paper has been found to accommodate a wide variety of requirements while still providing the functionality for teaching, demonstration, and customized application development.

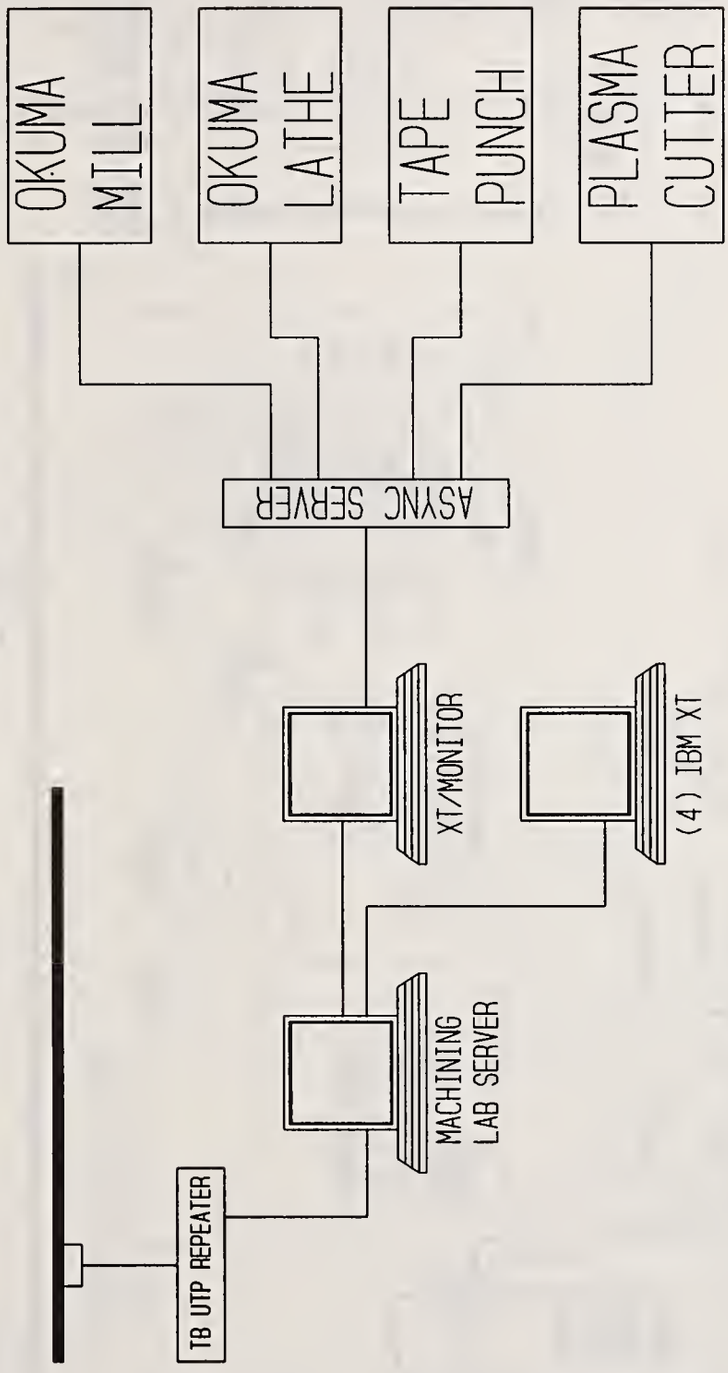


Figure 6. Machining Laboratory Network.

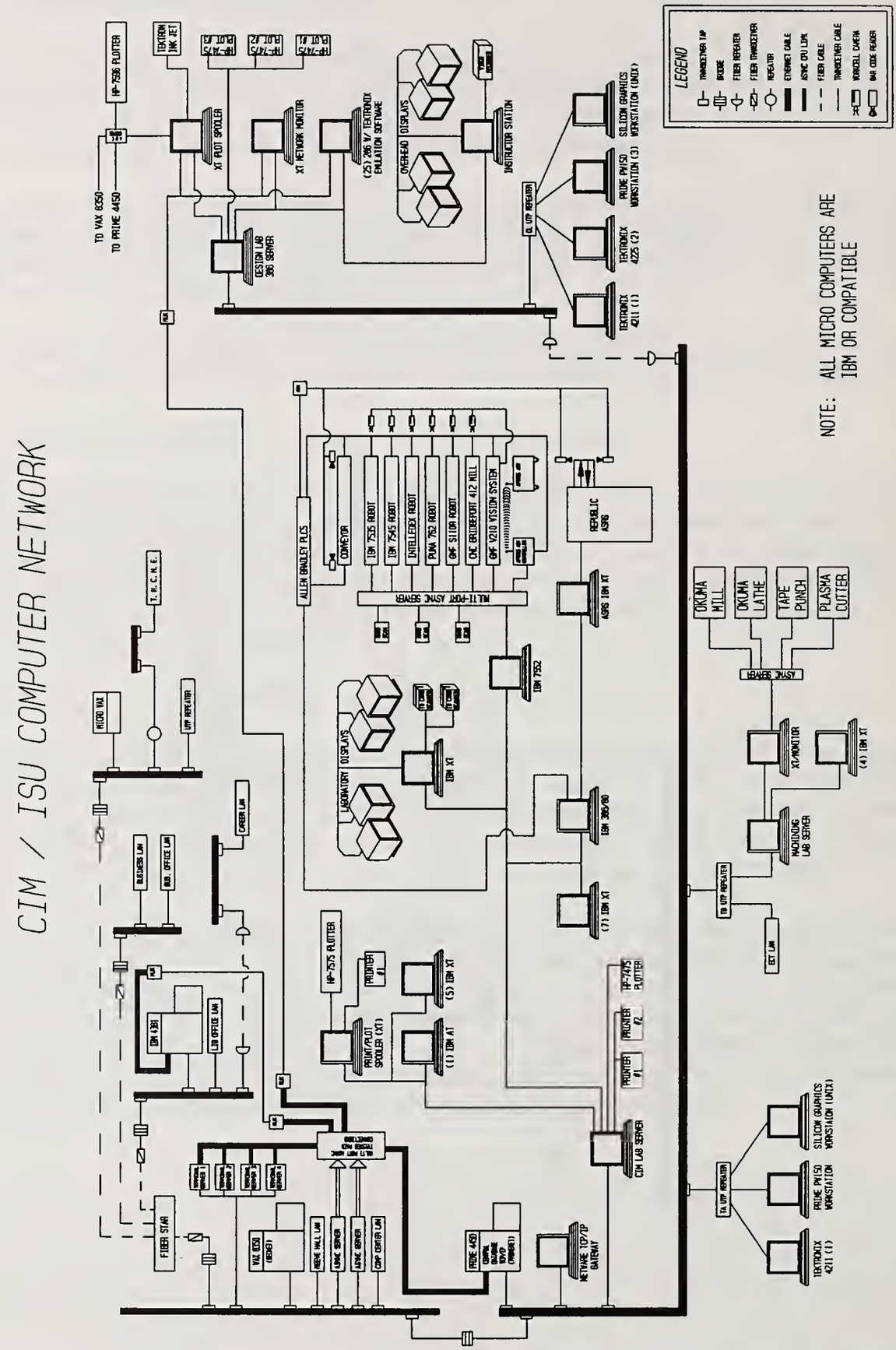


Figure 7. Complete CIM/ISU Computer Network.

REFERENCES

1. Paul Bates, Practical Digital and Data Communications ... With LSI Applications, Prentice Hall, Inc., New Jersey, 1987.

2. IntelCorporation, Microcommunications Handbook, Product Line Handbook, 1985.

CIM ARCHITECTURE: ONE PERSPECTIVE

ALLAN ANDERSON
THERESA JENNE
KRISHNA MIKKILINENI

Abstract

The implementation of Computer Integrated Manufacturing systems is made difficult in practice by a number of realities such as multiple and changing vendor sources, existing systems and equipment, changing business environments and management priorities and diverse and complex technology subsystems. An architecture based approach has been developed and used within Honeywell with success consisting of inter-relating business and system principles, integration models and subsystem formulation and implementation guidelines. We present here our definition and perspective of a CIM architecture and relate some of the experience we have gained to date.

1. Introduction

To guide the implementation and evolution of manufacturing operations toward integrated manufacturing and to help coordinate individual development efforts and projects, a CIM architecture is needed. Future manufacturing systems defined and implemented using this architecture are expected to increase the cost-competitiveness, productivity, and effectiveness of all aspects of manufacturing.

We developed a baseline CIM architecture in Honeywell that can be tailored to organization-specific needs, while maintaining coherence between implementations. The architecture provides a systematic methodology and framework for developing a top-down CIM plan which is used for implementing manufacturing activities or systems in a piece-meal fashion. We illustrate the process of developing a CIM architecture and present the key components of an architecture in this paper. The benefits of the architecture to Honeywell are also outlined.

There are many different reasons for needing and developing a CIM architecture. The major reasons for architecture development are presented in Section 2. The objectives of CIM architecture are then outlined in Section 3. The definition of a

CIM architecture we formulated in Honeywell is presented in Section 4. This definition is different from the traditional definitions of an architecture (e.g., functional definitions, philosophical statements of direction and policy from management, connectivity diagrams, implementation systems specifications, etc.). The first component of our CIM architecture, principles, is presented in Section 5. The second component of the architecture, integration models, is discussed in Section 6. The third component of the architecture, guidelines, is presented in Section 7. In section 8, we outline the benefits and uses of the CIM architecture in Honeywell. Finally, a conclusion is given in Section 9.

2. Reasons for needing and developing the CIM architecture

There are several reasons for needing and developing an architecture. The reasons that drive the need for architecture development include integration requirements, technology trends and business strategies. The following discussion examines the architecture drivers in more detail.

A CIM architecture is needed to provide a systems approach to guiding the development, modification and implementation of both manual and automated functions necessary to improve the productivity, cost-competitiveness and effectiveness of all aspects of manufacturing.

The top-down architectural approach of defining and implementing manufacturing systems is necessary to plan implementation and integration of manufacturing subsystems. An architecture can also be used for planning modifications to upgrade existing manual and/or automated systems to improve efficiencies and to reduce costs. In contrast, a bottom-up approach to planning and implementation of manufacturing systems usually results in "islands of automation." These islands initially provide significant improvements over a paper-based environment, but today, these systems often have redundant information in incompatible formats on diverse systems. In addition, the bottom-up approach only shows improvement in a small area, ignores the problems it may impose on the overall system and disregards changes in future requirements. An architecture is the cornerstone of the top-down planning approach and thus is an essential ingredient to planning or implementing a manufacturing system.

The speed with which key manufacturing systems technologies (e.g., information technology and distributed processing) have been changing recently is a major driver to develop an architecture. This rapid change in technology leads to (1) confusion in understanding the variety of technological solutions and approaches offered in the marketplace; (2) uncertainty of the availability, maturity, compatibility and stability of manufacturing systems solutions; and (3) dwindling confidence in the ability of the chosen solutions to meet *all* the key current and future requirements. Therefore, an architecture that provides guidance in technology selection and the implementation of manufacturing systems is essential if we are to deal effectively with rapid technological advances.

An architecture is essential for planning effective, efficient and integrated approaches for meeting emerging customer requirements such as data security

(security is a requirement for DOD contractors) and the capture and delivery of large amounts of product-related data (e.g., NASA contractors will be required to have a paperless environment).

The emerging business/industry trends—such as reduced product life and shortened time for the design-to-production transition (which in its extreme is simultaneous engineering)—imply the need for more flexibility in the operation of manufacturing systems. They also imply the need to automate the flow of information in the organization and to improve the interaction within and among the various functions of the manufacturing enterprise. An architecture is essential for developing cost-effective, efficient and mutually compatible approaches that would cope with these new trends.

Commonality is another key driver of the CIM architecture. A major corporate goal throughout manufacturing industries at this time is to reduce duplication of effort in the development and implementation of manufacturing systems. Obviously, there are benefits to be gained from commonality, such as cost reduction and improved operations resulting from shared knowledge and approaches. At the same time, a common set of solutions may not meet the requirements of individual business units in a corporation. The CIM architecture is not intended to provide a common set of manufacturing systems solutions or techniques but to provide a common starting point for all the manufacturing units in an organization to use in developing their own organization-specific architectures.

Standards are another important architecture driver. They provide the ability to integrate manufacturing systems easily. The architecture will provide the basis for choosing the right standards for implementing different aspects of the manufacturing system by capturing a set of applicable standards.

3. Objectives of a CIM architecture

The specific objectives of the CIM architecture include:

- Providing a top-down development approach that is needed to understand the requirements and interfaces of the various subsystems so that each subsystem can be implemented from the bottom up and independent of other subsystems with assurance that the various subsystems can be integrated with each other.

- Providing criteria for making selection/evaluation decisions regarding choices of implementation techniques and procedures.

- Providing an up-front plan of the interactions and integration among the different subsystems in manufacturing to reduce the complexity of incremental integration.

- Providing a common definition of subsystem requirements at a sufficient level of detail to ensure a robust or stable manufacturing system and to prevent subsystems from becoming obsolete or requiring special interfaces each time an interfacing subsystem is modified or replaced by a new subsystem.

• Providing the criteria for vendor evaluation and limiting the choice of alternate computer/control system or application vendors.

• Promoting communications and sharing of architecture development and subsystem implementation among CIM organizations.

• Providing a living architecture specification document that is validated, modified, and refined with further work and analysis as implementation experience is gained over time.

4. CIM architecture definitions

The term "architecture" conjures up various images to different people. To some, an architecture is a diagram of the connections between platforms in a certain hierarchy; to others, an architecture may be a set of data flow diagrams or statements which describe an organization's technology policies. We define the architecture to be the overall context or framework that provides the basis for making decisions and evaluating alternatives in a rapidly changing business environment. The architecture does not provide a solution; rather, it provides the discipline and the mechanisms needed to evaluate various alternatives and to select a solution with reasonable confidence in the effectiveness and impact of the selection. The architecture is documented as a living document that is initially developed and then enhanced and continually updated to reflect the changing business environment. Architecture is a process that totally transforms a company from the past ways of running its business into the future.

We define the CIM architecture as a structured interrelationship of principles, functional models and guidelines providing the framework needed for designing or modifying an integrated manufacturing system. The key pieces of the CIM architecture are:

• Architecture principles, which are broad, but well-focused statements of direction used in designing an integrated manufacturing system.

• Integration Models, which provide a description or a model of the different functions or activities performed within a manufacturing system, the information flows among these functions and the interrelationships among the information.

• Guidelines, which describe an approach to grouping or partitioning the different functions so that they can be assigned as implementation subsystems. The guidelines also provide a set of criteria to select or evaluate alternative implementation choices corresponding to different aspects of a manufacturing system: platforms, data communications, databases and applications. Figure 1 depicts the relationships between the architecture components, business strategies, project planning and projects.

HCAM Architecture

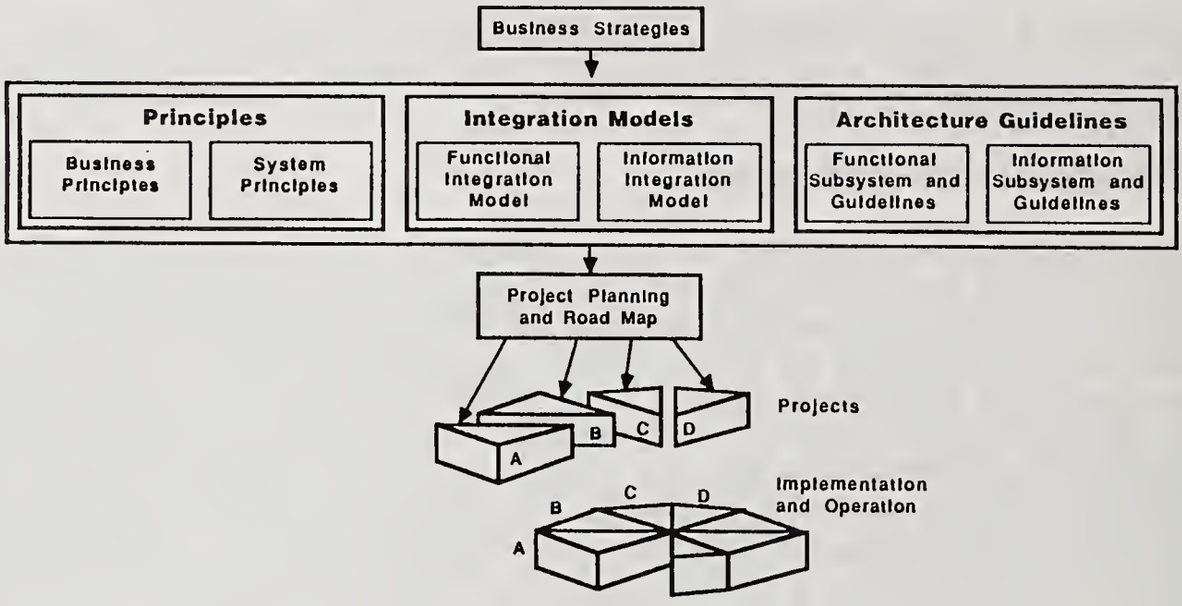


Figure 1.

A manufacturing systems architect must design solutions in the face of many unknowns, such as changing requirements, future technologies and so on. To deal with the complexities and unknowns, a manufacturing systems architect must develop a core set of architecture principles and use them to guide his/her decision-making. Architecture principles should be defined corresponding to each category of requirements defined by customers or the business environment. To provide an analogy, a building architecture is designed using a set of principles regarding categories of requirements, such as building codes, costs, aesthetics, floor space and other such typical categories of requirements. The architect uses these principles as criteria to make decisions during the design of the house. These principles guide the design to meet specific requirements. For example, a principle stating that a duplex outlet must be provided every 12 feet along the walls of a room in general fulfills the requirement of providing convenience. Note that there is a distinction between requirements (e.g., convenience) and principles (e.g., the principle of providing an electric outlet every 12 feet).

The second component of the architecture is a set of integration models. Integration models describe the structures for integrating and providing a set of functionalities, interfaces and features. They describe the functionality and information corresponding to the different aspects of the manufacturing systems. Integration models need to be developed for the different aspects of a

manufacturing systems architecture, including business processes; management methods; manufacturing, engineering and business functions and processes; computer/control platforms; databases; networks; applications; etc.

In the building analogy, the different aspects of the building architecture include its structural plans, a schematic for electrical distribution, a schematic for plumbing distribution, excavation plans, etc. All these plans together provide the architecture plans for building the house and they determine the features and functions of all the parts of the building. For example, the electrical schematic outlines the quantity and location of duplex outlets. These plans are analogous to integration models.

After the models are developed, the third component of the architecture—namely, the guidelines—needs to be developed. These guidelines or design rules are needed to guide the implementation of each of the various aspects of the architecture mentioned above. Guidelines are specific applications of the principles to each aspect of the architecture implementation. For example, the kitchen electrical subsystem guideline can be derived from the principle that a duplex outlet must be provided every 12 feet along a wall. A guideline for the kitchen electrical subsystem may state that power must be provided to the outlets specified in the electrical schematic by two 20-amp circuits. Similarly, guidelines corresponding to the various aspects of implementing a manufacturing systems architecture provide criteria for selecting and evaluating various potential implementation choices.

5. Architecture principles

Principles are the most stable element of an architecture; they represent continuity and relative stability through the evolution of an organization's manufacturing systems architecture. ***Principles are the criteria used to make decisions in designing and selecting different systems*** while implementing the integration models. Principles should be specific enough to guide the designer in making decisions during the development of the system solutions. They should avoid statements that are readily accepted by all and do not contribute to the decision-making process. Since principles are used to make decisions, each one should be discussed and evaluated for its pros and cons. Adherence to some of the principles will depend on cost-versus-benefit trade-offs. During evaluation and selection of the principles, the cost of adhering to a principle should be considered.

A principle is generally applicable across all layers of the architecture and across all the manufacturing functions. It assists the designer in creating a robust system, but the designer must determine how to implement it. In contrast, a guideline (presented in Section 7) is a specific implementation or application of a principle to a particular manufacturing function. A guideline is *not* applicable across all the manufacturing functions and addresses only the specific requirements of a particular function.

Principles may define the organization's direction along the following subjects:

• Orientation to risk—Principles can establish policies about the introduction of new technologies, experimentation with infrastructure, system development and management.

• User autonomy—If an organization values user autonomy, a principle describes how the users would make technology decisions.

• Technology perspective—Principles can establish whether new technologies are introduced primarily for cost savings or as strategic tools to maintain the organization's competitive edge.

• Technology solutions—Principles can show a preference for general technology solutions versus specific optimal solutions.

• System development—Principles can be used to guide decisions on in-house system development versus buying vendor packages, using standard vendor offerings versus customized packages, and establishing vendor preferences.

• System specification principles—Principles can establish system deliverables. For example, all of a particular department's users will have access to computing services, bar-coded input is preferred over keyboard input on the factory floor, all word-processors will use the same applications. Platform, application, data and communications principles can help specify the different aspects of the system architecture.

• Business principles—Principles can clearly state an organization's policies, goals, strategies, rules and constraints concerning general as well as specific aspects of all the functions within and related to manufacturing.

Let us now illustrate how principles can be derived from business objectives. For example, the organization may need to respond rapidly to market demand or outside competition for new products. This implies a frequent turnover of product lines and rapid obsolescence of specially designed systems. Therefore the organization's objective is to encourage flexibility. Based on these objectives, the following business principle can be derived:

> The manufacturing system shall manufacture the current product as an instance of a general class of products that it can produce. This class includes:
>
> Circuit boards:
>
> thru-hole and surface mount components and mixtures with thru-hole components on one side of the board.
>
> rectangular form factors up to 6 by 8 inches.
>
> component placement accuracy of 10 mils

These system environment objectives are now decomposed into architecture principles that state the system objectives explicitly with respect to some aspect of implementation of the system architecture, such as defining a standard set of tools, interfaces and applications. For example:

- Systems that can grow incrementally to accommodate new demands on networks, platforms and software capabilities;

- Systems that implement general technology solutions, which can be modified easily, rather than specific optimal solutions designed for a particular implementation;

- Systems that promote standard tools, interfaces and programming languages to minimize the time and cost of implementing new or modified systems, training users, etc.

6. Integration models

The integration models of the CIM architecture consist of two complementary components: the information model and the functional model. The functional integration model describes the functional activities within the manufacturing domain and the interactions between functions as flows of information between them. Information flows provide the foundation for the information integration model. The information model captures the manufacturing information and describes data to data relationships. The function integration model captures function to function relationships and information to function relationships but the actual structure of the information elements themselves -- the interconnecting relationships between information -- is not explicitly captured. Information integration models on the other hand capture the relationships among the information. The integration models provide the basis for analyzing and designing the systems design and for applying the architecture principles presented in the previous section.

6.1 Functional integration models

The objective of the functional integration models is to provide a tool to assist each organization in analyzing and developing a standard description of the manufacturing functions performed in the organization. The functional integration models also provide the interaction of the information flows among different functions so that system developers can logically analyze and group these functions into applications and implementation subsystems and determine which functions are be performed manually or which should be automated. The functional integration model provides the "big picture" containing an overview of all, not just a few, of the CAM/CIM functions that each organization can use to develop the "as-is" and "to-be" functional models. The ultimate objective of developing the functional integration models is to develop an implementation systems architecture that can smoothly and cost-effectively meet all the integration requirements that can be derived from the functional interactions. The functional integration model we present here is also intended to serve as a

common and consistent method of communication of the CIM functionality between organizations.

The function integration models consist of the following components:

- Function integration diagrams, which are graphical descriptions of the functions and information flows among the functions.

- Standard specifications, which are textual descriptions of each function.

- A data dictionary, which provides definitions for the information contained in the information flows of the function integration diagrams.

6.1.1 Function Integration diagrams

The function integration diagrams (FIDs) are figures that provide a graphical description of the functions and how information flows are integrated among these functions. FIDs can be considered to be equivalent to data-flow diagrams or context diagrams as they are defined by structured analysis methodologies. Each function in an FID is an activity or task that is performed within the manufacturing facility. Each function requires information from other functions to perform its task, and the function needs to provide information to support other functions. The function integration diagrams consist of several levels of detail. Each high-level function is decomposed into finer levels of detail by descending into lower level function integration diagrams that display the activities within the high-level function in much greater detail.

The information categories that describe the interactions between two functions are annotated on top of the arrow connecting the two function boxes. Each task or function requires certain information from other functions that it needs for accomplishing the task (inputs). Upon completion of the task, each function must provide information to other functions (outputs). The direction of the flow of information between two functions is shown by the direction of the arrow.

6.1.2 Standard specification of functions

A standard specification is a description of each function or process in the FID. The standard specifications provide a common means of describing and understanding CIM functions within organizations. The following is an example of the types of information a standard specification needs to include to describe the function adequately for reference and communication purposes.

- Objective—What is the purpose of this function?

  Example: Schedule operations to fulfill the production order. Calculate the dates when the operations are to be performed to meet the demand order. Allocate workcenters and sequence operations.

- Activities—Relate the activities needed to correlate inputs and outputs and how the objective is accomplished.

Example: Given the product definition and configuration and the process plan and configuration, calculate the priorities for the various orders (or receive priorities from MRP). Assign start dates and completion dates to batches of components (an order) and assign workcenters where the jobs are to be performed. Schedule operations backward from the due date of the demand order. Retrieve orders, calculate and assign priorities, analyze resource usage and expected throughputs, and determine schedule. Assign start/due dates to batches or lots of individual components and identify workcenters to perform the work. Once the schedule has been produced, determine the resource requirements and generate work orders for the various workcenters to provide resources such as tools, material and labor.

- Planning Horizon—if applicable (shift to 1 week).

- Scheduling Interval—if applicable (minutes or hours).

- Primary Inputs—Primary driver of the function (i.e., analogous to a command signal). An example is the production order.

- Primary Support Inputs (Support Data)—Data needed to support the decision-making or planning process. (This includes the process plan and configuration, product definition and configuration.)

- Primary Outputs—Primary outputs, which are input drivers to lower level functions (work orders and a dispatch list).

- Primary Support Outputs (Support Data)—Support queries, feedback of performance, etc. (e.g., production-order status).

- Implementation Trends and Methods—Optional.

6.1.3 Data dictionary

The data dictionary contains a definition and description of the information that flows into and out of the different functions in the FIDs. The data dictionary describes (1) the information category name annotated on the information-flow arrows in the FIDs, (2) the granularity of the information (i.e., an atomic or basic data item or an object of data or data structure containing any or all of several data items) and (3) the other information categories to which it is related. The data dictionary tool will then provide a variety of services including where-used type retrievals, change control, reporting and data modeling support.

6.2 Information integration models

The term "information model" means different things to different people and requires some clarification. There has been intensive interest in information modelling for two decades. Many types of diagrams and methodologies have been developed too numerous to describe here. For our purposes, an information integration model is a collection of clearly and precisely defined manufacturing

information using natural language descriptions (i.e. English) to help people understand clearly large systems of information.

The objective of the information integration model is to represent and communicate to people a wide range of manufacturing information clearly and precisely, explicitly showing the relationships between information elements. This representation should connect the information in the model to the function integration model information flows.

A secondary objective is the support for systems formulation and implementation. The information integration model should support the planning and analysis of systems for information management in manufacturing.

The information integration model allows the subsequent data models in manufacturing to provide integrated data across a wide range of manufacturing areas to support CIM. The model improves planning of implementation and integration activities, training of employees and analysis of changes in the manufacturing system. The clear understanding of the manufacturing information leads to better implementation guidelines and better implementations of computer integrated manufacturing.

The information model captures information at a multiple levels of abstraction across the manufacturing environment. The information is grouped into logical categories and relationships between these information categories are documented. Graphical descriptions are used to depict these relationships (information integration diagrams) and textual descriptions called standard specifications are used to define and document the entities and the information categories. The data in each particular information category is decomposed into lower levels of greater detail.

Entities in the information model are, loosely considered, the "things" that people think about in manufacturing such as a build order, a part, an assembly or a calibration of a machine. The entities identified in the information model are considered to be a prototype "thing" which has many instances or specific copies. For example, a build order entity is the concept of a build order and not a specific build order. In the information integration model, there is a single concept of a build order, called an entity, although it is know that there may be thousands of actual build orders in an organizations manufacturing environment.

An attribute of an entity is a detail associated with it. The attributes of the entity detail the entity and provide the means to identify the separate instances of an entity. For example, of the thousands of build orders in a manufacturing environment, how can a particular build order be identified? Often a entity has an attribute which is a unique identifier like a person can be identified uniquely by her social security number. For example, a build order has an assembly which it orders to be built, a unique build order number, a date which the build must be completed by, an authorization of the build order, etc. It is often unclear whether an information element (a "thing" such as an authorization) is properly an entity or an attribute or a relationship between two entities.

Relationships are the ways in which two instances of two entities may be connected to each other. For example, the attribute called the assembly that the build order referred to above orders to be built, may actually be an entity. In this case, a specific instance of a build order has a relationship to a specific instance of the entity called "assembly". This is the way the information model captures the fact that every build order (an instance of the entity "build order") is connected to an instance of an assembly (another entity in this case). While the relationship is said to be between two entities, the relationship must make clear which two instances of the entities are actually related.

Information models are abstractions of data models; the information model is a conceptual and general representation of the more implementation specific data model. An information model describes categories of data and relationships between these groups of data at a high-level of abstraction or detail. The data model is derived from the information model and describes the data in technical detail for use in a computer database. Both kinds of models are needed to address the big picture of how people think about the information used in performing their work and the details of how application programs can access that information to provide integrated services.

The information model consists of the following components:

- Information integration diagrams, which are graphical descriptions of the information categories and the relationships/interconnections between the information categories.

- Standard specifications are a textual description describing each information category on a information integration diagram and list the entities contained in the information category and describes each entity.

- The functional model's data dictionary which lists all of the data from the information flows on the functional model. The standard specification entity definitions are derived from the data dictionary.

6.2.1 Information integration diagrams

The information integration diagrams provide a graphical description of the information categories and the relationships between the information categories. Data at a particular level of abstraction is partitioned into information categories, and each information category contains a class of entities. The flows between the information categories describe the relationships between the information categories.

Due to the amount of information used within the CIM environment, attempting to develop the information model can seem overwhelming. But just as with any large project, the information model should consist of several levels of detail so that each particular level contains a coherent and manageable number of information categories that can be analyzed. This is accomplished by creating several levels of detail using the information integration diagrams. Each

information integration diagram can be decomposed into finer levels of detail by exploding an information category at the higher level diagram into an information diagram describing the entities within the information category and their relationships. This is analogous to the decomposition of the function integration diagrams, where relationships are maintained during the decomposition into sub-levels of detail.

6.2.2 Standard specifications

The information integration model's standard specification is a textual description of an information category and also describes each entity within the information category. The information category specification describes the scope or bounds of the information category, the information category criteria for data, and a list and description of the entities within the particular information category. The standard specification also lists each of the entities within the information category and how these entities relate to other entities which are external to the information category. The standard specification describes the entities within the information category, with the following details:

- the definition of each entity (a textual description of the entity),
- relationships between entities,
- attributes of the entities,
- operations on the entities,
- restrictions on the entities,
- and the relationship between the information model's entities and the functional model's data dictionary.

The relationships between entities describe information category interfaces such as how entities within the information category relate to entities which are external to the information category. Attributes are the individual pieces of information that define an entity. Finally, operations describe the ways in which the information modelled can be modified.

6.2.3 Data dictionary

The data dictionary lists all of the data from the information flows from the functional model. One problem with the data dictionary is that it gives functional definitions for the data and does not show the structure or relationships between the data. Therefore, the standard specification is derived from the data dictionary and enhances the data dictionary description of an entity by elaborating the data dictionary definition and also describing entity relationships, operations, restrictions and attributes. The relationship between the information model's standard specification and the functional model's data dictionary allows a traceability to the functional model.

7. Guidelines

This section provides a set of guidelines used in implementing a CIM system. We define the CIM system to be a set of systems (i.e., the combination of people, processes, machines and computer/control systems) that implement the manufacturing aspects of the business functions in a globally optimal manner so that overall business objectives are achieved. The first component of guidelines is a set of subsystem formulation criteria. These criteria are targeted to aid in grouping the different functions in the integration models as presented in the previous section into a number of integration domains and a number of different types of CIM subsystems. They are also useful in defining the boundaries between different CIM subsystems. The second component of guidelines is a set of subsystem selection and implementation criteria. These criteria provide recommendations on the core capabilities of the functional groups and on implementation features of the computer/control systems useful in designing or buying each of the subsystems are presented.

The guidelines presented here are specific applications of the general architectural principles defined earlier. While the principles remain general statements of direction, guidelines provide a detailed set of criteria to select or evaluate different implementations of the CIM subsystems. The guidelines are specific and detailed enough to serve as a starting point for guiding the system architecture and implementation evaluation/selection criteria definition.

These guidelines are based on current systems, available technologies, and technological solutions deemed ready for emergence in the next five years. Not all the guidelines in each category need to be followed in implementing different classes of subsystems in each category.

7.1 Subsystem formulation criteria

The first step in developing CIM systems consists of grouping the low-level functions (perhaps the second to fourth levels of decomposition) as identified by the functional architecture into logical groups or subsystems based on their interrelationships so that the overall system design is optimized. We present here a set of criteria for logically partitioning (or grouping) the functions into a set of partitions called horizontal and vertical integration domains. The criteria are used first to distribute the CIM functions in the function integration model into integration domains. Once the integration domains of interrelated functions have been established, the criteria are then used in conjunction with the architecture principles to logically group the CIM functions in each integration domain into subsystems (i.e., into application or people processes, which can be performed as logical units) and to define the boundaries between these subsystems. These criteria when applied appropriately will result in simplifying the integration of applications and CIM systems into coherent subsystems that can deliver the desired benefits of automation.

Applying the criteria to distribute and group the CIM functions is always a subjective process that changes with time as technology matures. However, the

criteria and the architecture principles provide a foundation that stays relatively constant over the changes in technology or time for making sound decisions on the number of different types of CIM subsystems and the boundaries of the applications running on them.

The criteria for partitioning and grouping functions into integration domains as well as partitioning the functions in an integration domain into different subsystems or functional groups are illustrated using the following two examples.

7.1.1 Scope or span of control or management

This criterion partitions and assigns the hierarchy of task responsibility and decision-making responsibility among the different horizontal levels of a vertical integration domain. Applying this criterion means determining the following types of information:

- The range of responsibility for how and what operation/function is initiated,
- How any problems encountered during the operation are reported,
- How a set of suboperations needed to perform a function/operation are planned or scheduled,
- Whether the function provides work direction and monitoring, and how the work direction is given,
- How the work activities are monitored,
- How the data is collected and reported.

7.1.2 Degree of work distribution

This criterion is useful for establishing vertical integration domains such as planning and scheduling or inventory management. Applying this criterion means determining the following types of information:

- How much a high-level activity is decomposed and distributed for improving manageability,
- How an activity is assigned to the associated manual or automated systems.

7.2 Subsystem selection and implementation criteria

These are technology-related criteria for selecting/evaluating computer/control subsystems that implement the CIM functions grouped into subsystems using the criteria presented in the previous subsection. The criteria are divided into four sets that are used for selecting or evaluating the following four aspects of any subsystem implementation:

- Platforms (both systems hardware and software),

- Data communications,
- Database management,
- Applications.

The following examples illustrate the above criteria.

7.2.1 Shop-floor control subsystems evaluation/selection criteria

Since shop-floor control subsystems consist of distinct component subsystems—workstations, machine controllers and supervisory controllers—we shall illustrate workstation selection criteria.

7.2.2 Workstation evaluation/selection criteria—

Platforms—

1. Workstations should either be low- to medium- resolution (up to 800 x 400) color graphics terminals or PCs; in some environments, monochrome terminals are adequate.

2. Workstations should support multiple input devices, including the keyboard, light pen and mouse as alternatives or together; they should be adaptable to accept data from automatic data capture devices.

3. Workstations should provide multiple methods of operator prompting, including screen highlights and audible alarms.

Workstation Communications—

1. Workstation communications should be done using point-to-point, or multi-access (if there are a large number of devices), twisted-pair media.

2. Workstation communications protocols should be based on RS-232, or RS-485 and HDLC or SDLC. For workstations based on PCs, a LAN connection should be supported. Device communications among I/O modules or sensors/actuators should be based on industry standards such as SP-50 and Bitbus.

3. Workstation communications link should support transfer of ASCII data and graphics data at rates of at least 9.6 kbps.

Workstation Database Management—

1. Workstation database management support should provide for presenting data to downloaded applications or PROM/RAM- resident data and for temporarily buffering (at least 2 kb) the collected data for a short time before it is sent on the communications media. For reliability reasons, up to eight hours of data should be buffered without communications to related systems.

Applications—

1. Workstation applications should work very closely with cell controller/workcenter controller applications.

2. Workstation applications should provide less than five seconds of response time for all human-initiated interactions.

3. Workstation applications should interact with the operator using menus, forms, windows and icons, and they should support external data capture devices, such as switches, automatic measuring equipment and bar-code scanners.

4. Workstation applications should support back and forth as well as go to paging of screens with a latency time of less than four seconds.

The criteria also include recommendations for the core set of operations that need to be included in the implementation of the application functionality The following are some examples of these recommendations for some applications which are part of shopfloor control subsystems..

Factory Floor Control and Monitoring

1. It should regulate the release of factory work orders consistent with the production order, current machine and labor loads, and material and tool availability.

2. It must be able to dispatch orders for all production activities in a prioritized manner.

3. It should be able to access up-to-date status of material, operations, rejects, etc.

4. It should be able to provide signals, reports or alarms for required action by people regarding maintenance of production schedules at different levels and aspects of the factory.

5. It must be able to receive instructions pertaining to various aspects of production and communicate them to appropriate applications running on the factory floor.

Detailed Factory Scheduling

1. It should be able to receive production orders from MRP and operational routings and plans to perform operational scheduling at the workcenter or factory floor level.

2. It should in scheduling consider material accumulations and movement times, queue times and lengths, and preproduction times.

3. It should present and store workloads for individual resources and make schedule adjustments in accordance with current actual workload variances.

4. It must allow flexible storage and modification of scheduling support information such as routings, rates and time standards.

5. It must be able to produce flexible reports of where used for parts, tools, data, etc.

6. It should be able to compute and present work input rates, work output rates and queue sizes and optionally signal corrective actions in case of exceptions.

7. It must provide feedback to the master scheduling system in case the schedule cannot be met.

8. It must be able to forecast production resource loads and provide that information to other applications, such as human resource management, administration and master scheduling.

8. CIM architecture use and benefits

We developed a complete reference architecture based on the definitions and applications presented in the previous section which has been used in many places within Honeywell. Already, there are reports of successful projects and results attributed to the use of an architecture approach.

A number of issues arose from application of the reference architecture. The architecture is complex and difficult to use and understand. Some kind of support environment or development tool might help in learning about and using the reference architecture. In using the architecture, there is considerable difficulty formulating an implementation procedure or methodology to develop specific architectures for specific purposes. There is a need for a methodology and architecture project guidelines. Further experience with CIM architecture approaches should help in clarifying this.

9. Conclusions

We presented in this paper our definitions of the compponents and interactions of a CIM architecture. We believe that this architecture is a means to creating and maintaining an integrated approach to a manufacturing enterprise which will reduce costs and improve customer responsiveness and competitiveness. Limited experience to date tends to support this conclusion.

Experience to date has shown a need for automated support for architecture to reduce the effort to develop architectures and a need for a methodology to build a specific architecture using our reference architecture.

References

Mikkilineni, Krishna, Bruce Clark, Theresa Jenne, Loren Krueger, HCAM Architecture 1988 Final Report, Vol. 1-4, Honeywell, 1988.

Anderson, Allan, Theresa Jenne and Krishna Mikkilineni, HCAM Architecture 1989 Final Report, Vol. 1 and 2, Honeywell, 1989.

LIST OF AUTHORS

Allan Anderson
Honeywell
Sensor and System Development Cen.
1000 Boone Avenue North
Golden Valley MN 55427

Edward Barkmeyer
NIST
Bldg. 220, Rm. A127
Gaithersburg MD 20899

Dirk Beeckman
Gap Gemini Sesa Belgium
Branch Services
Plaskylaan 144
1040 Brussels
Belgium

Frank Biemans
Senior Member of Research Staff
Philips Laboratories-Briarcliff
North American Philips Corp.
345 Scarborough Rd.
Briarcliff Manor NY 10510

H. Bohms
TNO, Netherlands Organization for
Applied Scientific Research
P. O. Box 49
2600 AA Delft
The Netherlands

R. Boykin III
CAMI
1250 E. Copeland Rd.
Suite 500
Arlington TX 76011 -8098

Dave Carlson
Dept. of Manage. Info. Systems
College of Bus. and Public. Admin.
University of Arizona
Tucson AZ 85721

Dai Chen
GRAI Laboratory of CIM & Auto.
University of Bordeaux I
351, Cours de la liberation
33405 Talence
France

I. Coutts
Department of Manufacturing Eng.
Loughborough, Leicestershire
LE11 3TU
England

Bruce Dallman
Indiana State University
School of Technology
Manufacturing and Construction
Technology
Terre Haute IN 47809

Wayne Davis
University of Illinois
Urbana-Champaign
117 Transportation Bldg.
104 S. Mathews Ave.
Urbana IL 61801

Guy Doumeingts
GRAI Laboratory of CIM & Auto.
University of Bordeaux I
351, Cours de la liberation
33405 Talence
France

John Ettlie
School of Business Admin.
University of Michigan
Ann Arbor MI 48109

Paul Fehrenbach
GEC-Marconi Research Centre
West Hanningfield Rd., Great Baddow
Chelmsford, Essex CM2 8HN
United Kingdom

Bill Foraker
Indiana State University
School of Technology
Technology Services Center
Terre Haute IN 47809

Dr. Barbara Fossum
Factorial Systems, Inc.
6300 Bridgepoint Parkway
Suite 200
Austin TX 78730

J. Gascoigne
Department of Manufacturing Eng.
Loughborough University of Tech.
Loughborough, Leicestershire
LE11 3TU
England

Cpt. Knute Hankins
Advanced Technology and
Systems Directorate
Watervliet Arsenal NY 12189

G. Harhalakis
University of Maryland
Department of Mechanical Eng.
College Park MD 20742

Larry Heath
Indiana State University
School of Technology
Electronics and Computer Technology
Terre Haute IN 47809

A. Hodgson
Loughborough University of Tech.
Department of Manufacturing Eng.
Loughborough, Leicester
LE11 3TU
England

T. Jenne
Honeywell
Sensor and System Development Cen.
1000 Boone Ave., North
Golden Valley MN 55427

Matthew Johnson
CIM Marketing and Product Develop.
Digital Equipment Corporation
MET-1/F3
600 Nickerson Rd.
Marlboro MA 55427

A. Johri
University of Maryland
Department of Mechanical Eng.
College Park MD 20742

Albert Jones
NIST
Bldg. 220, Rm. A319
Gaithersburg MD 20899

S. Joshi
Dept. of Indus. Manage. Systems. Eng.
The Pennsylvania State University
207 Hammond Building
University Park PA 16802

Robert Judd
Industrial Technology Institute
2901 Hubbard Rd.
Ann Arbor MI 48106

David Jung
Battelle Memorial Institute
5300 International Blvd.
N. Charleston SC 29418

Raymond Kapperman
Indiana State University
Industrial and Mechanical Technology
Terre Haute IN 47809

James Kirkley III
CIM Marketing and Product Develop.
DEC MET-1/F3
600 Nickerson Rd.
Marlboro MA 01752 -9917

Susan Lesko
Indiana State University
Electronic and Computer Technology
Terre Haute IN 47809

Eric Litt
Battelle Memorial Institute
5300 International Blvd.
N. Charleston SC 29418

Kai Mertins
Fraunhofer-IPK
PascalstraBe 8-9
1000 Berlin 10
West Germany

Krishna Mikkilineni
Honeywell
Sensor and System Development Cen.
1000 Boone Ave. North
Golden Valley MN 55427

I. Murgatroyd
Department of Manufacturing Eng.
Loughborough University of Tech.
Loughborough, Leicestershire
LE11 3TU England

Richard Panse
IBM Germany
Dept. 3114, Bldg. 7032-87
Am Hirnach 2
7032 Sindelfingen
Germany

Sudha Ram
Dept of Management Information Sys.
College of Business and Public Admin.
University of Arizona
Tucson AZ 85721

S. Sanoff
GEC-Marconi Research Centre
West Hanningfield Road
Great Baddow, Chelmsford
CM2 8HN
United Kingdom

John Sauter
Industrial Technology Institute
2901 Hubbard Rd.
Ann Arbor MI 48106

David Shorter
SC-Scicon plc
Centrum House
101-103 Fleet Rd.
FLEET, Hants GU13 8PD
United Kingdom

Hugh Sparks
MTS Systems Corporation
Advanced Technology Division
Box 24012
Minneapolis MN 55424

Gunter Spur
Fraunhofer-IPK
PascalstraBe 8-9
1000 Berlin 10
West Germany

Mukasa Ssemakula
University of Maryland
Department of Mechanical Eng.
College Park MD 20742

Wolfram Sussenguth
Fraunhoger-IPK
PascalstraBe 8-9
1000 Berlin 10
West Germany

S. Thompson
University of Illinois
Urbana-Champaign
Urbana IL 61801

James Ting
The University of Michigan
1205 Beal Ave.
IOE Bldg., Rm. 104
Ann Arbor MI 48109 -2117

Prof. F. Tolman
TNO, Netherlands
Organ. for Appl. Sci. Res.
P.O. Box 49
2600 AA Delft
The Netherlands

Bruno Vallespir
GRAI Laboratory of CIM & Auto.
University of Bordeaux I
351, Cours de la liberation
33405 Talence
France

Clyde Van Haren
James River Corporation
Neenah WI 54956

Raymond Vanderbok
Industrial Technology Institute
2901 Hubbard Rd.
Ann Arbor MI 48106

Johan Vesterager
Driftsteknisk Institut
Building 423
The Technical University of Denmark
DK-2800 Lyngby
Denmark

Roger Vicroy
Indiana State University
School of Technology
Manufacturing and Construction
Technology
Terre Haute IN 47809

Chris Vissers
Information Science
Twente University of Tech.
P.O. Box 217
7500 AE Enschede
The Netherlands

Roli Wendorf
Philips Laboratories
North American Philips Corp.
345 Scarborough Rd.
Briarcliff Manor NY 10510

R. Weston
Loughborough University of Tech.
Department of Manufacturing Eng.
Loughborough Leicestershire
LE11 3TU
England

Larry White
Weatherhead School of Management
Case Western University
Cleveland OH 44106

Theodore Williams
Purdue Lab. for Applied Industrial Control
Purdue University
West Lafayette IN 47907

Ron Woolsey
Indiana State University
School of Technology
Industrial and Mechanical Technology
Terre Haute IN 47809

Richard Wysk
Pennsylvania State University
College of Engineering
207 Hammon Bldg.
University Park PA 16802

James Yoder
CADS Network Development Div.
Sandia National Laboratories
1515 Eubank SE
Albuquerque NM 87185

Robert Young
Department of Industrial Eng.
North Carolina State University
Box 7906
Raleigh NC 27695 -7906

L. Zeidner
Department of Manufacturing Eng.
Boston University
44 Cummington Street
Boston MA 02215

NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NIST/SP-785

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
May 1990

4. TITLE AND SUBTITLE

PROCEEDINGS OF CIMCON'90

5. AUTHOR(S)

Albert Jones, Editor

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED
Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

Same as item #6

10. SUPPLEMENTARY NOTES

[ ] DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The CIMCON'90 Proceedings includes papers on the design and implementation of global CIM architectures. There are also papers on the design and implementation of control, database, and communications architectures.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

Computer Integrated Manufacturing; production; robotics; shop floor control; software.

13. AVAILABILITY

[X] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[X] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[ ] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
533

15. PRICE

ELECTRONIC FORM

☆ U.S.GOVERNMENT PRINTING OFFICE:1990-261-982:23423

NIST Technical Publications

Periodical

Journal of Research of the National Institute of Standards and Technology—Reports NIST research and development in those disciplines of the physical and engineering sciences in which the Institute is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Institute's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Institute's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NIST, NIST annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NIST under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NIST by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW., Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Institute on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NIST under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NIST administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NIST research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NIST publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NIST pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NIST Interagency Reports (NISTIR)—A special series of interim or final reports on work performed by NIST for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Institute of Standards and Technology
(formerly National Bureau of Standards)
Gaithersburg, MD 20899