U.S. Department
of Commerce

National Bureau
of Standards

NBSIR 86-3312
January 1986

AN ARCHITECTURE FOR DISTRIBUTED DATA MANAGEMENT IN COMPUTER INTEGRATED MANUFACTURING

NBSIR 86-3312
January 1986

AN ARCHITECTURE FOR DISTRIBUTED DATA MANAGEMENT IN COMPUTER INTEGRATED MANUFACTURING

Edward Barkmeyer and Mary Mitchell*

Krishna P Mikkilineni, Stanley Y W Su and Herman Lam**

* Factory Automation Systems Division, National Bureau of Standards, Gaithersburg, MD

**Database Research and Development Center, University of Florida, Gainsville, FL

1. INTRODUCTION

Computer Integrated Manufacturing (CIM) includes activities for supporting, planning, and controlling the manufacture of products. Our definition of a CIM system is one that is capable of supporting the total manufacturing process, from specification through production, and retaining the information generated for later analysis or reuse. The objective of CIM is to improve the aquisition and dissemination of manufacturing information and thus, improve manufacturing effectiveness and resource utilization that enables an enterprise to compete economically.

At this time, computer-aided systems and methods exist to support some manufacturing functions. Newer methods such as trial fitting of orders, net change planning and rough cut resource planning are expanding the capabilities of CIM [KOC82]. Most of these systems and methods have been designed and used to support individual manufacturing functions only. The result is a limited interaction among the different component functions of CIM. Consequently, a totally integrated manufacturing system does not yet exist.

In a recent survey of CAM [HAT83], software integration was identified as the technology with the greatest potential to make CIM a reality. To realize the benefits promised by CIM, the fully integrated manufacturing system must 1) link the activities on the shop floor with the design, planning, and manufacturing engineering functions, 2) provide an accurate and timely feedback from the factory floor, and 3) modify and replan activities in near

real-time based on the actual performance feedback.

Central to the fully integrated manufacturing system is the data administration system whose function is to manage the shared data resources throughout the design, planning and manufacturing processes. The issues of representation, integration, and administration of CAD/CAM data have been identified as some of the major problems in CIM [YOS81]. Recent efforts in manufacturing integration such as the NASA IPAD and the Air Force ICAM projects [FUL80, ICA79] have stressed the importance of data management in CIM by concentrating resources on the study and development of database management techniques.

This paper presents a distributed database management system architecture to support integrated manufacturing. The system was designed to support the Automated Manufacturing Research Facility (AMRF) under development at the National Bureau of Standards (NBS), a representative CIM effort. A prototype of this system is under development in that facility.

The remainder of this section summarizes the salient features of the NBS project. In section 2, we discuss the unusual requirements of the AMRF and the implications of these requirements on the management of data in a CIM system. Section 3 presents the overall design of a database administration system architecture for CIM, evolved from our understanding of these requirements. In section 4, we describe in some detail the major modules of this system. Section 5 presents various scenarios of the data administration system in operation, using the AMRF as an example. Section 6 summarizes the features of the proposed archi-

tecture, provides a status report and indicates the direction of future efforts.

1.1 THE AMRF AND ITS OBJECTIVES

The National Bureau of Standards Automated Manufacturing Research Facility is being developed as a testbed for automated small batch manufacturing [SIM82, MCL83]. It will be used to support experimentation in automated metrology and factory integration. The overall goals of the AMRF are: 1) to identify potential standards and interface protocols for automated manufacturing, and 2) to further measurement techniques and develop standard reference materials which provide a means of ensuring the traceability of processes and products to national standards. The specific goals of the AMRF data management effort are twofold: 1) to support the data needs of the AMRF, and 2) to identify and prototype data management interface protocols which are suitable to the CIM environment. The AMRF testbed, when completed, will be made available for selected research projects by the academia, industry, research institutions, and other government agencies. The testbed will be capable of producing a large number of part families similar to the part mix found in a typical machined-parts job shop. The production will be automated from the transfer of near net shape blanks from inventory to the delivery of finished, cleaned, and inspected parts.

The control architecture of the AMRF is shown in Figure 1. This architecture has five major levels: 1) Facility control, 2) Shop control, 3) Cell control, 4) Workstation control, and 5) Equipment control. The term "manufacturing cell" is used by some

flexible manufacturing systems and fixed automation applications to designate an equipment cluster, which the AMRF identifies as a "workstation". The AMRF architecture reserves the term "cell" for the control system coordinating workstations in part batch production. The typical workstation in the AMRF is configured with a machine tool, an industrial robot, automated material transfer areas, and an intelligent workstation controller, whose function is to coordinate equipment level operations. The control architecture utilizes hierarchical control techniques [ALB81], in which long range production goals are decomposed at the facility level into sequences of tasks, which are then assigned to a subordinate control module to manage. These tasks are further decomposed and assigned to lower level control modules; each control module takes commands from exactly one higher level module but may direct several lower level modules. The AMRF project, its architecture and design ideas, has been extensively described in the literature [BAR82, MCL83, MIT84, NAN84, MCL84, HOP83, KEN83]. From the data systems point of view, two characteristics distinguish the AMRF from other flexible manufacturing systems: the ability of the workstations to function autonomously, and the explicit separation of control data from control algorithms.

2. DATABASE MANAGEMENT REQUIREMENTS IN COMPUTER INTEGRATED MANUFACTURING

The design of a system to support the information requirements of automated manufacturing is affected by the environment in which it must operate. The following requirements, which seem generally applicable to any CIM system, have been expressed in the development of the AMRF testbed:

a) integration of heterogeneous systems;

b) component systems with different data management abilities

c) both autonomous and integrated operation of subsystems;

d) flexible manufacturing;

e) time critical operations; and

f) use of adaptive control techniques.

The main function of a database management system in this environment is to provide the component systems with access to the integrated database which contains data necessary to support the design, planning, manufacturing, assembly, inspection, and service of products. It must also provide other database management functions such as integrity and security control, concurrency control, error recovery, etc. Database management requirements for CIM follow directly from the characteristics above and can be stated as follows:

a) Integration of Heterogeneous Systems: The computing environment supporting the factories of the future will be a network of heterogeneous component systems acquired from numerous vendors. The integrated database must contain all data useful for

the control and management of the component systems and should serve as a medium through which the component systems communicate with one another. The software system which manages the integrated database should mask the differences inherent in accessing data in a heterogenous environment. A high level data manipulation language is needed to provide a standard format for all component systems to make database requests. Furthermore, data translators are required to translate the type, format, and structure of the data interchanged among the component systems. Command translators are required to translate the queries posed in the common data manipulation language into the commands which can be executed by different underlying data management software on the component systems.

b) Component Systems with Different Data Management Capabilities: The component systems in an integrated factory network typically have different data management capabilities. Some have very capable commercial database management systems, while others have only file servers and memory resident data. Distributed data management software needs to be installed on selected components and these systems can be assigned to perform the complex functions for less capable components. A query issued against the integrated database must be decomposed into database operations which are within the capabilities of the system on which the data resides. A defined set of primitive operations, a generalized method for representing the level of data manipulation ability, and a method of assigning operations to more capable systems are all needed to process requests in a uniform manner. The distributed data

management system must then be able to perform final assembly and formatting of the data retrieved from different component systems and deliver the result in the desired form.

c) Local Autonomy and Integrated Operation: Whenever new equipment or control software is introduced to the factory data network, the control and data systems must provide for the independent testing and subsequent modular integration of these elements. The same capability is required in order to reintegrate equipment which has been logically removed from the network for maintenance or refit. Note that in the case of a fully automated workstation, the "independent" test requires a separate integration of the systems within the workstation into an autonomous system separate from the integrated factory data system, and, more importantly, separate from the live production data. The data system within the independent test segment must have the functionality to acquire relevant segments of the integrated database and to manipulate this data. Therefore, an independent test segment must have at least one component system with resident distributed data management functions; and this system will support any functions which the remaining component systems cannot. The data system as a whole requires mechanisms to identify and control replication. Reintegration of formerly isolated components must occur without disrupting ongoing production. Local databases and data directories must be reconciled to resolve the inconsistencies arising from the "partitioned" operation. Thus, integrity control functions must be enforced throughout the operation of both integrated and partitioned factory network and are critical at the

time of reintegration. To make this possible, the data system must access and use configuration information during its regular operation.

d) Flexible Manufacturing: Meeting the objectives of flexible manufacturing, such as accommodating unknown production demands, effective resource utilization and modular expansion, affects the data management system in numerous ways. Modular expansion dictates support of network reconfiguration, while effective resource utilization may dictate support of dynamic control reconfiguration. From this we derive some precepts for the data system. First, the database management system must provide data to control processes in a completely transparent manner, i.e. independent of the data location, the network configuration and the current control configuration. Second, data delivery paths should not be built directly into the control structures of the component systems, but should be constructed by the database management system from the current repository to the consumer at the time that consumer's requirement for that data entity is identified. Third, the database system should allow frequent and dynamic updates to the data directory which in part contains the information about system configuration and data delivery paths.

e) Time Critical Operations: The systems that control shopfloor equipment typically operate in real-time and have time-critical requirements for access to certain data. A variety of sensory and feedback data originating in subordinate systems are used to perform control functions. It is important for this data

to be stored locally and in a form efficient for real-time operation. There is a critical set of these data resources which must be shared and the time criticality will effect the decisions on how to distribute this data. In database management terms, this means that:

- data units may have to be replicated from one component system to another;

- conversion between the local data representation and the neutral data representation is necessary and frequent; and

- a local data directory and some facility for accessing and maintaining the global directory must be provided at each component system.

f) Adaptive Control: The control system for integrated manufacturing must be able to detect and react to failures and unexpected events. The database management system must be able to:

- move data with sufficient speed for time-critical operations,

- monitor the state of the data management system and recover from a failed data system component,

- support the recovery of component systems by enforcing data consistency constraints,

- allow malfunctioning components to be logically disconnected, even though they may continue to demand data services, and

- act as a system resource to gather and retain data about the integrated operation for later analysis.

3. THE DATA MODEL AND ARCHITECTURE OF THE INTEGRATED MANUFACTURING DATABASE ADMINISTRATION SYSTEM (IMDAS)

In the CIM environment, the existing databases are already distributed across different component systems. IMDAS supports an integrated database composed of physically distributed databases. The construction and maintainance of the integrated database relies heavily on the data model used to define the data residing in the component systems.

3.1 A DATA MODEL FOR THE INTEGRATED DATABASE

The user facility which embodies the data model must balance the conflicting criteria of simplicity, logical completeness, and flexibility. It provides for the description of data definitions, interactions and mappings of the distributed databases, the declaration of integrity and security constraints, and the definition of access requirements. The data model used in CIM must especially be able to support the complex CAD/CAM data types which are not needed in business data models and must be able to maintain complex constraints and relationships. The data model selected for IMDAS is based upon the Semantic Association Model (SAM*) [SU83,SU85]. SAM* is rich in data semantics and contains a number of constructs for modelling the relationships among the data found in engineering, business, scientific and statistical databases. The SAM* model contains several semantic constructs to support the description of data partitions, constraints and mappings needed for distributed functionality.

To describe the mappings between the logically single integrated database and the numerous physically distributed

databases, three levels of view definition are supported by the IMDAS. These views are: 1) global external view, 2) global conceptual view, and 3) fragmented view. A global external view defines a segment of the integrated database as seen by a single control process. It identifies the data entities and associations used by the control process. There can be many global external views defined for the various control processes. The global conceptual view is the integration of all the external views and represents a comprised view of the factory database. The word "global" is used to distinguish these views from the external and conceptual views defined in some component DBMSs. Fragmented views represent global conceptual data which are physically partitioned or replicated across the component systems. They describe the distribution of the conceptual database; data within each component system is represented as a distinct fragmented view. Each fragmented view maps into the data representation supported by one local data management facility on one component system. We refer to the fragments of the global database which are distributed among the nodes in the system as sub-databases. The relationship among the different views is shown in Figure 2.

3.2 GLOBAL DATA MANIPULATION AND DEFINITION LANGUAGES

In the distributed database environment, a source system obtains access to data residing on a target system by issuing a database query which can be processed by the target system. One of the following two approaches is used to achieve this. The approach used by the IPAD distributed database project [FUL80] is to allow the source to issue queries in the native language of its

local DBMS. This system then translates the query into the language used by the target system and delivers the resulting query to the target system. An alternative requires all control processes to use a common global data manipulation language (GDML) to issue database queries, as in the ICAM IISS project [ICA79]. The first approach is designed to recover significant investments in applications development and training in the local data manipulation language. But it requires that every component system have the support of some local database management system and that there is a direct correspondence between the semantics of all query languages in use. These assumptions are not valid in a general CIM environment and in the AMRF in particular. For this reason, we have chosen to use the second approach in the IMDAS design. It not only eliminates the single-source-to-multiple-target translators but also provides a common query language and known semantics for all the component systems. The IMDAS GDML [KRI85] has been designed to support the SAM* data model and is strongly based on the draft proposed ANSI standard database manipulation language [ANS85].

The data model and the global data manipulation language will be discussed in more detail in subsequent papers.

3.3 DESIGN CONSIDERATIONS

There are two basic distributed database management architectures: centralized and distributed. In the centralized architecture, a single central data administration system handles all of the control and data management functions. Data can be

stored in either centralized or distributed locations but all database requests are routed to a central control module. This architecture is straightforward to develop and manage, and is being used in some partially automated factories. Performance of a centralized system depends on the number and activity level of the component systems competing for services. Centralized control, however, is not suitable for extensive CIM systems because the architecture does not meet the requirements such as predictable service times for time-critical operations, and adaptive recovery. Moreover, the single point of failure of the central system would create a serious reliability problem.

In the fully distributed architecture, every component system has a data administration system which controls and manages the distributed execution of the data management requests issued at that system and is capable of servicing requests coming from other component systems. This architecture makes the system as a whole more reliable and provides each component system with maximum autonomy. Unfortunately, fully distributed control of database integrity, concurrency, and recovery is far more difficult to achieve than in centralized control.

Most fully distributed database administration systems manage homogeneous business oriented databases on identical computer hardware. As noted in the requirements discussion, a CIM system generally contains heterogeneous databases on dissimilar component systems. Therefore, the considerations that led to the selection and use of distributed control in homogeneous distributed database management systems such as Distributed INGRES,

System R*, SDD-1, and POREL [WIL82, NEU84, CER84, BRA82] are not necessarily valid for the CIM environment. Implementing a fully distributed database administration system on every heterogeneous component system in a CIM environment is more difficult and costly than implementing it on many homogeneous systems. And, a significant overhead is involved in processing data using a fully distributed control strategy. This overhead is likely to have a negative impact on the performance of real-time operations.

The IMDAS design pursues another possibility; a hybrid architecture which is neither completely centralized nor fully distributed. The architecture of the IMDAS intends to capture the advantages of both architectures. To achieve separability, reliability and recoverability, the IMDAS minimizes centralized functions and replicates the software which implements these functions. To achieve cost effectiveness, particularly on small systems, IMDAS does not require full distributed data service capabilities on all components.

3.4 THE RELATIONSHIP BETWEEN IMDAS AND CIM SUBSYSTEMS

The design of a distributed database management system for CIM should consider the relationship among the three major topologies existing in CIM systems: the Factory Network Topology, the Control Topology, and the Distributed Database Management Topology. We shall now discuss the relationship between these three topologies in the AMRF and the IMDAS architecture.

3.4.1 FACTORY NETWORK TOPOLOGY

The factory network provides the means of communications among the component systems. As illustrated in Figure 3, the

AMRF network is logically a network of peer computer systems connected to a common high-speed bus. A boundary is forced at the workstations, within which highly time-critical traffic occurs. Each workstation and its equipment are connected to one another through a private workstation subnetwork. Each subnetwork is connected to the main factory network through a gateway. The workstation network is, therefore, also a network of peer computer systems. Thus every system on the network can access data from or provide data to any other system on the network.

3.4.2 CONTROL TOPOLOGY

There are several alternative control topologies used in CIM systems. In the AMRF, the various control systems are related to each other in a clear hierarchical structure. As depicted in Figure 1, the control topology, with the exception of the manufact-uring engineering functions, is a regular tree structure. It is important to note that most data enter the system at either the top level of the control hierarchy or the bottom. At the top, the facility level, orders, part designs and process plans are entered, while at the bottom, the equipment level, sensory information enters the system. Most of this sensory data is consumed at the equipment level or the workstation level where operation sequences are executed and judgements about the physical world are made by comparing the design and engineering information with locally acquired sensory information. Both the engineering data and most sensory data are used primarily by processes which do not originate them. The middle layers of the hierarchy tend to generate and consume intermediate data to perform coordination and

scheduling functions for the lower layers, and to provide the information useful in facility planning, cost accounting and other high level functions.

While commands and status reports follow the control hierarchy, it is neither necessary nor desirable to require most manufacturing data to do so. Forcing a hierarchical data flow causes "proxy retrievals" by component systems with no direct use for the data and induces propagation delays without providing any functionality. For example, a machining center at the equipment level may make a request for a NC Program which is maintained by a design system at the facility level. If the data were required to follow the hierarchical control chain, four transfers of the NC program would be required before the data arrived at the machining center. In the IMDAS, the data flow is not required to follow the control hierarchy. Instead, the control information identifies the needed data sets and the consuming process directly accesses the actual data.

3.4.3 DISTRIBUTED DATABASE MANAGEMENT TOPOLOGY

The IMDAS consists of three service layers, each of which is responsible for a defined set of distributed data management functions. These functions are distributed over the component systems according to their computational capabilities. The different layers of IMDAS software work together in establishing, manipulating and controlling the distributed databases. The IMDAS provides each component system with one or more classes of data management services. These classes are referred to as "basic",

"distributed", and "master".

The basic data management services - command translation, data translation and communication - are required to provide access to the data residing on every component system. These services are provided by a Basic Data Administration System (BDAS) which is implemented and installed on every component system.

The distributed database control and management functions - query translation, distributed query execution, and final result assembly - are handled by the Distributed Data Administration System (DDAS). This software resides on component systems that are powerful enough to support it. Not every system has a DDAS but every system and thus every control process has access to one.

The MDAS performs the functions of global directory management, data system initialization and data reconciliation. The "master" layer of data management services is required when groups of component systems containing more than one DDAS are integrated into an operating segment of the factory data network. For each independently operating segment, there is exactly one Master Data Administration System.

The IMDAS architecture is formed by a three-level hierarchy of BDAS, DDAS, and MDAS modules, as shown in figure 4. Each module in the hierarchy controls access to data within its domain, that is, within the entire sub-tree rooted by that module. Database management operations are confined to the domain of a module if they can be performed by that module. Otherwise, they are passed upward to the next higher level module for processing. This architecture and processing strategy allow most database functions to

be carried out locally (i.e. within a module's domain) without causing unnecessary inter-processor communication, synchronization and data transfers.

As illustrated in figure 5, the integrated database, which is physically distributed over the different component systems, together with the BDAS, DDAS, and MDAS modules of the IMDAS serves as a central core through which all the CIM subsystems interact and communicate.

Figure 6 illustrates the relationship among the IMDAS topology, the network topology and the control topology, using the AMRF as an example. In this figure, two independently operating segments of the factory are shown with the three types of topological connections among the component systems.

4. FUNCTIONAL DESCRIPTION OF IMDAS MODULES

This section describes the modular organization and functions of the three subsystems constituting the IMDAS: the Basic Data Administration System, the Distributed Data Administration System, and the Master Data Administration System.

4.1 BASIC DATA ADMINSTRATION SYSTEM (BDAS)

The BDAS provides a uniform interface between the data residing on each component system and the integrated database. There is therefore a BDAS software module resident on every component system in the automated factory. Since the component computer systems have widely different processing capabilities, the BDAS provides only the essential data management and communication functions. These functions may be implemented differently in each component system to support the unique hardware and software environment of the system. The functions of the BDAS are interprocess communication, network communication, data translation, command translation, and data management. The architecture of the BDAS is shown in Figure 7.

4.1.1 INTERPROCESS COMMUNICATION

If the component system houses multiple control and sensory processes, or if the communication and data management services are implemented as separate processes, a mechanism for communication among these processes is needed. Interprocess communication can be achieved either by a direct process-to-process message passing or by using a shared memory, depending on the capabilities of the local operating system.

The AMRF uses a shared memory scheme [MIT84], in which an

originating process stores information of a particular kind or for a particular recipient into a designated area of the shared memory, and a retrieving process interested in a particular information set fetches it from the preassigned area. This scheme has been implemented on different systems in the AMRF in different methods - using a true hardware shared memory, using message-passing to a memory-manager process, and using a process which copies the information between the local memory of each control process and a background common memory [BAR82]. The shared memory approach permits data to be used by more than one process without explicit action by the originator to deliver it to all users. The shared memory can be considered as a database in its own right, as well as a data communication vehicle between the control processes and the data services. The shared memory approach is particularly effective in equipment level systems which must perform real-time data acquisition.

4.1.2 NETWORK COMMUNICATION

Every component system must have access to the factory data network and must implement the protocols necessary to communicate with the other component systems. The IMDAS architecture presumes that the factory data network utilizes a bus or ring local area network topology, thereby providing a peer relationship between all component systems. The four lowest layers of the ISO/OSI reference model [ISO81] must be implemented for each component system, either within the system or with network appliances. These basic protocols allow reliable communication between any two systems on the network.

In the AMRF, the network communication function is performed by a Network Interface Process (NIP) in each component [MIT94]. The NIP maps the needed areas of a conceptual "global" shared memory into the local shared memory. The mapping is done by replicating the contents of its local shared memory into the shared memory areas of the remote components which require the copies of that data. The NIP is table driven, so that the mappings can be dynamically modified by creating or destroying entries in a "delivery table". The advantage of this global shared memory scheme is that it is a distributed database mechanism which can both contribute to and profit from other database techniques used in the IMDAS.

4.1.3 DATA TRANSLATION

Whenever data are to be moved from one system to another, it is necessary to translate the data from the source representation to the required target representation. The data translation process typically involves the translation of data type, structure, and format of the data. This translation can be performed by either direct source-to-target conversion or source-to-common-to-target conversion. We use the latter, since it requires substantially fewer translators at each component system, one local-to-common and one common-to-local, as opposed to (N-1) local-to-target translators for each of the N systems. Using this approach, the data translator converts the data from the local representation (data type, format, and structure) to a "common" representation on "outbound" sessions and from the "common" representation to the local representation on "inbound" sessions.

For each "session", the data translator must be given the syntax of the messages for both the local and common representations.

4.1.4 COMMAND TRANSLATION

In IMDAS, a global data manipulation language (GDML) query issued by a component system is translated into a tree of standard primitive operations by a DDAS. The function of command translation is to take each operation in standard form and translate it into the query language or access mechanism understood by the underlying physical data management tool, which may be a data manager or file manager. Command translation can often be table driven although some particularly intractable data structures may require preprogrammed procedures.

4.1.5 DATA MANAGEMENT

The BDAS in each component system must provide, through the local database or file management system, the actual access to and manipulation of the local databases. The database manipulation and management capabilities can vary substantially from one system to another, depending on the facilities provided by the software. In small control systems, the local database may be implemented only in the main memory and managed by a shared memory management process. In others, a file manager may meet all local requirements, giving the system little more than "read" and "write" operations. In larger systems, sophisticated database management systems may provide the actual database services. Moreover, there may be more than one such facility on a single component system.

The sequence of primitive operation commands given to a

BDAS command translator must be within the capabilities of the underlying data management system which manages the data. The data manager must also be capable of cooperating with the other data managers in controlling and maintaining consistency, concurrency, and recovery. The minimal data manipulation capabilities of a BDAS are therefore "read file" and "write file", and the minimal control capabilities are "lock file", and "unlock file".

4.1.6 BASIC SERVICE EXECUTIVE

The function of this element is to provide a single point of contact between the DDAS and the BDAS data services. The basic service executive accepts DDAS commands, routing them to the BDAS modules which perform the command translation, data manipulation, and data translation functions. It also constructs local data delivery paths, sequences the execution of query tree operations, and reports the completion or error status back to the DDAS transaction manager.

The Basic Service Executive, because it communicates with the DDAS and with its peer BSEs must have a standard set of interfaces. The interfaces to the subordinate BDAS modules are to some extent determined by the local system conventions.

4.2 DISTRIBUTED DATA ADMINISTRATION SYSTEM (DDAS)

The DDAS provides control and user processes (its users) with uniform access to the integrated database. Each DDAS is responsible for providing data management services to every process within the component systems assigned to it. It further provides and controls the distributed database management

functions for all the data maintained by the BDASs on these component systems. In each separable group of component systems i.e. each group capable of independent operation, at least one system must contain a DDAS. The major modules of the DDAS are: the distributed service executive, the data manipulation language service, the query mapping service, the transaction manager, the data directory service, and the data assembly process. Figure 3 illustrates the interrelationship among these modules.

4.2.1 DISTRIBUTED SERVICE EXECUTIVE

This module oversees all DDAS activities and provides the single point of contact between the data system and some set of users and between the DDAS and the MDAS. It is responsible for initialization of the DDAS and its subordinate BDASs and for coordination of all DDAS functions.

Whenever a DDAS is brought up, it must initialize its data directory to reflect only the data which is resident in the local BDAS. It then creates connections to its subordinate remote BDASs, requests and incorporates their data directory information. (Directory entries of a subordinate BDAS data or file manager are not accessible if the subordinate is down and must be integrated with the others when it comes up.) The result of the integration is a conceptual view of the entire sub-database constituting the domain of the DDAS. This view represents a comprised view of the BDAS sub-databases, in which all conflicts and inconsistencies existing in the BDAS directories are resolved. The DSE then waits for a connection request from the appointed MDAS. Upon connec-

tion, the MDAS requests, and the DSE transmits, the resolved DDAS directory for integration into the Master directory, containing the definition of the global conceptual view and the mapping to the fragmented views of the participating DDASs. The DDAS would at this time execute any MDAS-issued transactions for recovery or integrity purposes to bring the global database to a consistent state. Finally, the DSE initializes the interfaces to the control processes served by the DDAS and is then ready to begin accepting requests for service.

User processes running on the DDAS component systems issue data management transactions (often called "queries", even though they may be update requests) through the local or network interprocess communication services. Upon receipt of a transaction from a user process, the DSE routes the transaction to the Data Manipulation Language Service, and then determines whether the transaction can be locally executed. If it can, the DSE passes the transaction to the Query Mapping Service and then on to the Transaction Manager for execution. If the transaction cannot be locally executed, the DSE sends it to the MDAS for service. When execution of the transaction is complete, the DSE reports the completion status to the originating user.

The DSE must also service transactions originating from other DDAS domains which are passed to this DDAS by the MDAS Transaction Manager. The DSE routes such transactions to the Query Mapping Service and the local Transaction Manager and reports completion to the MDAS Transaction Manager.

4.2.2 DATA MANIPULATION LANGUAGE (DML) SERVICE

The DDAS receives transactions expressed in the GDML from the user or control processes within its domain. A GDML query is issued against the global external view of the integrated database by the user process. To process the query, IMDAS can either materialize that data defined in the external view and then perform the query operations on the data, or modify the query tree so that the query operations will operate on the data defined in the global conceptual view. We chose the query modification approach for its flexibility and efficiency.

The DML Server parses the GDML query into a tree of primitive operations on the external view (an extended set of relational algebraic operations) and then modifies the query tree to incorporate the mapping from the external view to the global conceptual view. In addition, the DML server encodes into the query tree data delivery operations, which reference the user-designated source or destination data areas. The query tree may be then further modified to incorporate security constraints defined as part of the global conceptual view. The modified query tree now constitutes a transaction on the global conceptual view, representing the user's request modified by the constraints on that user.

The DML Server then identifies whether the requested operations can be performed entirely by this DDAS, by examining the leaves of the query tree (which represent the referenced relations and attributes) and consulting its directory to see if the data referenced by the query reside in its component systems. If any of

the data referenced are out of this DDAS domain, the query in its translated form is flagged for transmission to the MDAS for further processing. Otherwise, the query is flagged for processing by the local Query Mapping Service.

During the above operations, the DML server makes use of view and constraint mapping information and data locality information provided by the local Data Directory Service.

4.2.3 QUERY MAPPING SERVICE (QMS)

This service accepts from the DSE query trees, as translated by the DML Server, which can be satisfied entirely within the sub-databases of this DDAS. These queries may have come from local users or from remote users via the MDAS. The Query Mapping Service decomposes each query into one or more queries on the fragmented views reflecting the partition of data across subordinate BDAS databases and the capabilities of the data servers managing those databases. At this time, integrity constraints defined in the mapping from the global conceptual view to the fragmented views are incorporated into the query tree. The resulting query tree represents a transaction whose execution will guarantee that the distributed database will be in a consistent state. The query tree is then restructured and optimized so that the query tree consists of an optimum sequence of operations. The optimized query tree is organized into subquery trees, each of which can be executed by a single subordinate BDAS command translator. The intermediate and final data assembly of BDAS results constitutes another subtree, which is assigned to the data assembly service in the DDAS. The subquery trees, including inter-

mediate and final delivery operations, and precedence relationships, are now passed to the transaction manager for execution. During the query modification and optimization steps above, the QMS uses data locality information, profile information (the amount of data at each node) and BDAS command translator capability information provided by the local Data Directory Service.

4.2.4 TRANSACTION MANAGER (TM)

The transaction manager performs the control and management of distributed query execution, including enforcement of database integrity, concurrency control and recovery. The integrity control mechanism of the TM realizes a serial execution schedule for each transaction received from the Query Mapping Server, while insuring that the transaction is processed as an atomic unit of recovery. The recovery facility of the TM brings the database back to a consistent state after a system failure or a violation of the integrity constraints has been detected. The TM performs the transaction and database recovery by selectively delegating recovery actions to the BDASs, by properly maintaining distributed checkpoints, and by directing parallel activities of the BDASs using the two-phase commit protocol, as necessary. The concurrency control mechanism detects conflicts between active and pending transactions by identification of common attribute references. Pending transactions which would conflict with active transactions are deferred, held in the TM and released when the conflicting transaction completes.

When a transaction is released, the TM transmits the proper sub-trees of the query tree, along with the directions for con-

structing delivery paths, to the proper BDASs (BSEs), using local or network communication services, depending on where the BDAS resides. The TM oversees the distributed execution of these operations, receiving status reports from all of the BDASs involved. When the final results of the transaction have been delivered to the requesting process or when the data is properly updated, the TM identifies the transaction as completed.

Transaction management in IMDAS must support not only transactions characteristic of conventional database environments but also transactions characteristic of the real-time environment, which control and reflect actual physical manufacturing operations. Such transactions reflect the state and location of objects in the factory and affect the perceptions and decisions of other control processes. They include simultaneous updates from tactile and visual sensors when a part is grasped or simultaneous updates to coordinate the actions of a workstation controller, a robot, and a machine tool when a part is being fixtured. In order to achieve adequate performance, the IMDAS TM and the subordinate BSEs support the automated replication scheme describe in an earlier paper. [MIT84]. The use of multiple techniques for managing concurrency and replication is under study for maintaining data integrity and its correspondence to physical reality.

4.2.5 DATA DIRECTORY SERVICE (DDS)

The DDAS data directory is built from the directory information provided by the subordinate BDASs when they are integrated. It includes the following metadata:

- the schema and mapping information for the global conceptual and fragmented views of data residing on subordinate BDASs,

- the mapping information for the external and conceptual views of the data referenced by the control processes which this DDAS supports,

- the security constraints associated with the views of the conceptual database referenced by the local control processes,

- the integrity constraints associated with the data in subordinate BDASs,

- the profile information on the collections of data at the BDASs,

- the data delivery information required by the network and interprocess communication functions to construct delivery paths,

- the information representing the capability of each subordinate DBMS or Command Translator.

When requested by the DML server or the QMS or the TM, the Data Directory Server performs a metadata lookup and provides data location, structure, and delivery information.

4.2.6 DATA ASSEMBLY SERVICE (DAS)

This module performs selection, join, union, intersection, merging, sorting, and other final amalgamations of data retrieved from subordinate BDASs, and multiple projections of user-provided data for distribution to subordinate BDASs for updates. This function is required in the DDAS to provide these services for BDASs which have only limited data management capabilities. It is also needed when data are retrieved from multiple sources. This process assembles the data and, in some cases, reconciles BDAS

data representations with the global conceptual representations of those data.

4.3 MASTER DATA ADMINISTRATION SYSTEM (MDAS)

The function of the MDAS is to coordinate the activities of multiple DDASs. This coordination consists primarily of managing the master data directory, resolving concurrency problems among DDASs, and directing initialization, integration and recovery. The software needed to accomplish this function is resident in every DDAS. The MDAS functions are accomplished using software modules largely identical to the DDAS modules; differences are concentrated in the construction and referencing of a master directory containing the data definition of the global conceptual view and the fragmentation mappings to the individual DDASs. Therefore, the MDAS represents another instantiation of the DDAS software, which is comprised of a query mapping service, a transaction manager, a data directory service, and a data assembly process; and provides services from the master directory perspective. The unique MDAS software element is the Master Service Executive, which oversees global integration, reconciliation and recovery.

The Master Service Executive treats the DDASs as its users, receiving transactions in the form of query trees referencing the global conceptual view, instead of GDML referencing user external views. It routes them to its QMS, which maps the global query tree onto a set of fragmented views, each representing that portion of the global conceptual database maintained by a single DDAS. The MDAS QMS is unaware of the partitioning and mapping of this database into the sub-databases supported by the DDASs. It

sees the DDASs as a set of uniform subordinates, each of which has the full spectrum of data management capabilities, and each of which manages a fragment of the global conceptual database. The fragmented view queries delivered by the QMS are then sent to the Master Transaction Manager, which distributes the fragments to the appropriate DDASs and oversees the execution of the query. When the transaction is completed, the MDAS TM reports the completion to the MSE. The MSE sends the report to the originating DDAS, who is responsible for the communication with the originating user. As in the case of all DDAS operated queries, the data delivery operations are contained in the query tree and the final data delivery is direct from the system which executes the root operation in the query tree to the designated area on the user's system.

While the Master TM can use the conflict detection mechanism employed by the DDAS TMs, this mechanism, which successfully avoids concurrency problems at the BDAS level, cannot guarantee the avoidance of concurrency problems at the DDAS level, because the DDAS may be executing transactions of local origin, about which the MDAS has no information. For this reason, the MDAS TM must employ a cooperative concurrency control protocol on every transaction which is fragmented to more than one DDAS.

In the integrated factory, there must be exactly one MDAS. When the data system is initialized, an operator designates one DDAS as the MDAS and directs it to integrate the other DDASs. The DDAS services at the MDAS site continue to function as for any other DDAS. If an MDAS goes down, an operator designates one of the remaining DDASs in the partition as the new MDAS. The new

MDAS must rebuild the master directory and initiate a recovery procedure. When the data system is partitioned to allow a number of sub-networks to run independently, an MDAS is established for each partition containing more than one DDAS. Each MDAS requests its subordinate DDASs to provide their directories and integrates them to form its own master directory. When the sub-networks are subsequently reintegrated into a single data system, the MDAS uses the recorded information to reconcile the sub-databases.

5. SCENARIO

To illustrate the operations of the IMDAS modules and their functions, we will present a discussion of two scenarioes which use the AMRF testbed to describe the interaction between modules. Scenario 1 illustrates the servicing of a query from one component system which accesses the data stored at another component system. In this scenario, the query is issued by a workstation control system for retrieving the work-in-process data from a robot control system (RCS). Scenario 2 describes the processing of a GDML query issued from an operator interface for the workstation status data which exists at more than one workstation in the AMRF.

SCENARIO 1: Retrieval of Data in a Single Component System by a Remote System

In this case, a workstation control system requests the part types and counts of all the workpieces in a buffer area maintained by a local robot control system (RCS). The workstation will make its request using the GDML. The following is a description of the different sets of operations in the scenario which represent the actions taken by the IMDAS to process this query:

Set 1: 1) The Workstation control program writes the query to its data service command area. 2) The IMDAS communication system at the workstation forwards the new query to its controlling DDAS by network or local interprocess communication, depending on the location of the DDAS.

Set 2: 1) The DDAS Distributed Service Executive receives

the request and passes it to the DML Server. The DML server parses the query, references the directory to map the workstation controller view into the global conceptual view, looks up the referenced entities, builds a query tree on the GC view, determines that the query can be done by this DDAS and reports to the DSE. 2) The DSE passes the query tree to the query mapping server and the QMS modifies the query tree into a query tree consisting of operations that the BDAS on the robot controller can execute on its data. During this process, QMS interacts with the directory server to perform the mapping to the view of the data held by the Robot Controller BDAS and then reports to the DSE. 3) DSE gives the resulting query tree, including the associated delivery information, to the transaction manager. The transaction manager checks whether there is any update of the workpiece data in progress. 4) Based on the existence and progress of the update, the transaction manager schedules the current query operations so that they will not conflict with updates. 5) The transaction manager then delivers the query operations, their schedule, and the delivery information to the BDAS at the Robot Controller.

Set 3: 1) The basic service executive in the BDAS at the RCS receives the command from the DDAS to execute the query. It constructs the data delivery path back to the workstation controller, according to the delivery operations in the query, and passes the operation tree to the command translator for the DBMS that manages the workpiece data. 2) The command translator translates the global primitives into local retrieval commands executable by the DBMS and interacts with the DBMS to obtain the data. 3) The

data translator converts the data into the IMDAS standard representation and delivers it to the Network Interface Process. 4) The NIP copies the data in standard form to the peer NIP at the workstation and reports the delivery to the BSE. 5) The Basic Service Executive reports the completion status to the DDAS transaction manager.

Set 4: 1) The workstation network interface process delivers the workpiece data through the inter-process communication service to the requesting control process. 2) The TM at the DDAS reports completion of the transaction to the Distributed Service Executive, which provides the final status report to the requesting workstation control process.

SCENARIO 2: Retrieval of the Data Distributed at Multiple Component Systems

In this scenario, a GDML query is issued by an operator interface to all the workstations requesting the workstation identification, the identification of the current job in process and the expected time to complete the current task. The data requested by this query are horizontally distributed over the different workstation status reports throughout the factory. The following sets of operations occur in the IMDAS to retrieve the data:

Set 1: 1) The operator interface program writes the retrieval transaction into its data service command area. 2) The BDAS at the operator station recognizes that a new request exists

and delivers it to the DDAS, in this case a local process, by inter-process communication.

Set 2: 1) The DDAS DSE receives the request and passes it to the DML Server. 2) The DML server parses the request and obtains the mapping and location information from the Directory service. The directory indicates that some of the referenced data is not local. The DML Server reports to the DSE that the query requires the MDAS. 3) The DSE forwards the query tree to the MDAS via the network.

Set 3: 1) The MDAS Master Service Executive receives the query tree. It then obtains the description and location information from its directory server, which indicate that the data reside in multiple DDASs. 2) The query mapping server, using the master directory and the directory server performs the decomposition of the query into a query tree of primitive operations to be performed by each of the workstation DDASs, plus a sequence of operations to assemble the distributed results and deliver the final result to the operator interface. 3) The MDAS transaction manager receives these sub-query trees and the delivery information. It then checks the current query operations for conflicts with the workstation status data updates that might be in progress at the different workstations. 4) The transaction manager then delivers the query tree of operations, including delivery information requesting return of results to the MDAS assembly process, to each of the workstation DDASs. The TM also sends the assembly and final delivery operations to the MDAS data assembly service.

Set 4: 1) The Distributed Service Executive at each worksta-

tion DDAS receives its portion of the query tree from the MDAS and passes it to the local Query Mapping Service. 2) The QMS assigns the primitive operations to the underlying BDASs according to the physical locations of the data to be processed by the primitive operations. 3) The transaction managers at these DDASs then schedule these primitives for execution and send these primitive operation commands and the delivery information (pointing to the MDAS assembly area) to the proper underlying BDAS.

Set 5: 1) The BDASs at the workstations with the data create the delivery path to the MDAS assembly area, retrieve and convert the data using command translation and data translation as in the first scenario. 2) The network communications services of the BDASs deliver the retrieved data in standard form to the network communication service at the MDAS station, which delivers it to the data assembly service of the MDAS. 3) The assigned operation completion status is reported by the basic service executives in the BDASs to their DDAS transaction managers. 4) Each DDAS transaction manager, upon completion of its subtree, reports the completion, via the DSE, to the MDAS Transaction Manager.

Set 6: 1) The data assembly service of the MDAS receives the intermediate data from the different workstation BDASs and performs the data assembly function. 2) The assembled data in final form is delivered, via network services at the MDAS and operator interface stations, from the MDAS assembly service to the area designated by the originating operator interface process. 3) The MDAS assembly service reports completion of its assignment to

the MDAS TM, and the TM reports completion of the whole transaction to the Master Service Executive. The MSE in turn reports completion to the DDAS from which it received the request, namely the DDAS servicing the operator interface, and the DSE there reports final completion of the transaction to the operator interface process.

6. SUMMARY, STATUS AND FUTURE DIRECTIONS

The development of distributed database management techniques is of fundamental importance in realizing a totally integrated and automated factory of the future. The characteristics of the CIM environment introduce database management requirements which are quite different from those of other applications. In this paper, we have identified these requirements, and based on them we have designed an Integrated Manufacturing Data Administration System (IMDAS).

The IMDAS architecture addresses a hetergeneous distributed database environment for managing manufacturing design, planning, and control databases. It consists of a three-level hierarchy of data management subsystems, the Basic, Distributed, and Master Data Administration Service modules (BDAS, DDAS, and MDAS), which are distributed over the component systems of the automated factory. A BDAS exists on every component system which can provide the basic data management and communication capabilities necessary to process data residing on that system. The BDAS modules resolve differences between their component systems and the "IMDAS standard form". Every BDAS is supervised by exactly one DDAS. A DDAS, through its subordinate BDASs, manages all elements of the integrated database residing within its domain, and provides all data manipulation and data directory services on these elements. The DDAS also provides the control processes within its domain with a uniform interface to the integrated database. Typically a DDAS resides on major component systems in the factory. The MDAS handles the scheduling and

arbitration of transactions requiring the coordination of multiple DDASs, manages a global data directory, reconciles the DDAS directories, and controls the recovery of the data distributed across all the DDAS domains. In every operating partition of the factory, there is exactly one MDAS. But MDAS functions are actually DDAS functions that operate on the master directory instead of the DDAS directory, so that any DDAS can be designated as the MDAS.

The IMDAS is being implemented in the NBS AMRF testbed in order to demonstrate the feasibility of this design and to provide a heterogeneous distributed database prototype for performing further research in data systems to support the factories of the future. At the time of this writing, most of the BDAS functions have been implemented and tested on several systems including a DEC VAX*, a Sun workstation and other MC68000-based systems, and Intel Multibus-based systems. Considerable effort has been put into identifying uniform interfaces between the various modules. In the encoding of query trees and responses, a number of application specific types have been declared using the ASN.1. Some of the DDAS functions, such as the DML Server, are operational and others are still under implementation. The goal is to have one DDAS operational by late spring of 1986 and the second shortly thereafter on the VAX and SUN systems with a globally replicated data directory. There-

* Commercial products are identified for descriptive purposes. Such identification does not imply a recommendation or endorsement by the National Bureau of Standards.

after, a third DDAS software set will be implemented on an IBM 4300 series system and the first MDAS prototype will be attempted.

There are a number of areas requiring further research in the IMDAS framework. Proper techniques for initialization, recovery and reconfiguration of the factory data system are critical to the successful support of future CIM environments. There needs to be a detailed investigation of these techniques in the context of their effect on factory databases.

Factory data operations often create or reflect changes in the physical world. The physical changes, however, have dramatically different time-scale from their logical counterparts. This presents special problems in integrity and concurrency control in the factory databases. Coordination and synchronization of distributed transaction management functions, and the proper cooperation between the data administration system and the control systems are key research areas.

The database query primitives that can be executed by capable systems are fairly clear (e.g. relational algebra operations). However, the primitives to support systems with limited capabilities and the interaction with more capable systems need to be investigated further.

Classifying data into different classes and designing proper techniques for these different classes can simplify the various database control techniques and can provide better data management performance. An intelligent classification of the data by

its data service requirements and the design, organization, and placement of the integrity and concurrency control mechanisms for these different classes require more experience with the nature and use of the CAD/CAM data.

All of these areas, and probably others yet to be discovered, are candidates for future research in data systems for factory automation. Construction of the prototype IMDAS, like construction of the AMRF itself, is only the first step in a program of research supporting the factory of the future.

REFERENCES

[ALB81] Albus, J. S., Barbera, A. J., Nagel, R. N., "Theory and Practice of Hierarchical Control," Proceedings Twenty Third IEEE Computer Society International Conference, Septemeber 1981, 18-39.

[ANS85] "Draft Proposed American National Standard Database Language SQL," American National Standards Institute, Inc., ANSI BSR X3.135-198x, February 1985.

[BAR82] Barbera, A. J., Fitzgerald, M. L., Albus, J. S., "Concepts for a Real-Time Sensory Interactive Control System Architecture," Proceedings of the Fourteenth Southeastern Symposium on Systems Theory, April 1982.

[BER80] Bernestein, P. A., et. al., "Introduction to a System for Distributed Databases (SDD-1)," ACM TODS, Vol. 5, No. 1, May 1980.

[BRA82] Bray, O.H., "Distributed Database Management Systems," Lexington Books, 1982.

[CER84] Ceri, S., and Pelagatti, G., "Distributed Databases: Principles and Systems," Mc Graw Hill Co., 1984.

[FUL80] Fulleton, G., " Integrated Program for Aerospace Vehicle Design(IPAD)," NASA TM-81874, 1980.

[HAT83] Hatvary, J., Merchant, M. E., Rathmill, K., and Yoshikawa, H., World Survey of CAM, Butterworth & Co. Ltd., Kent, UK, 1983.

[HOP83] Hopp, T., and Lau, K., "A Hierarchical Model-Based Control System for Inspection," American Society for Testing and Materials, May 1983.

[ICA79] ICAM Program Prospectus, Air Force Systems Command, Wright-Patterson Air Force Base, OH, USA, September 1979.

[ISO81] "Data Processing - Open Systems Interconnection - Basic Reference Model," ISO standard 7498, International Standards Organization, Geneva, 1981.

[KEN83] Kent, E., Albus, J., Mansbach, P., Nashman, M., Palombo, L., Rutkowski, W., Wheatley, T., and Shneier, M., "Robot Sensing for a Hierarchical Control System," Proceedings of Thirteenth International Symposium on Industrial Robots/Robots 7, Chicago, IL, April 1983.

[KOC82] Kochhar, A., "Advances in Computer Software Packaages for Manufacturing Control," Proceedings of Fourth IFAC/IFIP Symposium.

[KRI85] Krishnamurthy, P., "A Data Minpulation Language for the Semantic Association Model, SAM*," Masters Thesis, College of

Engineering, University of Flordia, Gainsville, FL, 1985.

[MCL83] McLean, C. R., Mitchell, M., and Barkmeyer, E., "A Distributed Computing Architecture for Small Batch Manufacturing Systems," IEEE Spectrum, May 1983.

[MCL84] McLean, C. R., "Process Planning for Distributed Manufacturing Control," Proceedings of IEEE Conference on Decision and Control, Las Vagas, NV, Dec 12-14, 1984.

[MIT84] Mitchell, M. and Barkmeyer, E., "Data Distribution in the NBS Automated Manufacturing Research Facility," Proceedings of the National Symposium on Advances in Distributed Data Base Management for CAD/CAM, NASA Publication 2301, April 1984.

[NAN84] Nanzetta, P., "Update: NBS Research Facility Addresses Problems in Setups for Small Batch Manufacturing," Industrial Engineering, June 1984, pp. 68-73.

[NEU84] Neuhold, E. J., "Distributed Database Systems with Special Emphasis Toward POREL," IPAD II, Proc. of a National Symposium, NASA Conference publication, 2301.

[SIM82] Simpson, J. A., Hocken, R. J., and Albus, J. S., "The Automated Manufacturing Research Facility of the National Bureau of Standards," Journal of Manufacturing Systems, Vol. 1, No. 1, 1982.

[STO77] Stonebraker, M., and Neuhold, E., "A Distributed Database Version of INGRES," Proceedings of 1977 Berkely Workshop on Distributed Data Management and Computer Networks.

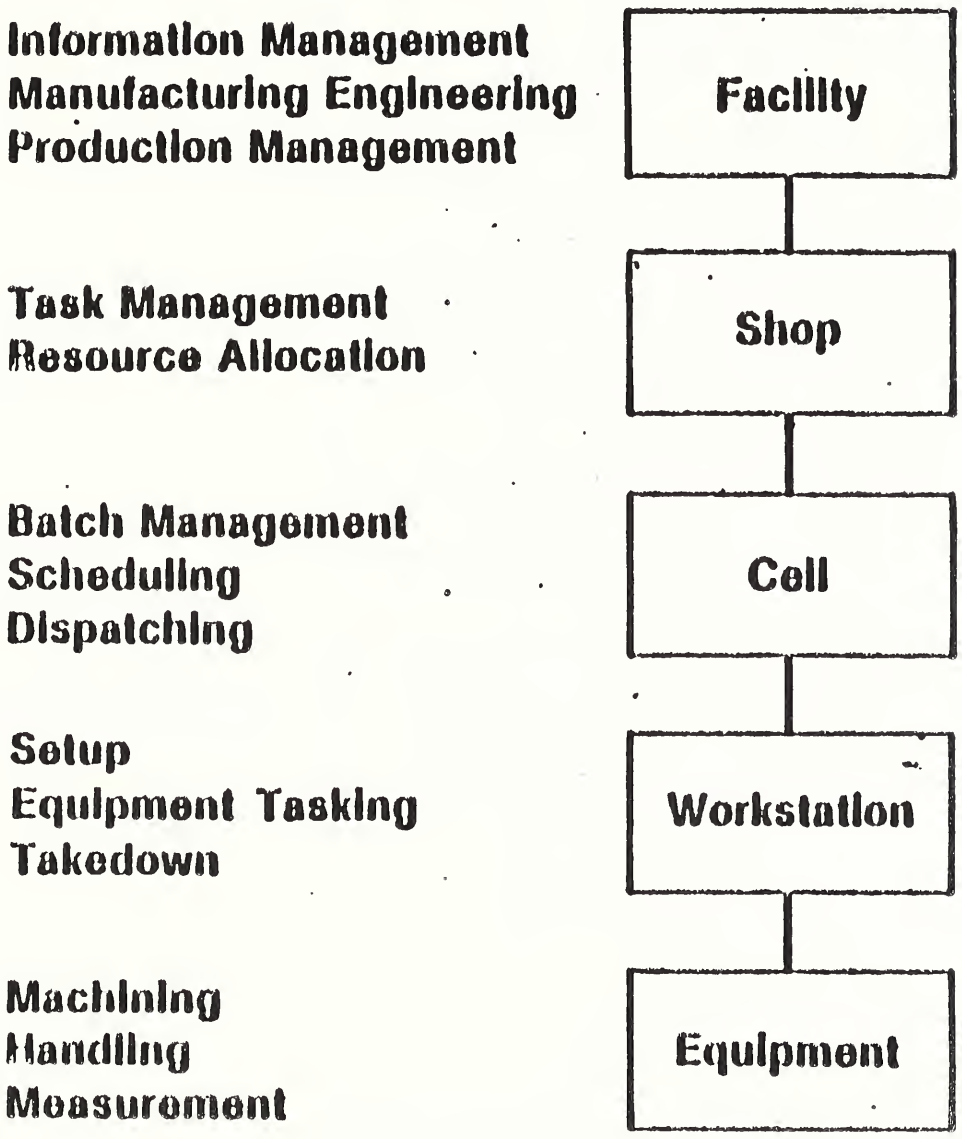
[SU83] Su, Stanley "SAM*: A Semantic Association Model for Corporate and Scientific-Statistical Databases," Information Sciences, Vol. 29, 1983 pp. 151-199.

[SU85] Su, Stanley "Modeling Integrated Manufacturing Data using SAM*," Proceedings on Database Systems for the Office, Engineering and Science," Karsruhe, West Germany, Mar 20-22, 1985.

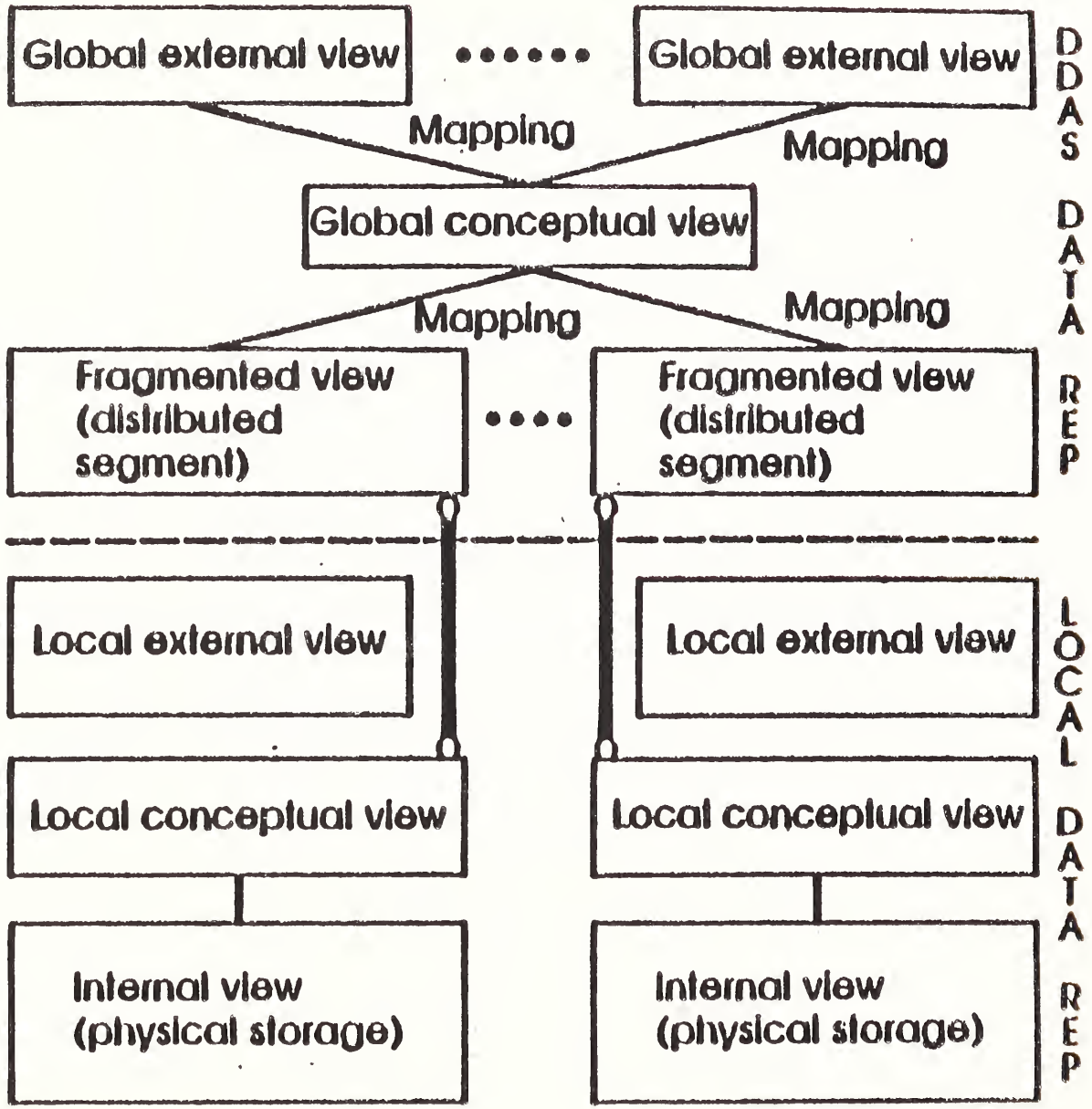
[WIL82] Williams, R et. al., "R* : An Overview of the Architecture," Proceedings of the Intl. Conference on Database Systems, Jerusalem, June 1982.

[YOS81] Yoshikawa, H., Rathmill, K., and Hatvany, J., "Computer-Aided Manufacturing: An International Comparison," National Academy Press, Washington, DC, 1981.

The AMRF Control Hierarchy



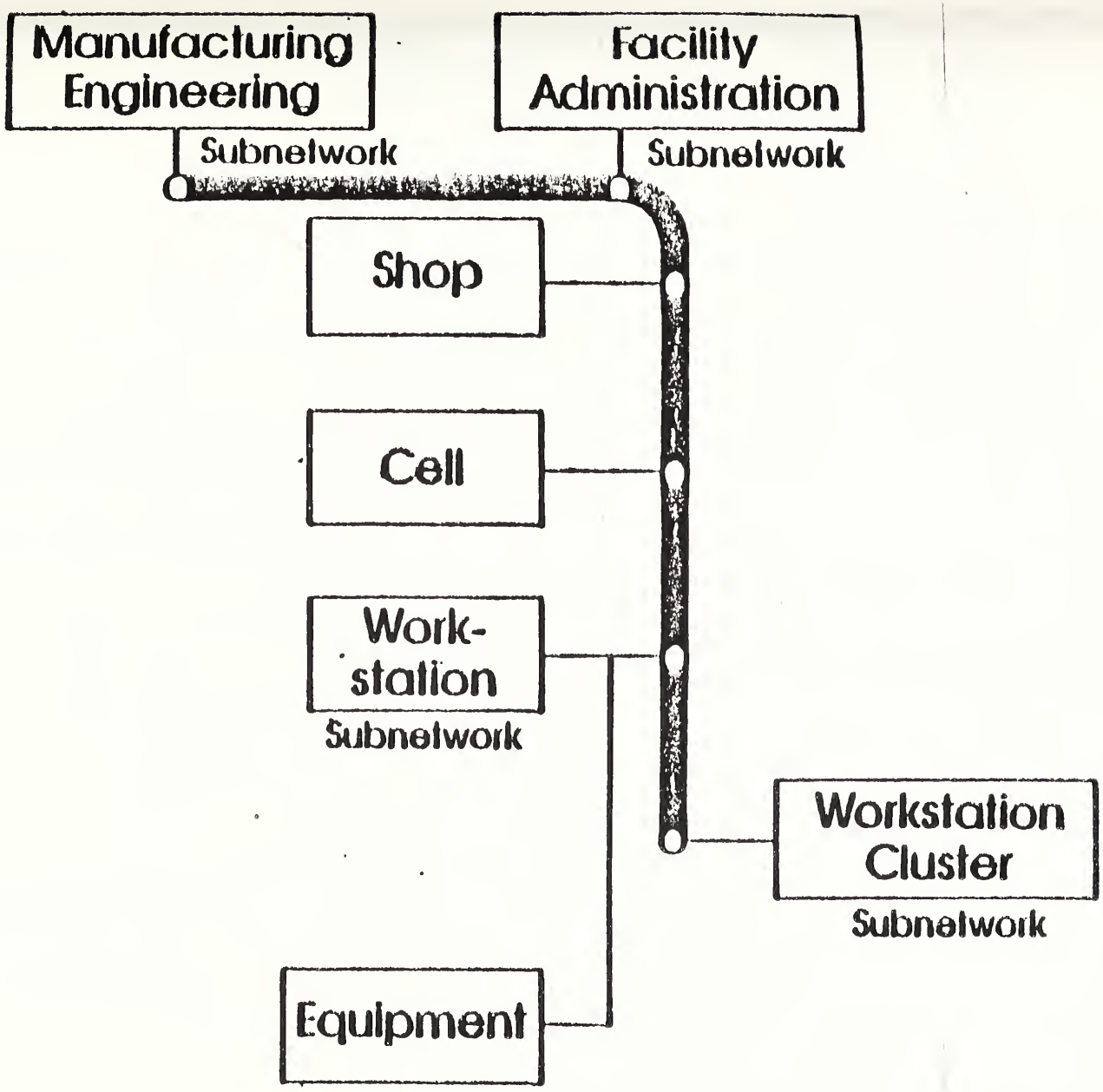
The Relationships Among Data Representations



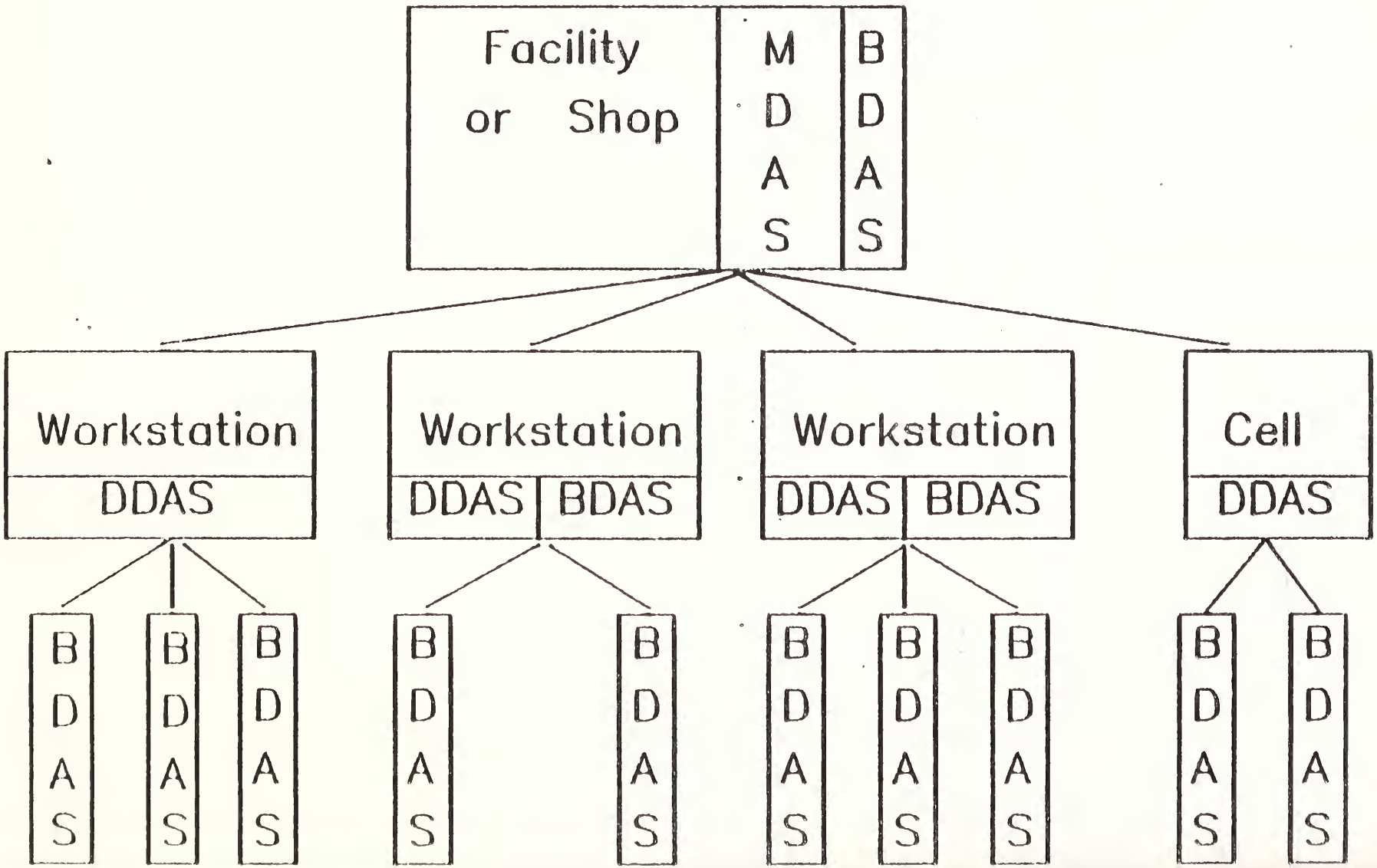
AMRF

NBS

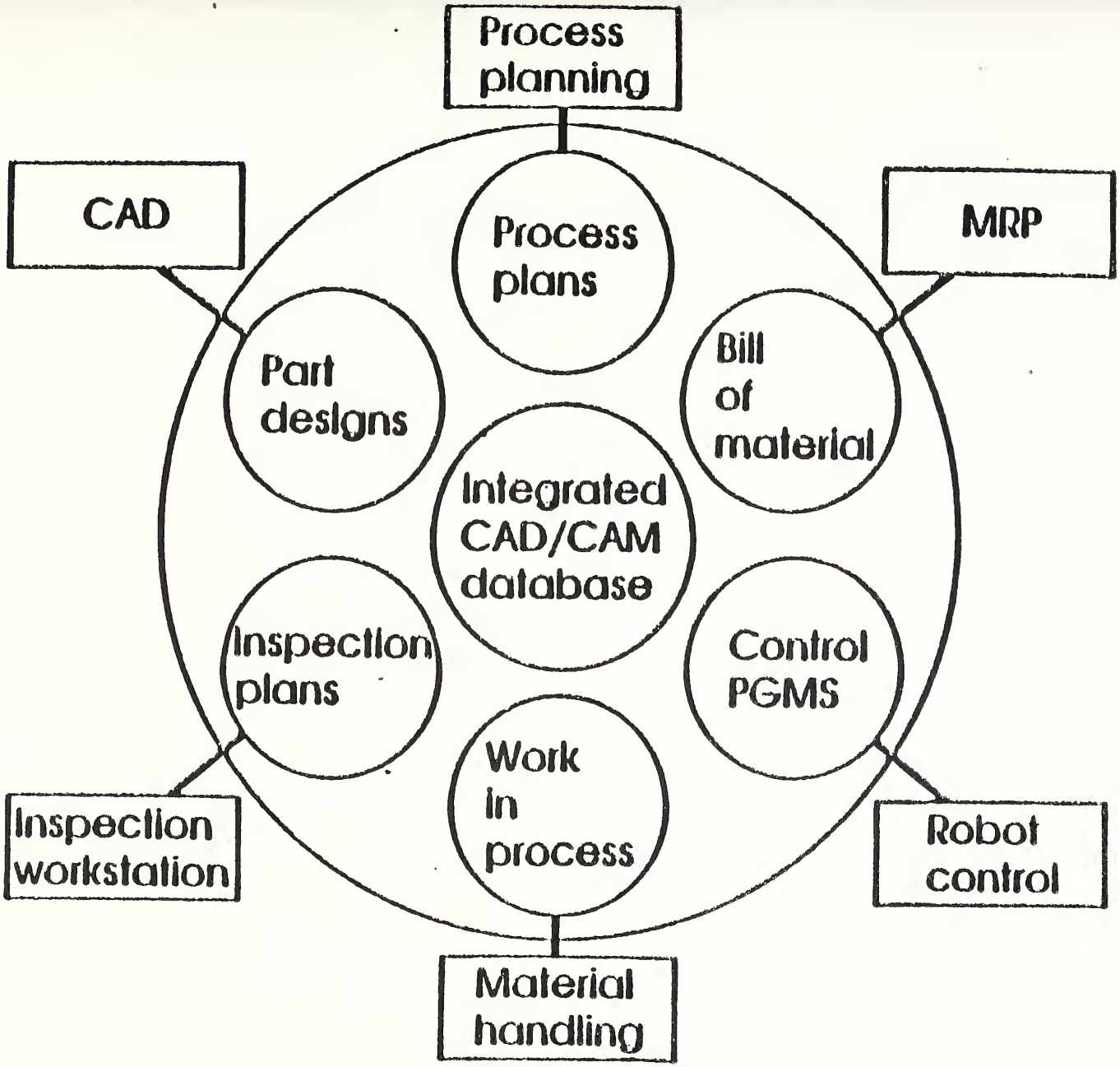
Factory Network Architecture



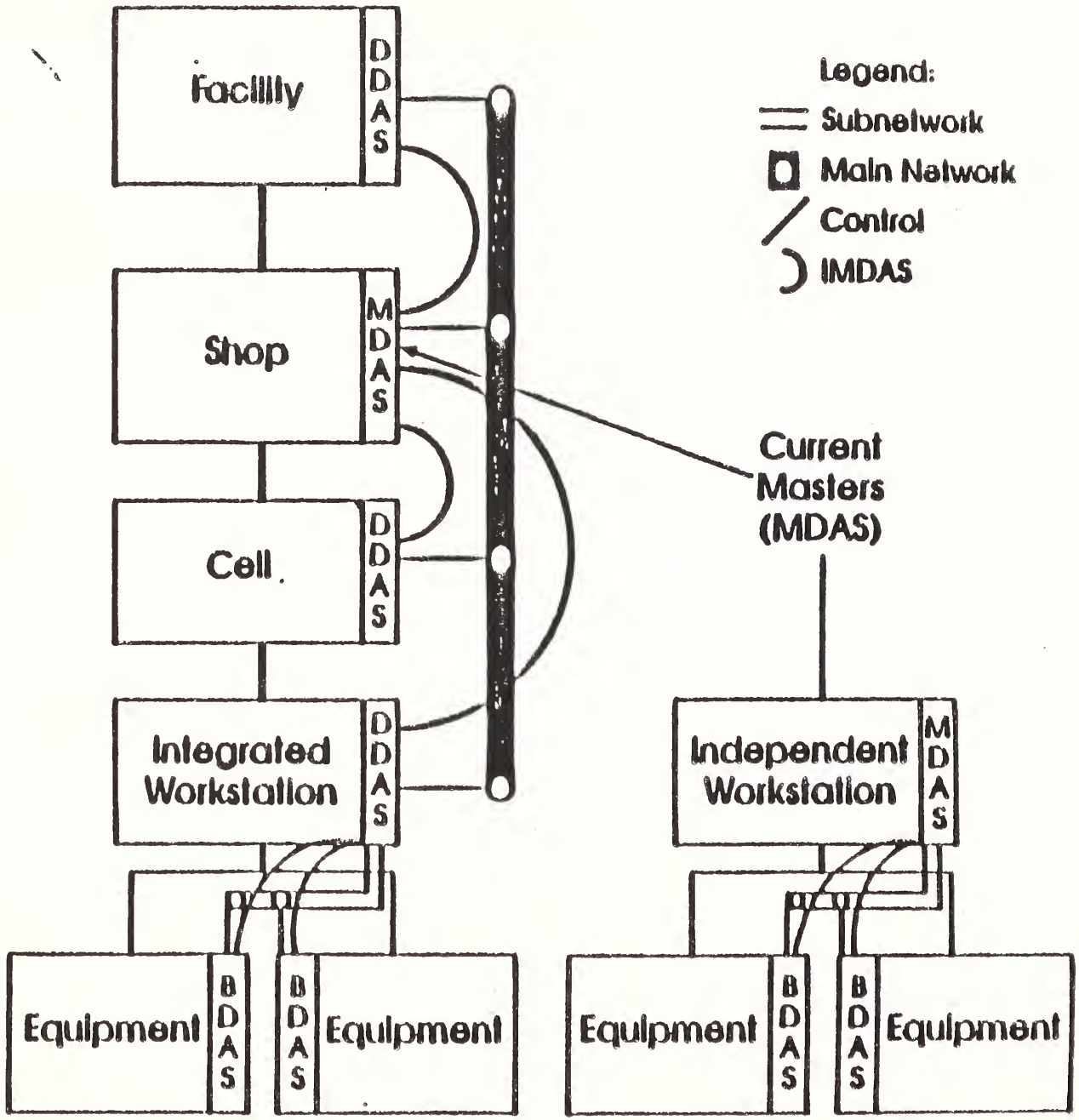
DISTRIBUTION OF IMDAS COMPONENTS ON A FACTORY DATA NETWORK



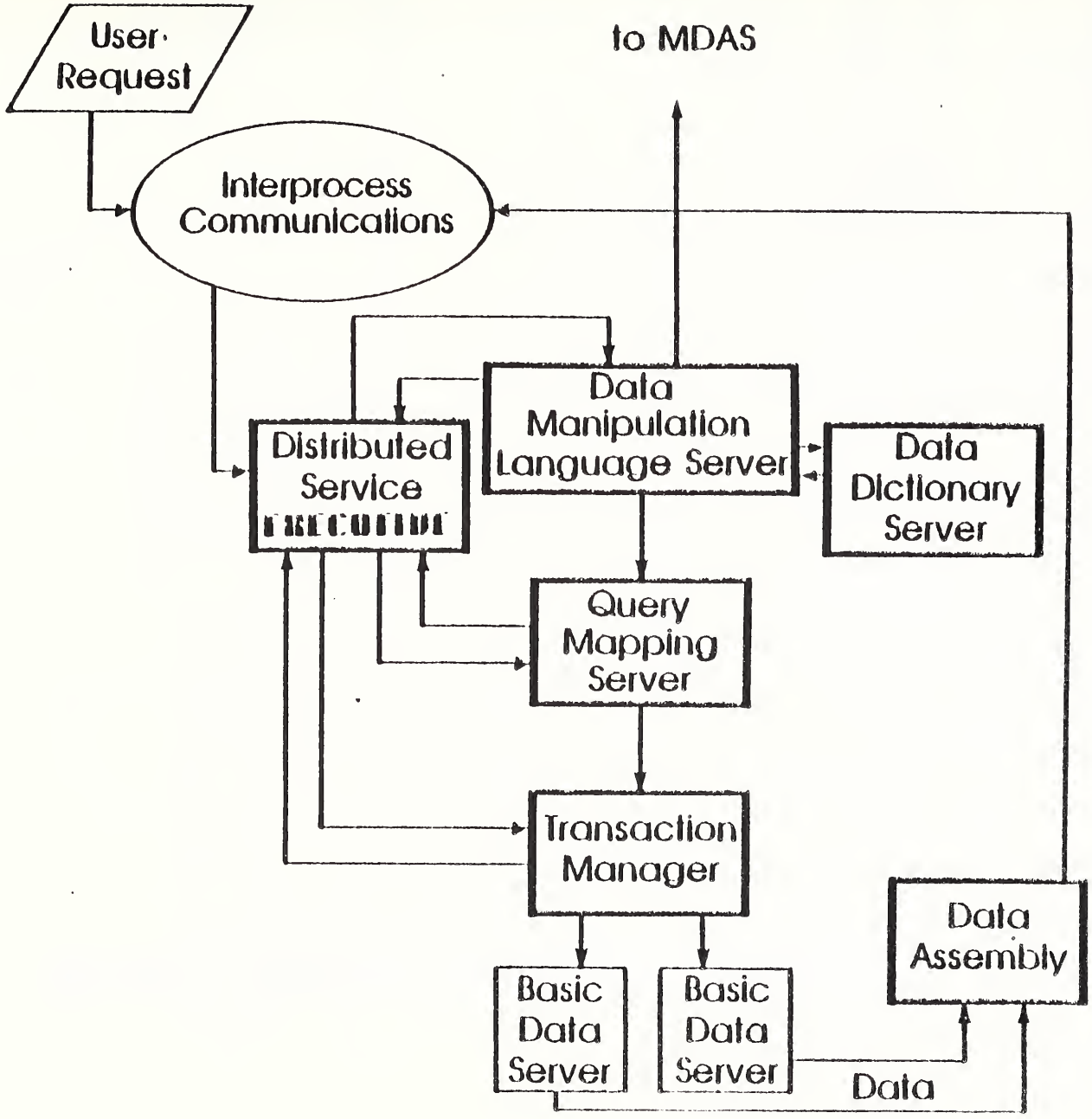
Integrated Database



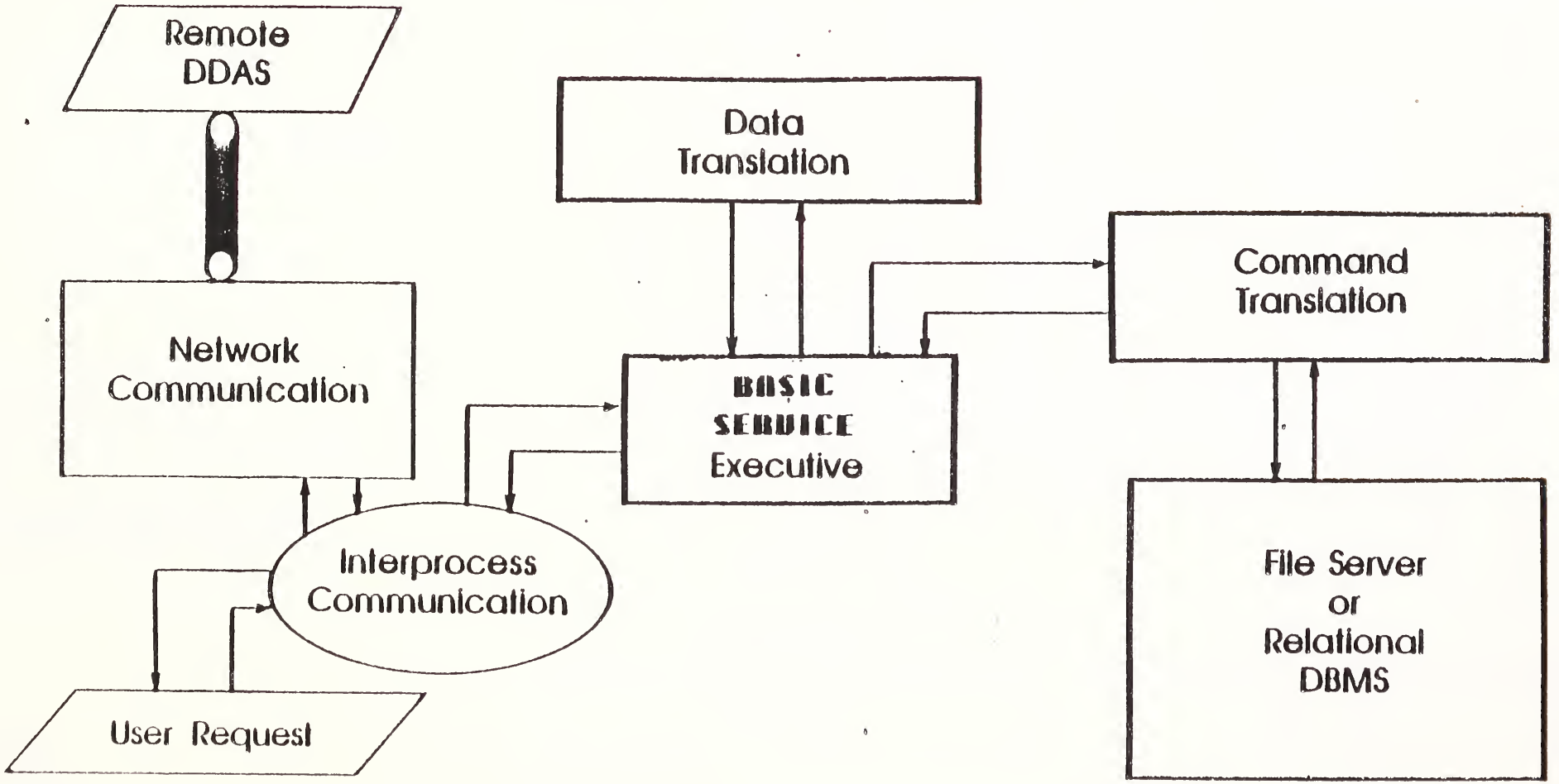
Network, Control, and IMDAS Topology



Distributed Data Server (DDAS)



Basic Data Server (BDAS)



NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 86-3312	2. Performing Organ. Report No.	3. Publication Date APRIL 1986

4. TITLE AND SUBTITLE

An Architecture for Distributed Data Management in Computer Integrated Manufacturing

5. AUTHOR(S) Edward Barkmeyer and Mary Mitchell (NBS), Krishna P. Mikkilineni, Stanley Y. W. Su, and Herman Lam (Univ. of Florida)

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

National Bureau of Standards
Building 220, Room B-229
Gaithersburg, Maryland 20899

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 requirements of Computer Integrated Manufacturing (CIM) are examined using the Automated Manufacturing Research Facility (AMRF) project at the National Bureau of Standards as the basis for this study. The impact of a CIM environment on data management is described. An architecture for distributed data management is proposed which will support the heterogeneous computing environment and allow for autonomous operation of manufacturing cells.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only the first letter of first key word unless a proper name; and separate key words by semicolons)

Automation; Computer Integrated Manufacturing; Distributed Database Management; Factory Integration; Heterogeneous Database Management; Query Translation; Query Optimization

13. AVAILABILITY

__ Unlimited

__ For Official Distribution. Do Not Release to NTIS

__ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

__ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

84

15. Price

$11.95