NISTIR 88/3828

NBS AMRF PROCESS PLANNING SYSTEM— SYSTEM ARCHITECTURE

July 27, 1988
Issued
March, 1989

By:
Peter F. Brown
Steven R. Ray

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards & Technology

NBS - AMRF

PROCESS PLANNING SYSTEM

SYSTEM ARCHITECTURE

Peter F. Brown
Steven R. Ray

Contents

Page

I. INTRODUCTION 1
1. PURPOSE OF THIS DOCUMENT 1
2. AUDIENCE 1
3. CONTENTS FLOW. 1
3.1. Casual User 1
3.2. Research User 1
3.3. System Implementor. 1
II. OVERVIEW 3
1. WHAT IS PROCESS PLANNING?. 3
1.1. The Interpretation of Design. 3
1.2. Facility Capability Models. 4
2. BACKGROUND SURVEY OF PROCESS PLANNING. 4
2.1. Approaches to Process Planning. 5
2.1.1. Variant Approaches 5
2.1.2. Generative Approaches. 6
2.1.3. AMRF Approach. 7
3. PROCESS PLANNING IN THE AMRF 7
4. THE INPUTS AND OUTPUTS OF PROCESS PLANNING 8
4.1. Part Model. 8
4.2. Process Plans 8
4.3. Inventory and StatuS. 9
III. ARCHITECTURE. 11
1. PROCESS PLAN DEFINITION FOR MANUFACTURING SYSTEMS. . .11
1.1. Representation Issues 11
1.1.1. Internal 11
1.1.2. External 12
1.2. Formal Language Definition. 12
1.2.1. Neutral ASCII Format for Process Plans . . .12
2. DATA ENTRY 12
2.1. Factory Configuration 13
2.2. Operation Sequence. 13
2.3. Requirements. 13
2.4. Header. 14
2.5. Parameters. 14
3. PROCESS PLAN MANAGEMENT. 14
3.1. Local Database. 14
3.2. Distributed Database System (IMDAS) 15
3.3. Part Model Access 15
IV. DESIGN CONSTRAINTS 17
1. CURRENT TECHNOLOGY 17
1.1. Design Systems. 17
1.2. User Interface. 18
1.3. AI Systems. 18
1.4. Communications. 18
1.5. Database Management 19

V. FUNCTIONS OF MAJOR COMPONENTS. 21
1. CONFIGURATION TOOLS. 21
1.1. Shop Floor Definition 21
1.2. Work Element Definition 23
2. PROCESS PLAN MODIFICATION TOOLS. 26
2.1. Graphic Network Editor. 26
2.2. Forms Editor. 26
2.3. Requirements Editor 30
2.4. Header Specification. 30
3. PROCESS PLAN STORAGE 30
3.1. Local Plan Storage Utilities. 30
3.2. Database Access Utilities 32
4. PART MODELLING MODULE. 32
5. PARSERS AND GENERATORS 32
6. EXPERT PROCESS SELECTION MODULE. 32
VI. IMPLEMENTATION 35
1. OBJECT-ORIENTED PROGRAMMING. 35
2. CAPABILITIES DATABASE. 36
2.1. Shop Floor Configuration. 36
2.2. Work Elements 36
3. PROCESS PLAN PRECEDENCE GRAPH. 39
3.1. Editing the Precedence Graph. 42
4. INTERNAL PART MODEL REPRESENTATION 42
5. DATABASE FACILITIES. 42
5.1. Protocols 44
5.1.1. Nfile. 44
5.1.2. Common Memory. 44
5.1.3. DML. 44
6. USER INTERFACE 45
BIBLIOGRAPHY 48
Appendix A - Relevant Publications A-1
Appendix B - Backus-Naur Specification B-1
Appendix C - Process Plan Format C-1
Appendix D - Example Process Plan. D-1
Appendix E - Hardware and Software Requirements. E-1

List of Figures

		Page
Figure V-1.	The Process Planning Facility Editor.	22
Figure V-2(a).	The Work Element Editor Tool, Showing the Machine Lot Work Element . . .	24
Figure V-2(b).	The Work Element Editor, Showing the Data Types Menu . . .	25
Figure V-3(a).	The Graphic Net Editor, Displaying a Cell Level Process Plan. The menu lists available work elements. . . .	27
Figure V-3(b).	The Graphic Net Editor. Menu shows network editing operations. . . .	28
Figure V-4(a).	Procedure Specification Editor.	29
Figure V-4(b).	Procedure Specification Editor. Menu shows available work elements. . . .	31
Figure VI-1.	Facility Editor	37
Figure VI-2.	Work Element Editor	38
Figure VI-3.	Network of Process Plans and their Elements . .	40
Figure VI-4.	Structure of Procedure Specification.	41
Figure VI-5.	Flavor Inheritance Graph for the Supervisor . .	43
Figure VI-6.	Process Planning System Operations Menu	46
Figure VI-7.	Procedure Specification Editor.	47

I. INTRODUCTION

1. PURPOSE OF THIS DOCUMENT

The purpose of this document is to provide a general description of design and implementation of the Automated Manufacturing Research Facility (AMRF) Process Planning System. The document should provide the reader with an understanding of the concepts behind the work in the process planning project as well as on the approach adopted. Details on system implementation are provided.

2. AUDIENCE

The intended audience for this work is the technical community already familiar with current process planning issues and practices, both research and commercial. While this document does contain a brief review of other planning systems, it is not intended as a tutorial on process planning. Rather, it discusses a new approach to process planning in an automated facility.

3. CONTENTS FLOW

3.1. Casual User

The casual user does not need to read this document if time does not permit. The User's Manual [1] is the primary document needed to simply operate the planning system.

3.2. Research User

A user who intends to use the planning system as part of a related research effort should at least read the sections on architecture, the functions of major components, and the User's Manual.

3.3. System Implementor

A system implementor should be familiar with this document and with the User's Manual. It should provide enough insight to enable the installation and operation of the planning system on a local machine.

II. OVERVIEW

1. WHAT IS PROCESS PLANNING?

Process planning is the translation of part designs into a representation of activities that will transform raw materials into finished products. In traditional manufacturing environments a part drawing is given to a process engineer, who first determines coarse requirements. These requirements will include the tightest tolerance that needs to be produced, rough estimation of the work volume, fixturing constraints, etc. With this information the process engineer can then make determinations of part routings: that is, which machine tools are capable of producing the part, how the part will be fixtured, what tooling will be required, and finally the determination of manufacturing features. This information can then be given to a machinist or N/C programmer who will do the detailed equipment level programming.

In a fully automated facility such as the AMRF all of the manufacturing steps will have to result in the development of process plans (these can be thought of as a program) for all relevant control systems.

1.1. The Interpretation of Design

Recognizing form features on a part and mapping those features to appropriate manufacturing processes is a fundamental step in process planning. Human process planners have been performing this mapping for quite some time. Currently, there are many research efforts underway to automate this process. There is also great interest in extending it to provide feedback from manufacturing to design.

On current CAD systems, the designer translates the functionality of a part, or group of parts, into geometry and tolerance notations on part drawings. A poorly conceived part geometry can overly constrain the manufacturing process. A better approach would be to force the designer to describe a product in terms of features that completely represent part functionality. These design features can then be mapped into manufacturing or process-oriented features. These manufacturing and design features will then be used to generate the geometry as it is realized by creating the manufacturing features. The goal is to give the geometry generation process the flexibility to change geometry to optimize the manufacturing operations without significantly changing the original functionality. This approach permits the

optimization of the manufacturing operations in order to make best use of processing equipment, select the lowest cost or simplest operation to produce the geometry, and modify geometry to ease the manufacturing or assembly tasks.

A set of tools is being developed to define a part in terms of its geometrical and topological entities, and functional entities known as features. These functional features are collections of geometrical and tolerance attributes on commonly reoccurring design or manufacturing patterns. Common manufacturing features include pockets, holes, and slots. This representation scheme allows computer programs to reason about, query, and interpret information between one another, without human intervention. A neutral part model file format has been established within the AMRF to convey geometry, topology and feature data between application systems [2].

1.2. Facility Capability Models

One very important aspect of the process planning problem is the understanding of the capabilities of the manufacturing environment. Trying to produce a part on the wrong piece of equipment leads to lost time, wasted raw materials, and in general, wasted or redundant efforts. Having the wrong mix of process equipment for the types of part families can lead to low utilization. In conventional shops this processing model is held by a number of individuals, who probably have a model of only a portion of the entire manufacturing operations. This leads to under/over-utilization of certain pieces of equipment. In future automated environments we expect much of this will be done away with.
In the process planning system we have started to develop a set of tools that provide models of the shop floor capabilities. These capabilities include the activities (work elements) that all control systems can perform, the required hardware and software needed to perform those activities, performance models of the machine tools and robots, and information about inventories such as raw materials and tooling.

2. BACKGROUND SURVEY OF PROCESS PLANNING

Process planning systems can play a major role in manufacturing automation. Numerous industries have developed and used process planning within their organizations for quite some time. In most cases this has been to provide the production planner, production scheduler, and machine operator with the information they need to do their job. This leads to lower costs, better parts, and a reduced manufacturing cycle. Process planning systems have been used to help develop more consistent parts by following standard

operations, and part routings. Coupled with group technology, standard plans can be created for part families, then modified to represent the unique characteristics of the individual parts. All of these reasons contribute to more consistent parts, lower costs, and shorter turn around times. Process planning is the link between the part drawing and the activities to be performed within a factory. It is important to identify previous work to show how our work fits with other process planning systems.

2.1. Approaches to Process Planning

This section describes several approaches to computer aided process planning. Primarily they can be divided into two separate classes: those requiring human intelligence to make decisions and those that do not. Variant systems and their derivatives require human input and decision making. Generative systems start with some form of part description and can automatically generate a process plan. All commercial systems today are variant systems or derivatives, some generative systems do exist and are being used within companies and universities for restrictive part families. Much research needs to be done before fully generative systems will be developed. For a more detailed discussion of the state of the art of computer-aided process planning systems, see Chang and Wysk [3].

2.1.1. Variant Approaches

Variant planning systems are based on a library of standard plans for different part families that a process engineer retrieves and edits, creating "variants" of basic plans. Variant systems typically rely on group technology classification and database management systems for their implementation. Standard process plans are developed for each family of parts produced and are stored in the database. When a new part enters the system, it is first classified by part family. The part classification code is used as a key to select a copy of the appropriate default plan from the database. This copy is then modified to reflect the specific processing required due to the unique characteristics of the new part. If a plan does not exist for the part's family, then a new default plan is created by an experienced process engineer and stored in the database system.

The technology that is required to implement this type of process planning system is readily available on main frame as well as personal computer systems. Indeed, almost all of today's commercial process planning systems employ variant techniques. With this approach most knowledge resides in the mind of the process engineer, the computer serves mainly as an organizing

tool. Although intelligent generative systems are often more desirable because of much less human involvement, there are significant benefits that can be obtained from the variant approach. The development of a variant system forces an organization to study and classify the activities that it can perform in order to understand the part families it can produce in its shop. This exercise, in turn, reveals the kinds of equipment and labor skills that the shop really has and needs. But, there are some limitations in the variant approach. It can often be impractical if the shop produces small batches of widely varying parts. More time has to be spent defining new part families and modifying default plans. Furthermore, it does not capture the real knowledge or expertise of process engineers. The generative approach to process planning does address these issues. The capture of this information can be of critical importance to industry which is facing large numbers of process engineers retiring, and new engineers not being trained to fill those positions. Secondly, the hope with generative systems is to develop much more consistent process plans, this is important in the competitive marketplace of today's industry.

2.1.2. Generative Approaches

The main thrust in process planning research today is in the area of generative systems, for some examples see [2,3]. In these systems, techniques from the field of artificial intelligence are used to automatically create a plan for a new part. An expert problem solving system uses an internal process knowledge base and part specific data to generate new plans. This approach requires that a full product definition or part model exists in a form that is accessible by the expert system software. This model should include geometry and topology, a tolerance model, and information about the functionality of a part. The knowledge base contains information gathered from process engineers on the how and why of making process decisions for various types of parts. Decisions are often keyed to the different types of features that are typically produced on parts. With this approach, the knowledge base becomes a repository of knowledge gained from the many years of experience of many process engineers. It also permits the separation of the process knowledge from the part data, which facilitates data driven automation. This is important because it separates the tolerance or functionality of the part in question from the techniques that will be used to produce it. This will allow new or different processing techniques to be substituted without major changes to the part representation.

To date, fully generative process planning has proved to be an elusive goal, but there are some signs of progress. The biggest problems have included the representation of features (pockets,

slots, holes), processes (drill hole versus bore hole), and precedence information (make pocket before hole). Furthermore, the outputs produced by process planning systems are non-standard. That is, the organization of data into forms or structures such as routing and operations sheets differs from system to system. As a general rule, plans are meant to be interpreted by human readers, rather than by a computer system. In the future, it will be essential that process planning interact more closely with automated control systems. Major questions with respect to the inputs and outputs of process planning systems must be resolved before fully computer-integrated intelligent manufacturing systems become a reality.

2.1.3. AMRF Approach

The AMRF process planning project is tackling questions regarding the fundamental role of process planning in automated manufacturing facilities where all operations are under some form of direct computer control. Our research focuses on the identification of basic concepts and principles that support the integration of process planning with manufacturing process control systems, scheduling, inspection, and all facets of the manufacturing life-cycle. Important steps leading to plug-compatibility between these systems include:

*the establishment of a system architecture for factories and their planning systems which accounts for both current and future expected capabilities,

*the definition and modeling of manufacturing activities or processes,

*the specification of a data representation scheme that can be used to organize and exchange information among planning, control, and other factory systems,

*the development of generic product descriptions in terms of both design and manufacturing feature geometries, and

*the verification of these potential interface specifications and protocols under realistic test conditions.

3. PROCESS PLANNING IN THE AMRF

Process planning in the AMRF currently exists in the reactive level of planning and control. Control systems are driven by

predetermined state tables. The planning system requires significant human input and is not based on state space and heuristic search. Models of work elements and requirements exist for major control systems. These are used when defining the process plans for those control systems. The sequences for the process plans are determined by the user and input into the process plan editor (described in chapter V). This set of editors is used to create, edit, and browse through process plans. These plans are communicated to the AMRF through the Integrated Manufacturing Data Adminstration System (IMDAS) [6]. These process plans are interpreted by control systems to sequence through their assigned activities. There exists no direct feedback to the process planning system about the performance of the process plans or for checking on the condition of the shop floor. Process planning data structures have been developed for all the current levels of the AMRF hierarchy. The representation of these plans is done with the process plan file format (see Appendices B and C). This format is used by all control systems that retrieve and execute process plans. Within the process planning project some work has been done on generative planning. Starting with a set of features defined in the part model, SIPS (Semi-Intelligent Process Selection) determines the process or set of processes required to produce the feature (see Appendix A). The system reasons about the tolerance constraints on the features using a set of process models about the capabilities of various machining operations. For more information about the AMRF process planning system see the papers in Appendix A.

4. THE INPUTS AND OUTPUTS OF PROCESS PLANNING

4.1. Part Model

The input to the process planning system is the AMRF part model for the part to be produced. This model contains the geometry, tolerances, and user definable "features". These features are items of particular technological significance (reoccurring patterns). These features are used by control systems to represent important information about the part, such as edge loops of a deburring operation, or machine features like pockets for the Vertical Workstation. The part model format can also be used to define intermediate part geometries. The process planning system will use the part model format to specify intermediate part geometries.

4.2. Process Plans

Process plans are the output of the planning system. These are the steps necessary to transform the part geometry into an actual

part. The process plans are stored in the AMRF process plan format. This storage is on the local Lisp machine file system or on the AMRF database system on the VAX.

4.3. Inventory and Status

Currently, inventory and status information resides only on the local planning system. An interface to the IMDAS has been developed, currently we do not get inventory and status information through IMDAS.

III. ARCHITECTURE

1. PROCESS PLAN DEFINITION FOR MANUFACTURING SYSTEMS

1.1. Representation Issues

Before a process planning system can be implemented in a general way, the fundamental issues of how to represent a process plan must be addressed. By adopting a general and flexible representation scheme, the planning system can evolve while still using the same data structures. This problem can be broken down into two categories, internal representation, and external representation.

1.1.1. Internal

The first problem to tackle is the representation of an individual activity or process step for an entity within a factory. In the AMRF, these process steps are described in terms of "work elements". Work elements can be thought of as operators within a state space. Whenever a work element is invoked, a state transition takes place. A process plan corresponds to a sequence of operators applied to an initial state, resulting in a goal state. The challenge is to represent the manufacturing task in the framework of artificial intelligence concepts such as the ones described here, so that the problems can be handled using current and future AI techniques.

The concept of frames was used to represent work elements. While the current implementation is not a frame-based system yet, the design is such that it will naturally fit into a frame implementation when one becomes available. Basically, each work element consists of a name and a set of attribute-value pairs, like a frame with slots. The name identifies the work element, while the attributes and values serve to completely specify the details of a particular work element instance. Such specifications could include necessary serial numbers of parts and trays, coordinate information for feature creation, target workstations for a particular task, etc.

With a robust representation for individual activities in place, the next problem was to develop a representation for an entire process plan. As will be discussed more fully in the implementation section of this document, the core part of a plan consists of a precedence graph of connected work elements, where the precedence relationships imply sequencing in time. This precedence graph is actually part of a larger structure (the process plan itself), which contains pointers to the various parts

of the plan. These are: a header section, which contains various bookkeeping information; a parameters section which identifies variables used within the plan which cannot be bound before execution time; the procedure specification, discussed above; and the requirements section, which lists all the hardware and software necessary for the execution of the plan. Internally, the entire structure is represented as a network of connected objects (see Chapter VI) which can send and receive messages.

1.1.2. External

In keeping with the need for simplicity and universality, the external representation of a process plan and of work elements does not take advantage of any advanced programming concepts, such as object-oriented programming. This representation is used for storage and communication of process plans throughout the factory. As shown in the example of Appendix D, it is in human readable, ASCII text form, called the neutral data exchange format [7]. The structure of the exchange format process plan is outlined in Appendix C, showing the four major sections identified above.

1.2. Formal Language Definition

1.2.1. Neutral ASCII Format for Process Plans

A brief overview of Backus-Naur form can be found in Appendix B. This notation is useful for the unambiguous specification of a formal syntax, such as the process plan neutral format. The process plan format specification can be found in Appendix C. This format is used by all controllers on the AMRF shop floor for the interpretation of process plans. Parsers have been implemented in a number of computer languages, including C and Lisp.

2. DATA ENTRY

When generating process plans to support the AMRF, several different types of information are necessary. First, there must be a context within which the plan has meaning. This means that the configuration of the factory must be known, including what equipment exists and what its capabilities are. Only then can the plan be evaluated as to its feasibility or optimality. Second, the operation sequence itself must be provided, (this is really the core part of a process plan). This information currently is provided by a human process engineer, with one exception at the equipment level, where an expert system, (SIPS), can provide process sequencing suggestions. The third piece of information is the list of requirements needed to perform the steps specified. Most of this is automatically provided by the planning system, which scans the procedure specification section. Finally, higher level information such as the part material, process engineer,

Group Technology (GT) code and so forth must be specified. One important piece of data which falls into this category is the name of the part model which may be referenced in the process plan. The part model contains all the topological, geometric, and tolerance information needed to fully describe the part. Work elements can refer to features defined in the part model to specify coordinate and tolerance information. Details on how data is actually entered into the system can be found in the User's Guide.

2.1. Factory Configuration

The representation of the factory configuration is accomplished by maintaining a tree of all the entities capable of handling process plans. Each entity has an associated collection of work elements which it can understand. Each time a process plan is read, edited or created, the configuration model is consulted to determine the validity of the work element or requirement being added. The configuration model can be altered at any time, and alternative models can be loaded, which could represent different facilities. Changes to the configuration model should only be made by the manager of the planning system, since a single change in the model can make large numbers of existing process plans unreadable. Changes which the manager can make include the addition or deletion of entities on the shop floor (cell, workstation or equipment), and addition of, deletion of, or changes to existing work element definitions. Thus, the configuration model maintains the "language" in which all process plans are expressed.

2.2. Operation Sequence

The task of generating a process plan consists mainly of constructing a precedence graph of the work elements to be invoked upon execution. The work elements maintain pointers to "parents" and children within the graph. Editing the sequence of operations is carried out by adding, deleting, or modifying work elements in the graph. The modularity of the work elements making up the operation sequence lends great flexibility to a process plan. The precedence graph, which is actually the internal representation of the procedure specification, is one slot in the data structure known internally as the plan. Other slots include the requirements and the header.

2.3. Requirements

The requirements section of the process plan structure is actually represented in a similar manner as the procedure specification. This is most clearly seen by the inheritance structure discussed in the implementation section of this document. Again, the requirement entities are represented as nodes in a graph. This time, the relationship between nodes does not imply any ordering.

There is the capability, however, to establish an explicit relationship for identifying sets of requirements. This capability might be needed, for example, to describe a kit of tools. Any individual tool may be referenced, or the entire kit may be named. The individual tools can be defined as components of the kit.

Requirements for a process plan can be either pieces of hardware or software needed for the execution of the plan. The software includes other process plans referenced in a given plan as well as N/C programs. Hardware includes the necessary workstations, tools, trays, parts, etc. These are all represented within structures analogous to work elements for the procedure specification.

2.4. Header

The header is the third major part of a process plan. It contains generic information pertaining to the plan. Entries include the process engineer, the part material, the lot quantity, etc. A particularly important entry is that of the part model which accompanies the plan. The part model is used to fully describe the part topology, geometry, tolerances, and functionality. Work elements within a process plan may refer to features appearing in the part model as a pointer to more detailed information on dimensions and tolerances.

2.5. Parameters

The parameters section is simply a collection of all attribute values occurring in the procedure specification section beginning with the "$$" prefix, which denotes a parameter. These are really just "dummy" variables, which will be replaced with actual values at execution time. Examples include serial numbers for trays and tools. The parameters section is useful at execution time to identify what information must be bound before execution can begin.

3. PROCESS PLAN MANAGEMENT

There are two locations where process plans can be stored: locally, and in the AMRF distributed database system. The two alternatives are provided to allow continued operation of the planning system even if the AMRF network system should fail.

3.1. Local Database

The local storage of process plans is carried out by converting the internal representation of a plan, (a structure referencing the three sections described above), into the ASCII format described in paragraph 1.2.1 of this section. This text is then stored as a simple file on the local file system, with a name

stored as a simple file on the local file system, with a name specified by the process plan naming convention. Retrievals are carried out by file name alone, rather than through database key field queries.

3.2. Distributed Database System (IMDAS)

Process plans can also be stored and retrieved via the Integrated Manufacturing Data Administration System (IMDAS). Here, the plans are again converted to the neutral ASCII format before storage, but key fields are also assigned to the plan. These fields include the plan name, the executing system and the version number. The storage is requested using a generic query language (DML) and supplying the file name where the process plan is stored. The IMDAS copies the file into an internal database, along with the key fields. Plan retrieval is done by key fields. This can be simply the specification of the plan name, but may include combinations of the other fields as well. Once a plan has been retrieved, the planning system converts the neutral format back into the internal graph representation. At this point, there is no difference between a plan retrieved locally and one retrieved from the IMDAS.

3.3. Part Model Access

The AMRF part model is used in conjunction with a process plan, both when creating and editing a plan, and when executing the plan. In the planning system, the part model is parsed into an internal representation of connected programming objects (see the next section). This internal representation can be used as an aid to the process engineer, or to the expert process selection module (SIPS) in use at the equipment level of planning. It is the part model that gives the feature characteristics to the SIPS module when it is reasoning about the optimum process for the creation of a feature. Currently, the part model is provided to the planning system by simple file transfer from the Geometry Modelling System (GMS). In the future, this will be handled through the IMDAS.

IV. DESIGN CONSTRAINTS

This section outlines a number of the original system design constraints placed upon the design team.

1. CURRENT TECHNOLOGY

The original intent of the process planning system is to develop a tool that could support the process planning requirements of the AMRF, as well as to serve as a tool for testing advanced concepts in process planning research. A number of general goals are outlined, such as friendly, easy-to-use systems and advanced concepts from computer science such as artificial intelligence. This system is to incorporate the latest user interface techniques, mouse pointing devices, pull down menus, integration of text and graphics, and new representation techniques such as frame systems and object oriented programming. Most of this led to the choice of a specialized computer environment provided by a Lisp machine. A Lisp machine provides an interpreted software environment, which leads to improved programmer productivity. In addition this environment provides the best tools for the development of expert systems which will play a critical role in the automation of process plan generation. One of the most important benefits from this environment is the capability to do rapid prototyping of system components.

1.1. Design Systems

The process planning project has the function of deciding what shape the original stock should take, specifying what processing should be done to the part leaving it in various intermediate geometries. In addition the process engineers also specify the fixtures for the part. All of these items call for the use of some kind of design tool. Early in the AMRF project it was decided that the resource did not exist to do research in the area of computer aided design systems; consequently, the project has never had a good design tool available. Most of the work was to be done with commercially available CAD systems. As individual systems (such as inspection, cleaning and deburring, and process planning) matured they outgrew the capabilities of the CAD tool. This led to the development of the AMRF part model format, which contains many of the aspects required for advanced manufacturing systems.

1.2. User Interface

The long term goal of the planning project is to provide tools which can automatically develop a complete process plan from some description of the part. This goal is many years away and many issues need to be resolved. The original design called for a very easy-to-use tool for developing process plans that required minimal user typing and bookkeeping. In addition, there was a desire to build a system which provides a means for integrating expert planning modules as they were developed. The overall goal was to make the process engineers as productive as possible, and in complete control of plan development. Any portion of the plan generated automatically would still be available to the process engineer for review and modification. As confidence grew in the expert modules, less review would be required, and efforts could be spent on the further development of these expert planning modules.
This has lead to the development of a friendly easy to use system for the creation of process plans. As much error checking as possible is done during plan development to prevent plans from being distributed with wrong information.

1.3. AI Systems

While the original planning system was to be interactive, the design called for the integration of expert planning systems as soon as was possible. This called for the development of the planning system in a computer language and environment that allowed for the development of such AI systems. The Symbolics[1] Lisp machine environment provides a wide array of artificial intelligence tools. These tools are available from both universities and commercial organizations. They provide a wide variety of techniques to model the reasoning mechanism used by humans.

1.4. Communications

The AMRF process planning system is integrated with the AMRF network. The Lisp machine provides support for the Transmission Control Protocol / Internet Protocol (TCP/IP) network protocol.

---

[1] Certain commercial equipment, instruments, or materials are identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

Common memory which is used by most AMRF systems has not been implemented on the Symbolics. To date we have a protocol called Nfile running between the Symbolics and the Sun computers. The Nfile process on the Sun communicates with the common memory. In the future, if sufficient resource are available, we would like to have some form of common memory running on the Lisp machine.

1.5. Database Management

The original system design called for integration of the planning system with the AMRF database systems. This integration was originally envisioned to cover the complete storage of process plans and their associated elements in relational tables. This would allow for a wide variety of searching capabilities. As the system is currently implemented, process plans are only stored as files. A mechanism has been implemented to interface to IMDAS to make it possible to store and retrieve a wide variety of information required by the process planning system.

V. FUNCTIONS OF MAJOR COMPONENTS

This section outlines the functions of the major process planning system components. It includes a description of the configuration tools, editing tools, plan storage and retrieval tools, part modeling, parsers, and the expert system tools.

1. CONFIGURATION TOOLS

The configuration tools of the planning system are meant to be used by the planning system personnel alone or with workstation or system implementors who are very familiar with the process planning system. The tools allow the development of alternate factory floor configurations, as well as other manufacturing facilities. These tools are used to define work elements and requirements to be used by equipment within these factories.

1.1. Shop Floor Definition

This is the first tool one would use in creating an entirely new planning system for a facility. This tool allows the creation of models of the factory floor systems and other control systems. It only maintains a logical view of the relationships between systems. Figure V-1 contains a screen-dump of the actual tool. There are several major sections to this tool. The window for this tool consists of four panes, the largest pane in the upper left, contains the actual view of the factory hierarchy. In the referenced figure, the cell, workstation and equipment levels of the AMRF are shown. The second pane in the upper right quadrant gives an overview of the entire hierarchy, showing where the current viewport is located. This is often necessary because of limited space available on the screen. The third pane, in the lower right quadrant, contains an object description pane. The fourth pane, lower left quadrant, is a text-interaction pane for responding to questions from the planning system. This tool is used to describe the logical relationships between equipment on the shop floor. Its second major function is to keep track of the available work element definitions for all of the cell, workstation and equipment identified. These work element definitions are used when a user creates a process plan for a particular system. In the referenced figure we are showing the AMRF configuration of the cell, the Inspection Workstation, the Horizontal Workstation, the Cleaning and Deburring Workstation, the Vertical Workstation, and all of the associated pieces of equipment. The cell and workstations are abbreviated CELL, IWS, HWS, CWS, and VWS respectively. Displayed in the object description window is a description of the Vertical Workstation,

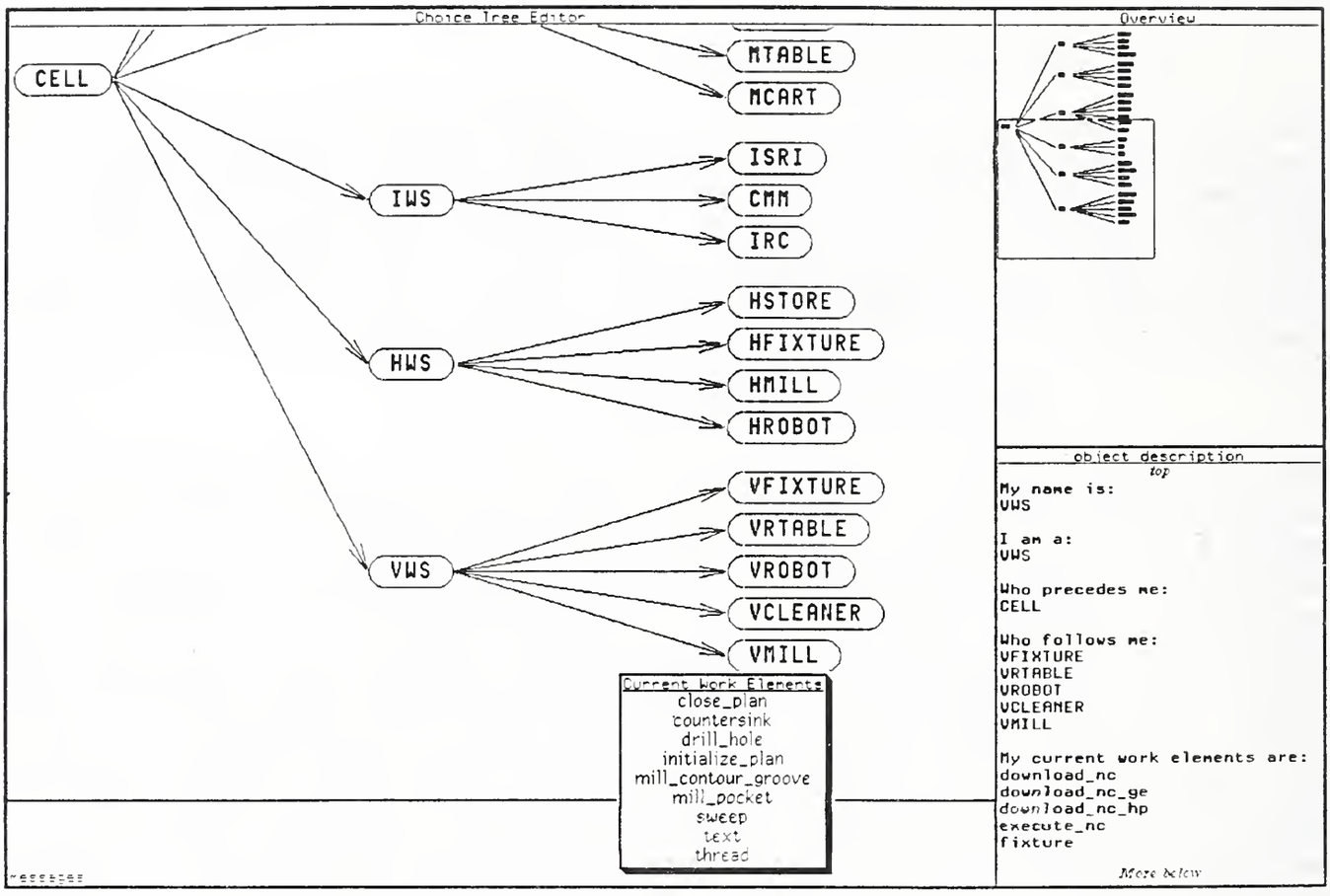


Figure V-1. The Process Planning Facility Editor

which is called VWS. The menu in pane one is currently showing the defined work elements that can be used in a process plan for the milling machine of the vertical workstation. This tool can be used to edit the definition of the work elements which will be described in the next section.

1.2. Work Element Definition

Figure V-2 (a) shows a screen dump of the work element editing tool. It consists of five panes. The first pane is the title pane which tells the users which work element is currently being edited. The second pane, to the left, immediately below the title shows all of the work elements defined for the portion of that hierarchy. The third pane, to the right, immediately below the title is the work element template to be filled in by the user. The fourth pane is the menu pane, below panes two and three. It describes major functions for the window such as editing,viewing, saving, etc. The fifth pane is a text-interaction pane for displaying instructions to the user, and for the user to answer questions. The figure shows the machine lot work element of the cell being edited. All of the other cell level work elements are displayed in pane number 2. The system allows a user to copy a previously defined work element and make changes to it or to create a new work element from scratch. The user fills in a template in the third pane where, currently, the major fields are "Autogen *", "time", and "User Property". The autogen nodes are used to automatically add items to a process plan. The "Autogen Rqmts" slot allows a user to define what items from this work element need to be added to the requirements list section of a process plan. The other autogen slots are currently not used but are kept for backward compatibility with a pre-release version of the software. The time field allows the user to specify a default time for the action to take. This field is in all work elements, it can be changed when editing a process plan. The "User Property" is where the attributes and their data-types are defined for the work element. The attributes can be any set of characters up to 16 in length. The data types are chosen from the menu displayed in Figure V-2 (b). In this figure, the current work element, machine-lot, shown has the attributes system, type, plan_id, lot_id, and lot_size. Their associated data types are symbol, symbol, symbol, symbol, and a number, respectively. When the user has finished editing a work element, it is saved in the current factory database and available for use in new and existing process plans. The internal Lisp flavor objects are also created for the system.
One of the major reasons for creating this tool is to allow for quick and easy modification of work element definitions without having to write or edit a single line of Lisp source code.

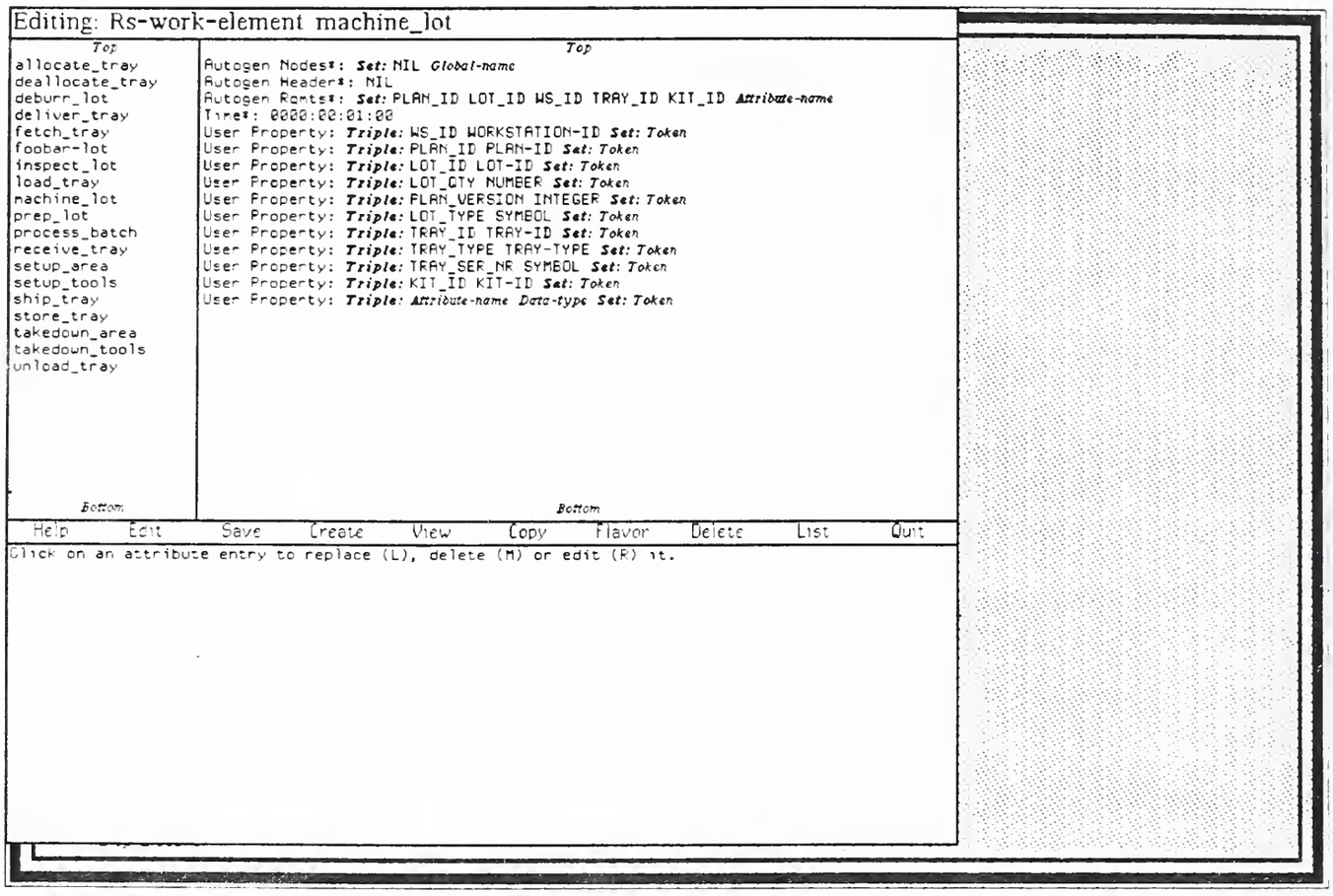


Figure V-2(a). The Work Element Editor Tool, Showing the Machine Lot Work Element

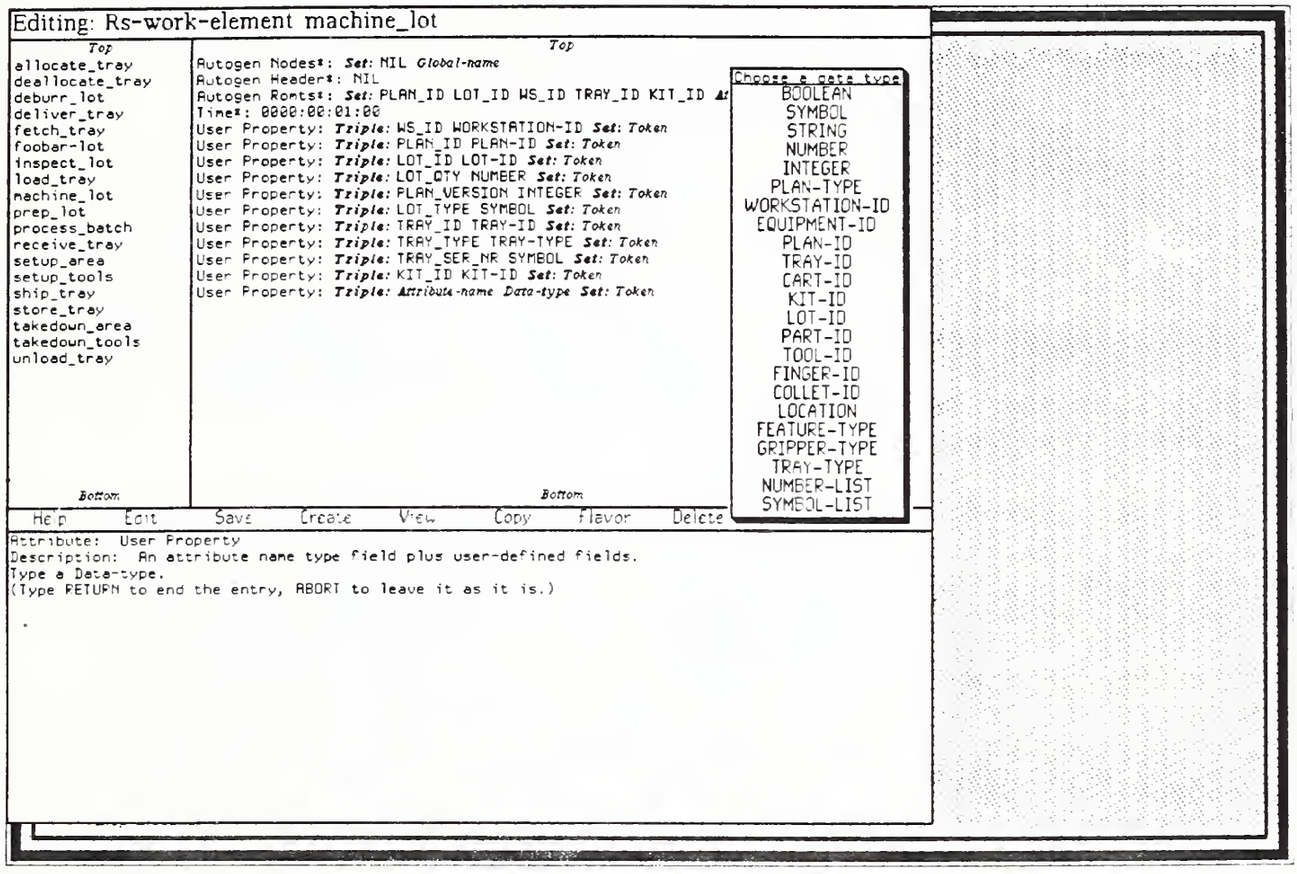


Figure V-2(b). The Work Element Editor, Showing the Data Types Menu

2. PROCESS PLAN MODIFICATION TOOLS

The second major set of process planning tools are those to be used by individuals wishing to edit, modify, or create process plans. There are two such tools: the Forms Editor and the Graphic Network Editor.

2.1. Graphic Network Editor

The internal representation of a process plan is a precedence graph. This representation allows the specification of parallel activities for those workstations that can handle them. Figure V-3 (a) shows a screen image of the Graphic-net editor. The Graphic-net editor consists of 5 major panes. The first, in the upper left quadrant, is a viewport showing a portion of the current procedure specification being edited. The second pane, in the upper right quadrant, shows the entire procedure specification that is being edited. The third pane, below the overview, shows the object description pane. Below this is the fourth pane displaying various editing functions. The fifth pane, in the bottom left quadrant, is a text interaction pane where the user responds to system queries.

The figure shows a precedence graph representing a procedure specification for the cell level. This tool allows one to describe activities that can be done in parallel as well as strict linear sequences. In the current graph are two parallel paths of delivering items to a workstation followed by a machining operation then shipment of finished parts and tools to the appropriate workstations. The choice of the actual sequence can then be made by the local control system. In future work, we envision the planning system being able to sort the graph based on some criterion such as minimizing tool changes or tolerance stackup. The menu shows the work elements that can be added to a process plan. Figure V-3 (b) shows a menu of available options for modifying the precedence graph and selecting other windows. The items include cutting links, adding links, editing a node in the graph, etc.

2.2. Forms Editor

A second tool, the Forms editor, exists for viewing and editing the procedure specification. The information in this window is exactly the same as that in the previous section; it is simply presented to the user in a different format. Both tools use the same underlying precedence graph representation. Figure V-4 (a) shows a screen image of the forms editor. The forms editor consists of 4 panes. The first pane, in the upper left, shows the procedure specification being edited; it has a title section giving the name of the work element, and a body showing the

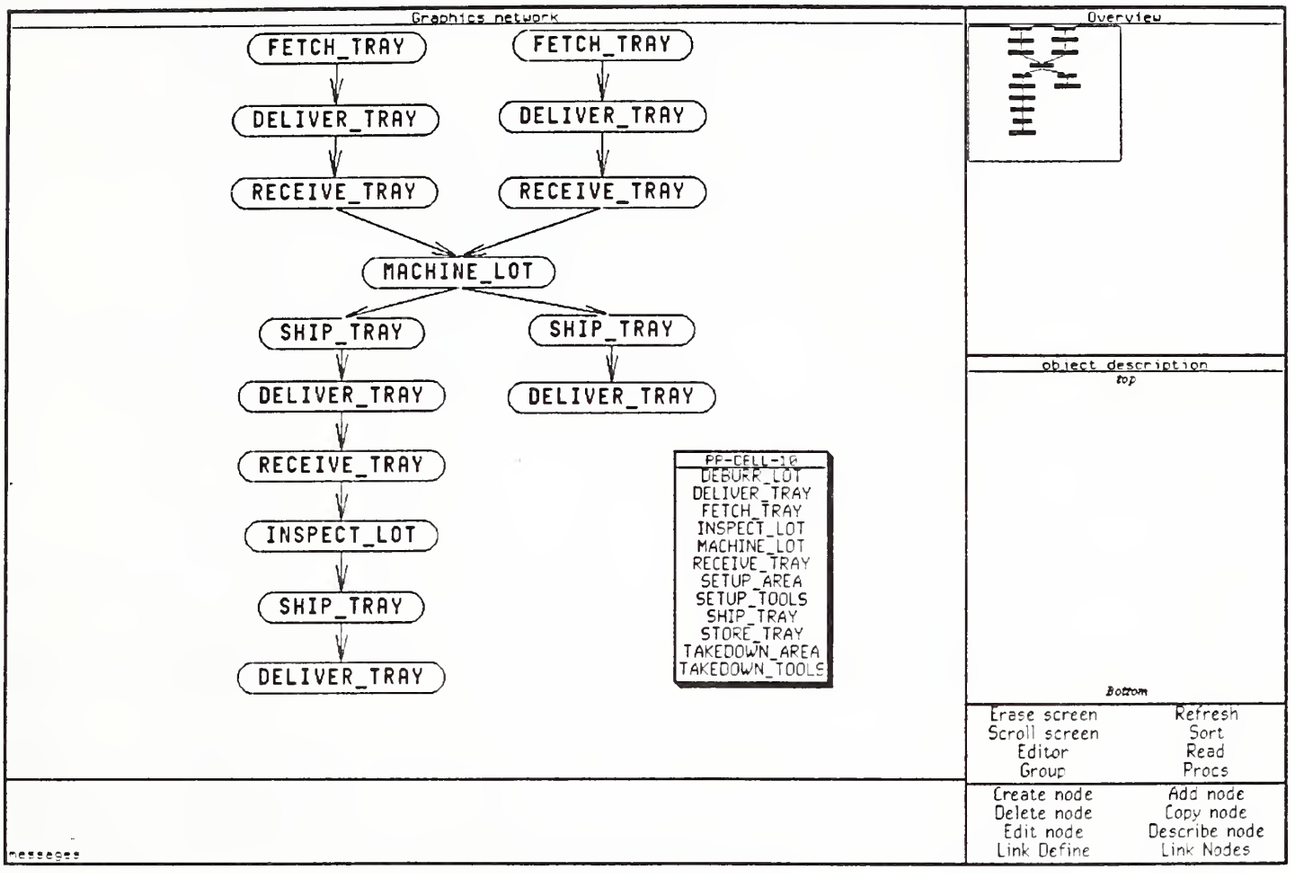


Figure V-3(a). The Graphic Net Editor, Displaying a Cell Level Process Plan
The menu lists available work elements.

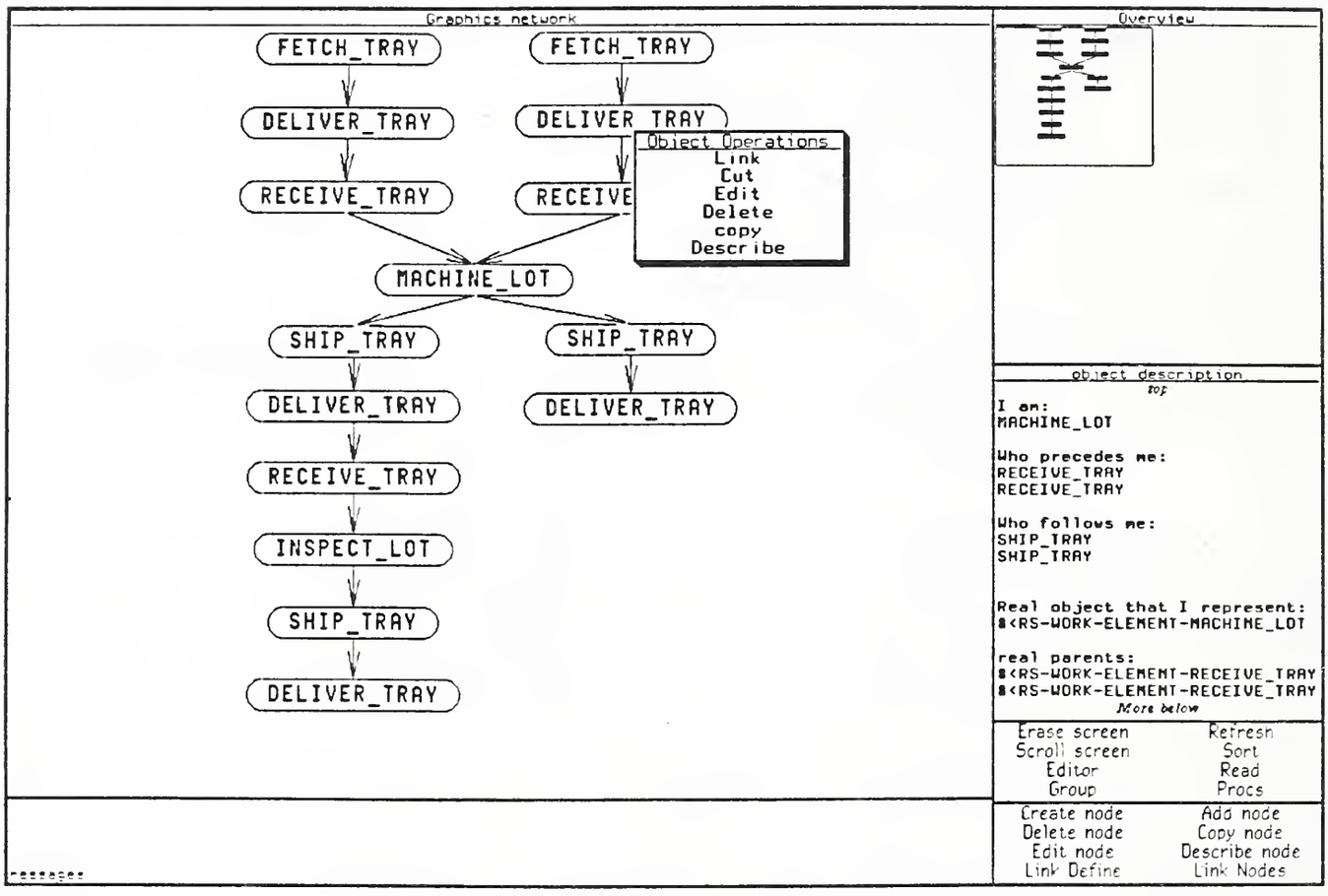


Figure V-3(b). The Graphic Net Editor Menu shows network editing operations.

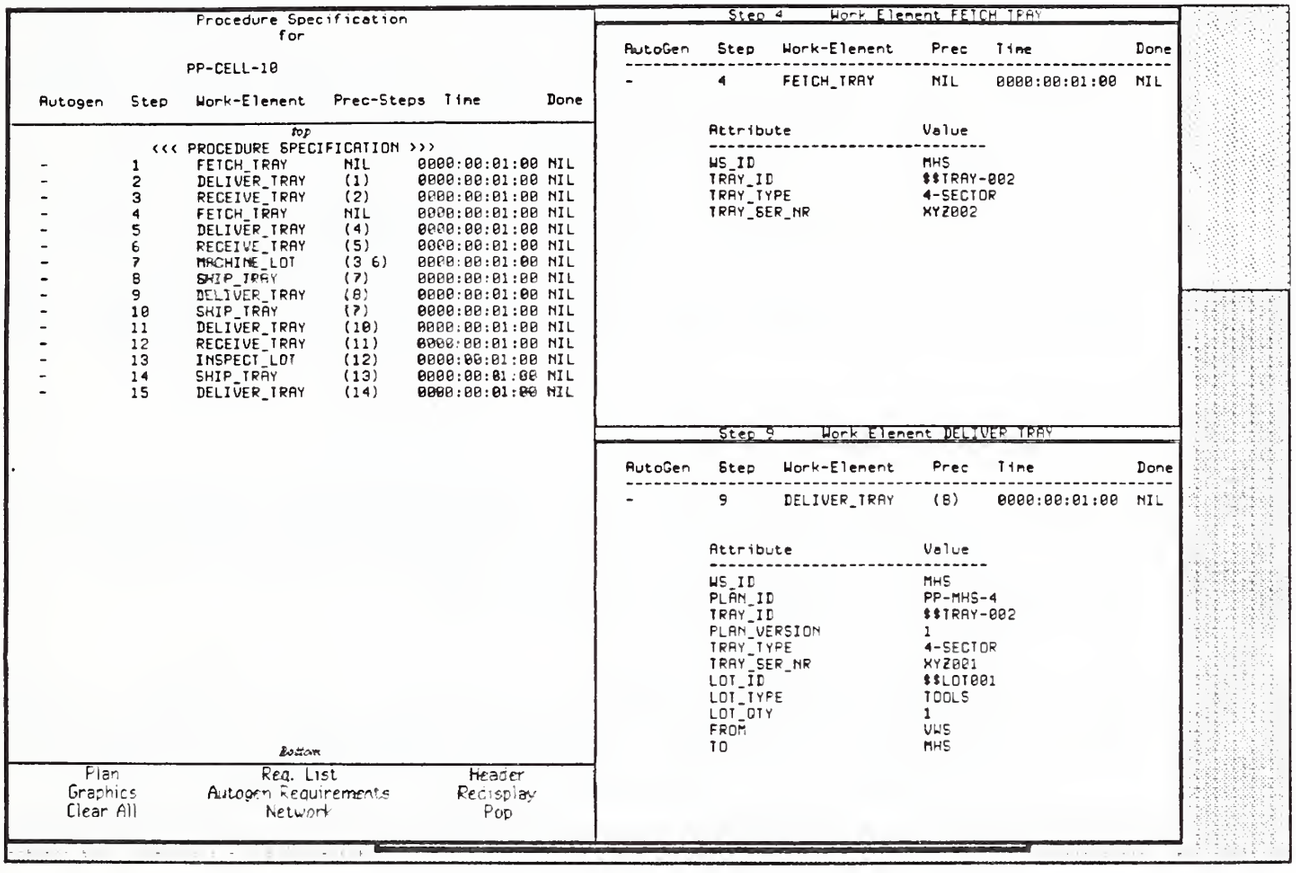


Figure V-4(a). Procedure Specification Editor

procedure specification. Panes two and three are on the right upper and lower portion of the screen respectively. They are examiner panes for the procedure specification. The fourth pane, lower left, is a menu bar showing other process planning tools available to the user.

A procedure specification is displayed in pane 1; it shows the current step number, the name of the work element, the precedence steps for the work element (those steps that must be done before this step), and an estimate of the time it will take for this work element to complete, followed by two entries labeled done and autogen. These last two are used to tell the user if all of the attributes have been given values and whether the node was automatically generated. The examiner panes are used by the user to view the details of an individual work element (to look at all of the current value bindings for a work element's attributes), and for editing those values. In the examiner pane the user can also edit the time and precedence steps of a work element. Figure V-4 (b) shows the available work elements that can be added to the system.

2.3. Requirements Editor

The requirements editor has exactly the same functionality as the forms editor but is used for the requirements list section of the process plan. The look and use of the requirements editor is the same as the procedure specification editor.

2.4. Header Specification

The header editor is a simple menu containing the current fields of the header and their current variable bindings. The values can be edited by selecting the value with the pointing device. Clicking on the element will allow the user to edit the field.

3. PROCESS PLAN STORAGE

Process plans can be stored in two ways: first, as ASCII text files on the Lisp machine file system, second, as process plans in the AMRF database system.

3.1. Local Plan Storage Utilities

The local storage exists on the local file system on the Symbolics Lisp machine.

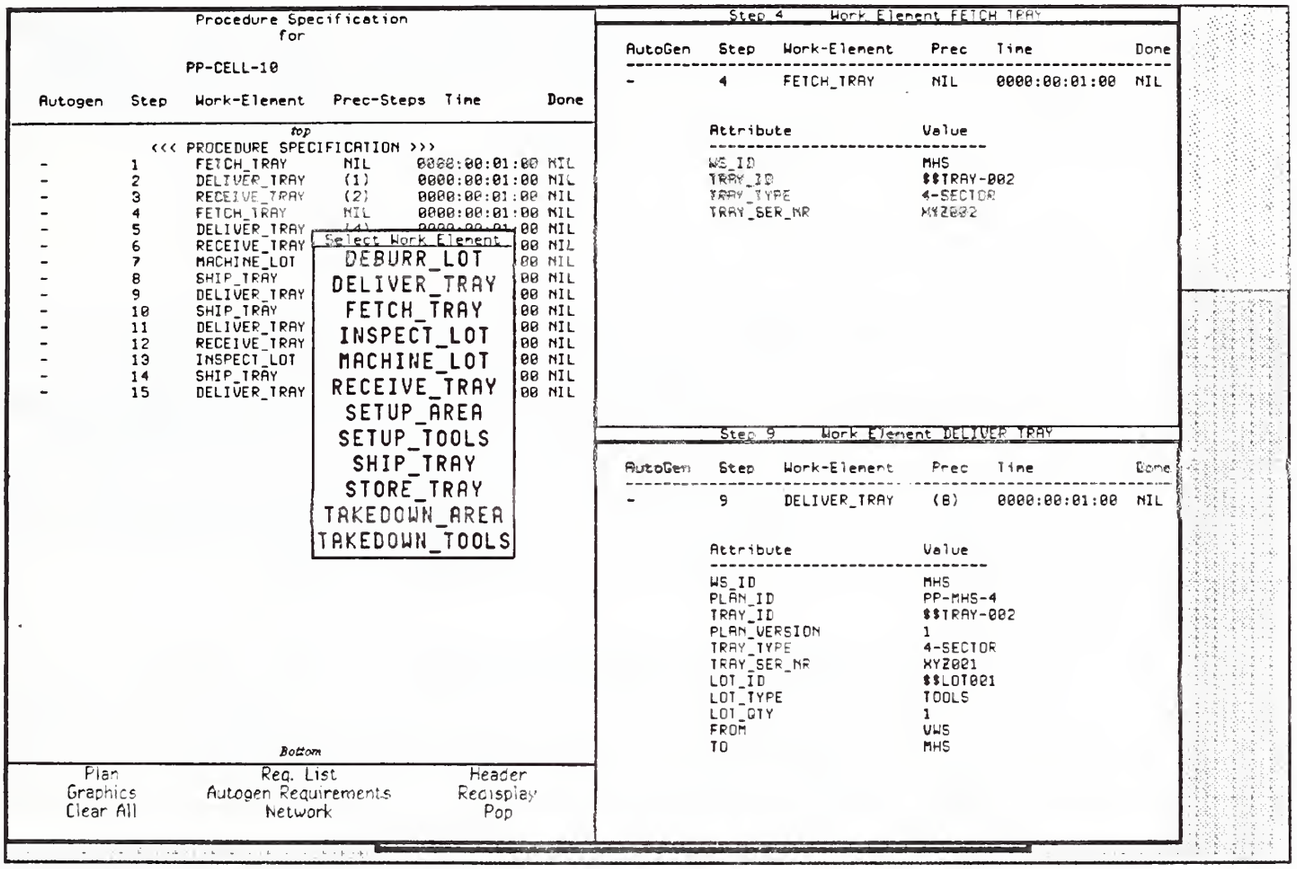


Figure V-4(b). Procedure Specification Editor Menu shows available work elements.

3.2. Database Access Utilities

Currently, the database access facilities are only used by the planning system for storage and retrieval of process plans in the ASCII file format.

4. PART MODELLING MODULE

The process planning system has the capability to read and create an internal representation of a part from the AMRF part model format. This representation is used by the process planning system to determine the machining sequence and requirements of the part to be fabricated within the AMRF.

5. PARSERS AND GENERATORS

Parsers have been developed to create internal representations of both process plans and part models. Care was taken when developing the parsers to separate the low level character reading and manipulation functions from the larger language specific elements. The same code is used as much as possible by both the process plan and part model parsers. Similarly, a set of tools has been developed for writing the internal representation of both the process plan and part model out to ASCII files.

6. EXPERT PROCESS SELECTION MODULE

The final tool to be described in the process planning system is an expert process selection that has been developed in collaboration with Texas Instruments and University of Maryland. The system is known as SIPS for Semi Intelligent Process Selection. Using features obtained from the AMRF part model or defined within SIPS the system will reason about the process or set of processes that will be best suited to fabricate the feature. Figure V-5 shows a screen image of the graphics interface of the SIPS system.

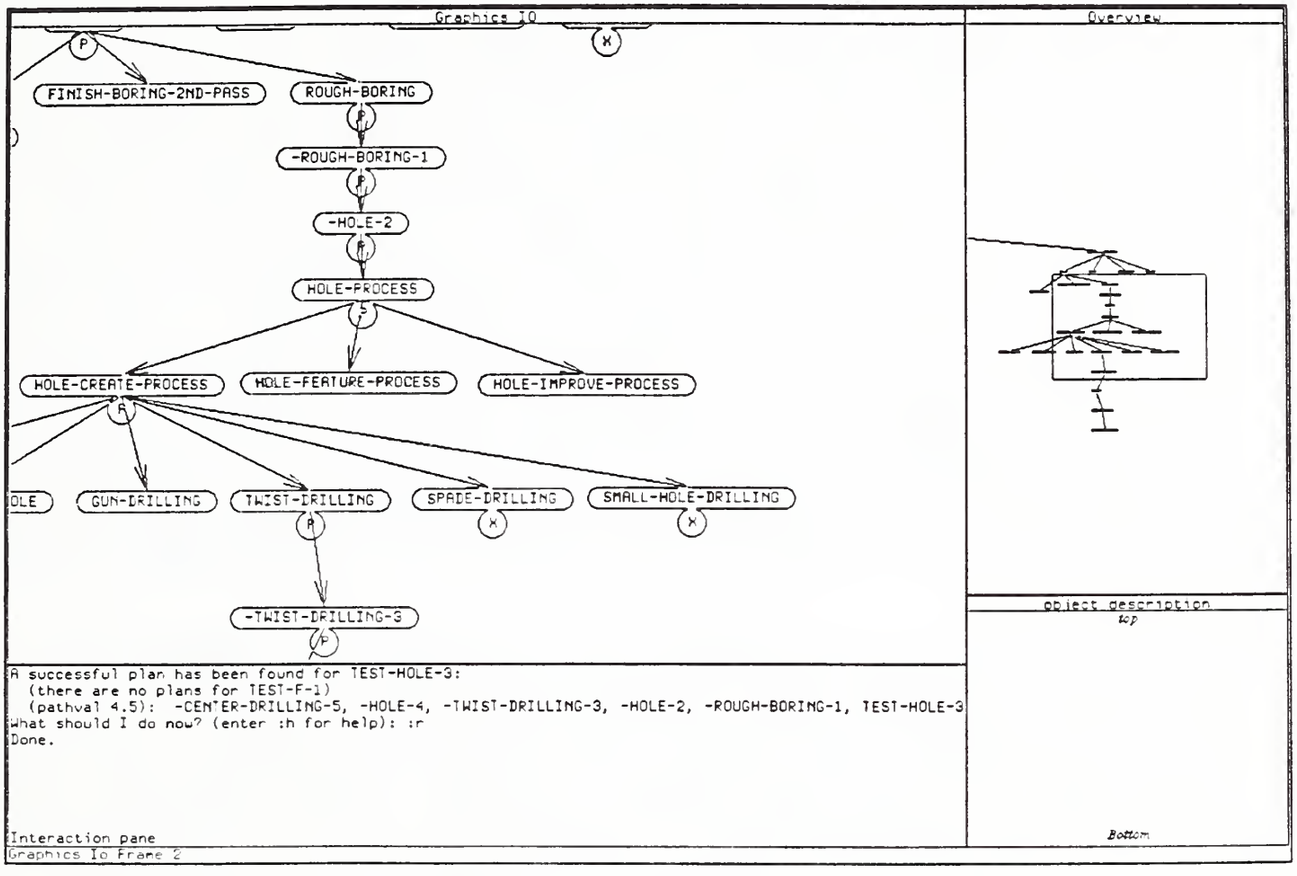


Figure V-5. SIPS (Semi Intelligent Process Selector) Showing Successful Machining Sequence for Creation of a Hole

Blank page

VI. IMPLEMENTATION

1. OBJECT-ORIENTED PROGRAMMING

From the beginning of the process planning project within the AMRF, it had been known that a large portion of the development would ultimately be implemented using expert systems or other artificial intelligence approaches. This approach is necessary because of the large number of decisions which must be made, often from incomplete information. For this reason, the computer chosen as the main development station was a Lisp machine. The environment provided by a Lisp workstation provides useful tools which dramatically increase the productivity of a programmer. These tools include the ability to examine, in detail, every data structure used by the system. Another important feature offered by the environment is object oriented programming. An object oriented approach to programming allows a great deal of flexibility and modularity in tackling a problem. By virtue of the inheritance mechanism of objects, computer code can be re-used many times by different modules. Further, it becomes possible to rapidly generate extremely powerful capabilities by inheriting behavior from other objects. On the Lisp machine, an object oriented approach is used to control the window environment, menus, mouse interaction, as well as any application models written by the user. All of these features and the interpretive nature of Lisp create a development environment which allows rapid development of prototype systems.

The terminology for objects in the Zetalisp dialect used in the process planning system is as follows. The specification of the behavior of a set of objects is known as a flavor. A flavor definition identifies which other flavors are inherited, what local variables exist, (called instance variables), and what initialization actions should occur. In addition, the behavior of a given flavor can be specified in terms of methods. Once a flavor and its methods are defined, an instance of the flavor can be created. This is called instantiation. An instance corresponds to a member of the set defined by the flavor. Thus, a single flavor can have any number of instances. All instances of a given flavor have the same instance variables and methods, but the variables can have different values. Finally, any flavor may inherit the instance variables and methods of any other combination of flavors. Further information on flavors can be found in [5]. In the remainder of this text, it is assumed that the reader has some familiarity with the concepts of object oriented programming.

2. CAPABILITIES DATABASE

Before a process plan can be created, there must be a specification of the factory capabilities and of the shop floor configuration. The planning task consists essentially of programming in terms of work elements for the equipment present in a factory. Thus, there was a need to create tools to allow the specification of a factory in terms of its components and their capabilities. It was felt that these tools should allow the reconfiguration of a factory and specification of new equipment, (or workstations or cells), and new capabilities, all without having to write any code. Thus, much attention was given to having a convenient, intuitive user interface to allow the entry of this data.

2.1. Shop Floor Configuration

Figure VI-1 shows a view of the screen presented when using the configuration tool. In the early phases of development, this tool was called the Choice Tree tool, since it presented the list of work element choices available for each piece of equipment. While this name is no longer particularly valid, many references to it can be found within the code. The configuration tool can be considered a graphical interface to a database. The database contains a description of every entity on the shop floor which is capable of interpreting process plans in the standard AMRF format. It identifies the control links between these entities, classifies them according to the AMRF hierarchy, and, most importantly, identifies the work elements understood by each of the controllers. This tool serves as the dictionary for plan creation and editing. The configuration tool has the capacity to store any number of alternative shop floor configurations, which are called worlds by the system. Thus, before beginning a planning task, the appropriate world is loaded into the configuration tool to identify the factory being programmed.

2.2. Work Elements

When specifying a new work element or editing an existing one, the configuration tool will invoke a secondary tool called the work element editor, shown in Figure VI-2. This tool displays the current definition of the work element being edited or created. Each field is mouse-sensitive and can be altered at will. In particular, the fields labelled "User Property" identify the parameters necessary for the successful execution of a work element. These are specified in terms of attribute-datatype pairs. The attribute name can be anything. The datatype field must be a member of the defined datatypes provided by the work element editor. The valid types are presented in a menu. The fields "Autogen Nodes" and "Autogen Header" are no longer used but are

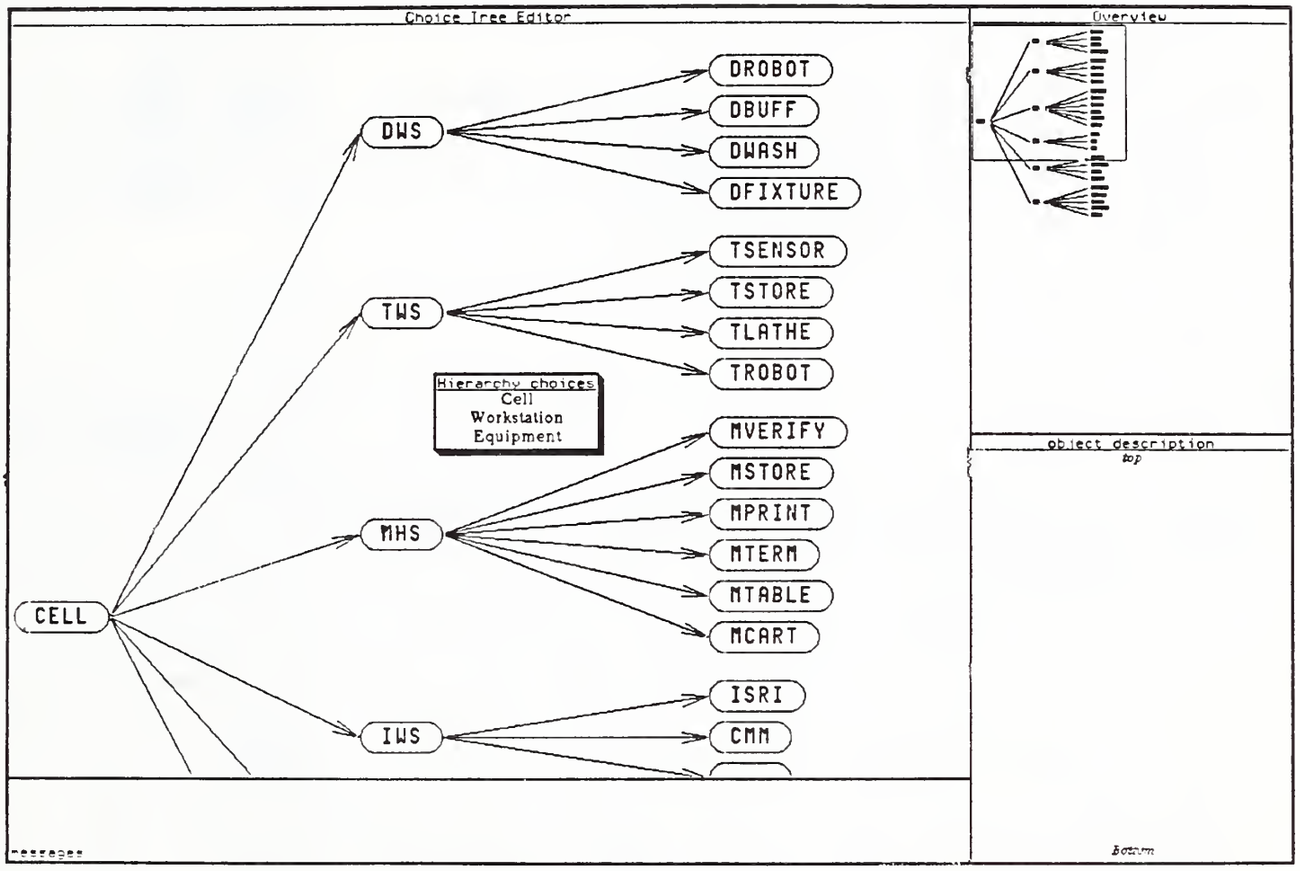


Figure VI-1. Facility Editor

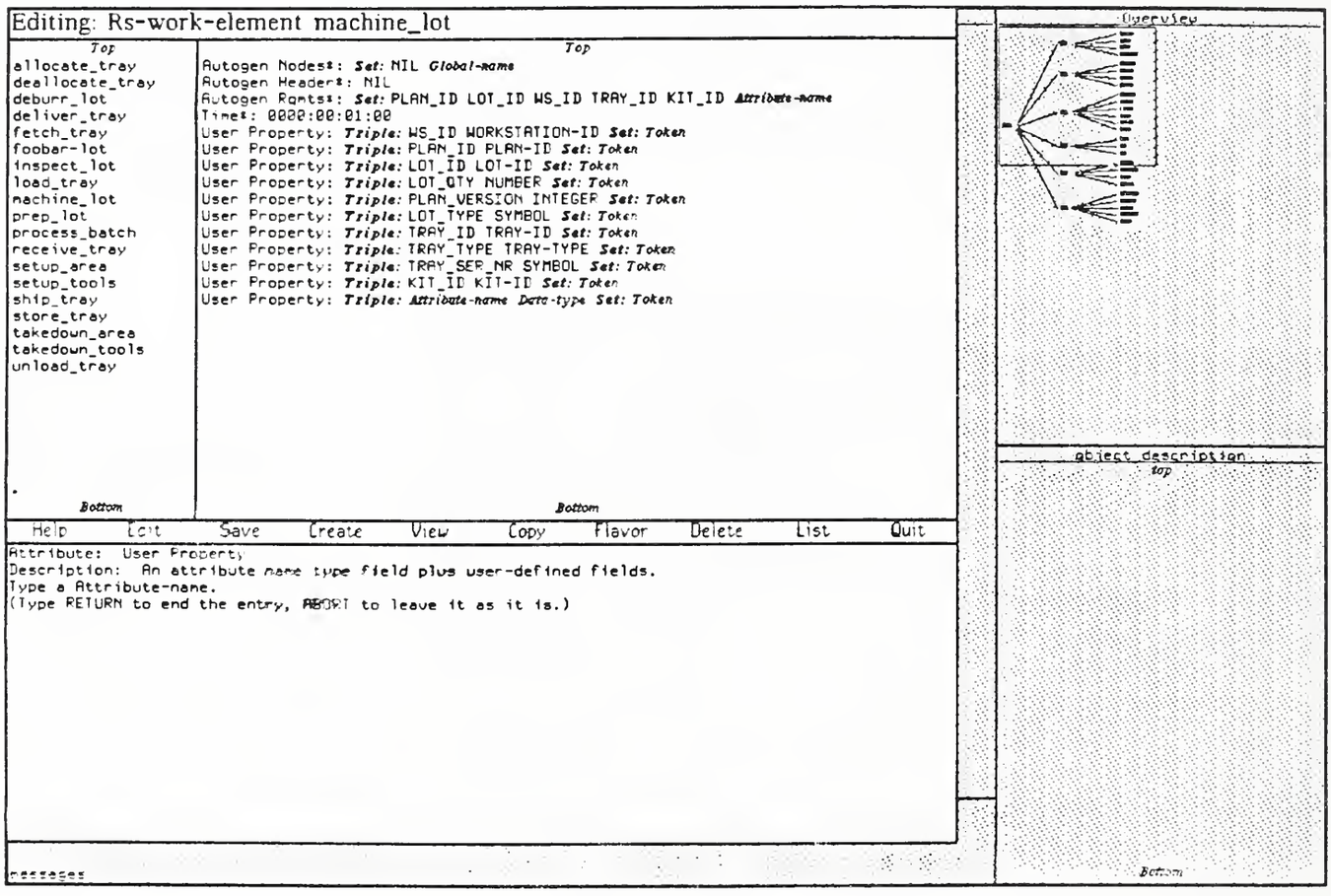


Figure VI-2. Work Element Editor

maintained for backward compatibility with a pre-release version of the software. The "Autogen Rqmts" field is used to identify those attributes whose values correspond to hardware or software which should appear in the requirements list. It should be noted that only attributes with a datatype ending in "-ID" are eligible to be automatically included in a requirements list. This is because the "-ID" suffix is used by convention to identify entities which have a corresponding requirement flavor definition. The "Time" field should contain an estimate of execution time for a work element.

When a work element has been defined, a flavor is automatically created with the appropriate instance variables and methods. During process plan creation, these work element flavors are instantiated to form the nodes in the process plan precedence graph.

3. PROCESS PLAN PRECEDENCE GRAPH

To manufacture a part using the AMRF hierarchical approach, a tree of individual process plans must exist. This tree consists of plans for each level in the AMRF hierarchy; i.e. Routing Slips for cell control, Operation Sheets for workstation control, and Instruction Sets for equipment control. This collection of plans (a "meta-plan") is represented within the planning system as a network of linked flavor instances, as shown in Figure VI-3. Each meta-plan contains an instance of the flavor "Plan" , which the system uses to reference the meta-plan. The plan instance has pointers (implemented as instance variables) to other objects, notably one or more "Routing Slips". Each routing slip refers to zero or more "Operation Sheets", and each operation sheet refers to zero or more "Instruction Sets". In addition, each routing slip, operation sheet or instruction set points to a "Header" object, a "Procedure Specification" object and a "Requirements List" object. The instruction set objects also point to an "End Node" which is also pointed to by the original plan object.

The procedure specification object is itself the head of a precedence graph which describes the sequence of operations to be performed in a given process plan. A diagram of the structure of the procedure specification graph is given in Figure VI-4. As can be seen in the figure, each procedure specification begins with an object called Proc-Specs, and ends with one called End-node. The distinct beginning and end to the graph allow unambiguous traversal of the graph during an editing operation. Between the beginning and end nodes, any number of work element nodes can exist. A similar structure exists for the requirements list, where a work element node is replaced by a requirement object.

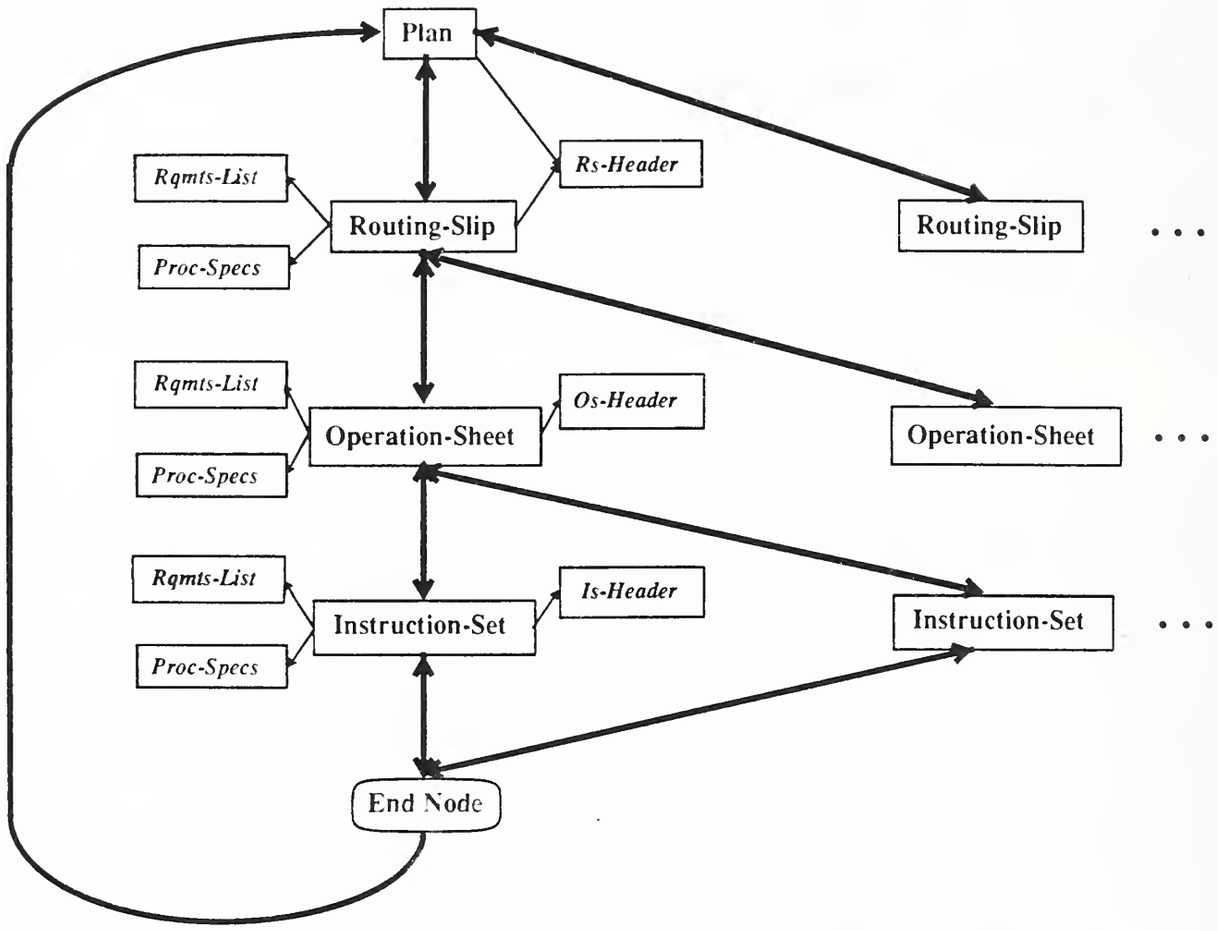


Figure VI-3. Network of Process Plans and Their Elements

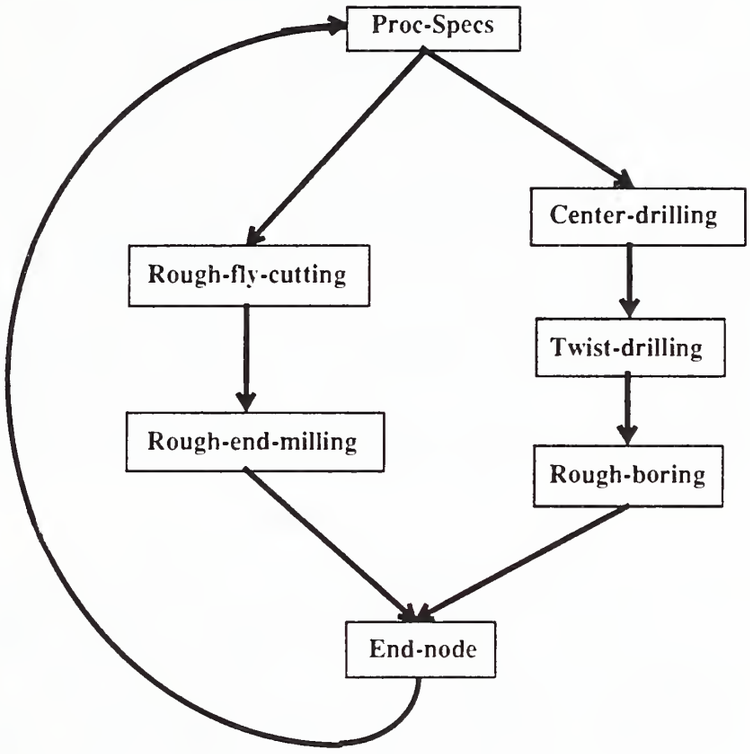


Figure VI-4. Structure of Procedure Specification

3.1. Editing the Precedence Graph

In order to maintain some modularity within the planning system, attempts were made to separate the internal modeling of a process plan from the user interface functions needed to allow human editing of the plan. To accomplish this, a supervisor object was created to control the flow of editing commands to a plan network. The flavor inheritance graph for the supervisor is shown in Figure VI-5. The flavor which is instantiated to perform the duties of a supervisor is "Plan-Sup". It maintains a record of the current plan being edited, the level within the AMRF hierarchy of the plan and other bookkeeping information. These capabilities are inherited from "Bookkeeping-Mixin". When a user wishes to alter a plan, a request is sent from the user interface code to the supervisor to perform the necessary alteration. It is the job of the supervisor alone to maintain consistency of a plan and to perform the actual alterations to the internal model. In this way, the user interface system has a set of specific messages it can use to accomplish changes, leading to a well-defined, controlled environment. This approach proved to be particularly important in a team programming environment, where one team member is usually unaware of the details of implementation of another member's code.

4. INTERNAL PART MODEL REPRESENTATION

The planning system has the capability of maintaining an internal representation of a part, as represented in the AMRF part model format. This internal representation is not a full solid model of the part, but it does contain all pointers between connected faces, edges, vertices, tolerances and features. This information is accessible to the planning system by querying the object which is the value of the variable "plan:*part*". It is this representation which allows an expert system such as SIPS to automatically generate process steps when given a feature description. The feature is identified by the feature-id from the part model file. The planning system parses the part model file if necessary, prepares the description of the needed feature from the internal model, and hands it to SIPS.

5. DATABASE FACILITIES

A major function of the process planning system is to store and retrieve process plans from a database of plans. In the current system, plans can be stored and retrieved from either a local file system, or via the network from the IMDAS. To accomplish the latter, several protocol layers are needed. The plans and the database requests are handled using the TCP-IP medium over the ethernet, with the Nfile protocol developed by Symbolics, Inc. On top of this layer, the system uses common memory, described in section 5.1.2. The Symbolics machine itself does not directly

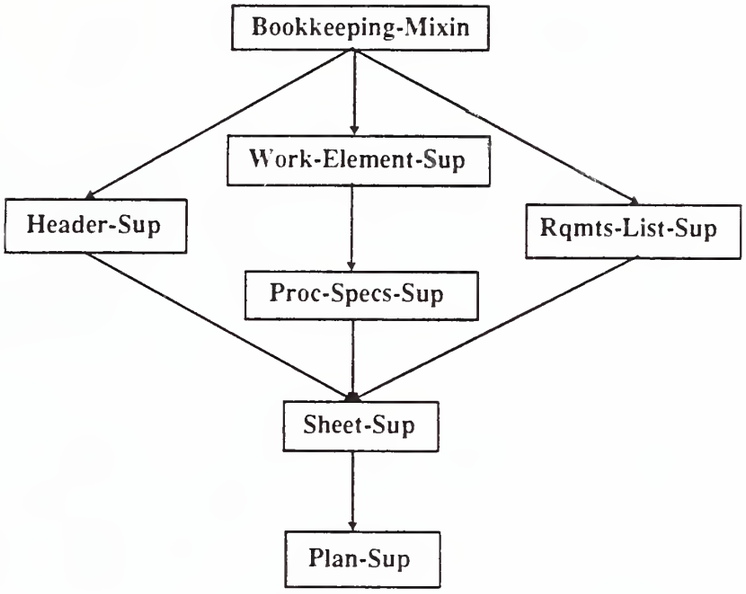


Figure VI-5. Flavor Inheritance Graph for the Supervisor

support the AMRF common memory implementation. Rather, it communicates with a Sun-3 workstation using Nfile, and the Sun handles all the common memory manipulations, reporting back to the Lisp machine. This approach was adopted for ease of implementation, rather than elegance. Layered over common memory is the mailbox protocol, which is simply a formatted common memory variable. Finally, Data Manipulation Language (DML) requests are placed in mailboxes to communicate with the distributed database system.

5.1. Protocols

5.1.1. Nfile

Nfile is a file transfer protocol developed at Symbolics, Inc. as an alternative to TCP-FTP (Transmission Control Protocol-File Transfer Protocol). It is a token-based protocol which works with TCP-IP as well as other media. For the implementation within the AMRF, an Nfile server implementation was installed on a Sun workstation to allow for transactions between the Sun and the Symbolics. This was done to enable process-to-process communication between the two machines. Such communication was not possible using the TCP support normally sold for use on the Symbolics machine. New services defined between the machines include "db_initialize_or_startup", "db_access" and "db_status". The first performs the University of Virginia startup calls for configuration management [9], the second handles the actual database calls, and the third is used to query common memory for the status of any previous database transaction.

5.1.2. Common Memory

The common memory system currently runs on a Sun workstation to support the planning system running on the Symbolics machine. The implementation is such that the requests come from the Symbolics via Nfile as mentioned above. The Sun server performs the necessary common memory mailbox declarations (declaring mailboxes named "DS_PP_CMD" and "DS_PP_STS") and reads and writes to these mailboxes. The common memory implementation looks identical to other Sun implementations of common memory elsewhere in the AMRF. The entire Sun server process is started by running the executable /usr2/nfile/nfiled as super-user. This must be done before invoking any database queries from the Symbolics. The file "db_access.h" contains the name of the common memory manager host which will be supporting the common memory. If this is changed, a new executable must be linked.

5.1.3. DML

The DML transactions used by the planning system are in accordance with the DML specifications described elsewhere in the AMRF documentation set. The queries used are SELECT and INSERT although nothing should prevent other queries being used in the future. Because of the large size of process plans, the plan text is communicated to the database via files, rather than putting it in a mailbox. Thus, the DML queries are stored in mailboxes but refer to normal files where the plans themselves are stored. These files reside on the Vax 11/780 system. The two files of particular importance to the planning system are "/user1/ray/db_buffer" and "/user1/ray/db_out". The former is used for storing plans into the database, while the latter is used for plan retrieval. Thus, to store a process plan in the database, the planning system first transfers the text of the plan to the file "db_buffer". Then it places a DML request in the mailbox DS_PP_CMD requesting an INSERT operation, referring to the file "db_buffer". The database status is read from the mailbox DS_PP_STS. Conversely, when retrieving plans, a DML SELECT message is placed in the command mailbox, then upon successful completion, the plan is read from file "db_out".

6. USER INTERFACE

A significant amount of work went into the development of an intuitive user interface design to support the planning system. This was deemed important since ultimate users may not have much computer experience. Thus, at all times, the user is shielded from actually editing Lisp code. Indeed, the actual language used for the implementation is not readily identifiable from the users point of view. Examples of the screens presented to the operator are shown in Figures VI-1, VI-2, VI-6 and VI-7. Details of the use of all of these screens are given in the Users Guide [1]. The general philosophy was to rely heavily on mouse interaction and graphical displays to represent plans, work elements, workstations and equipment.

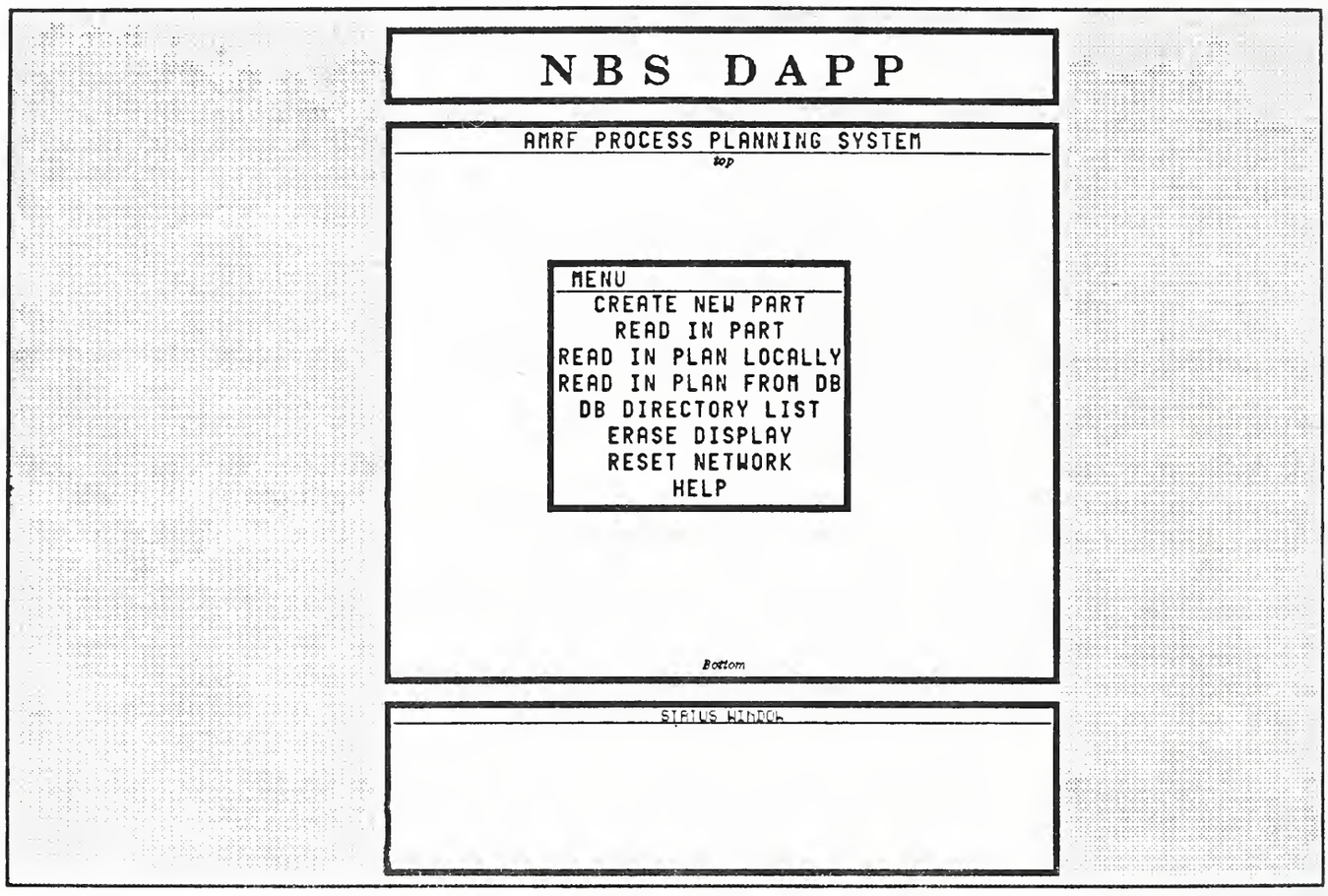


Figure VI-6. Process Planning System Operations Menu

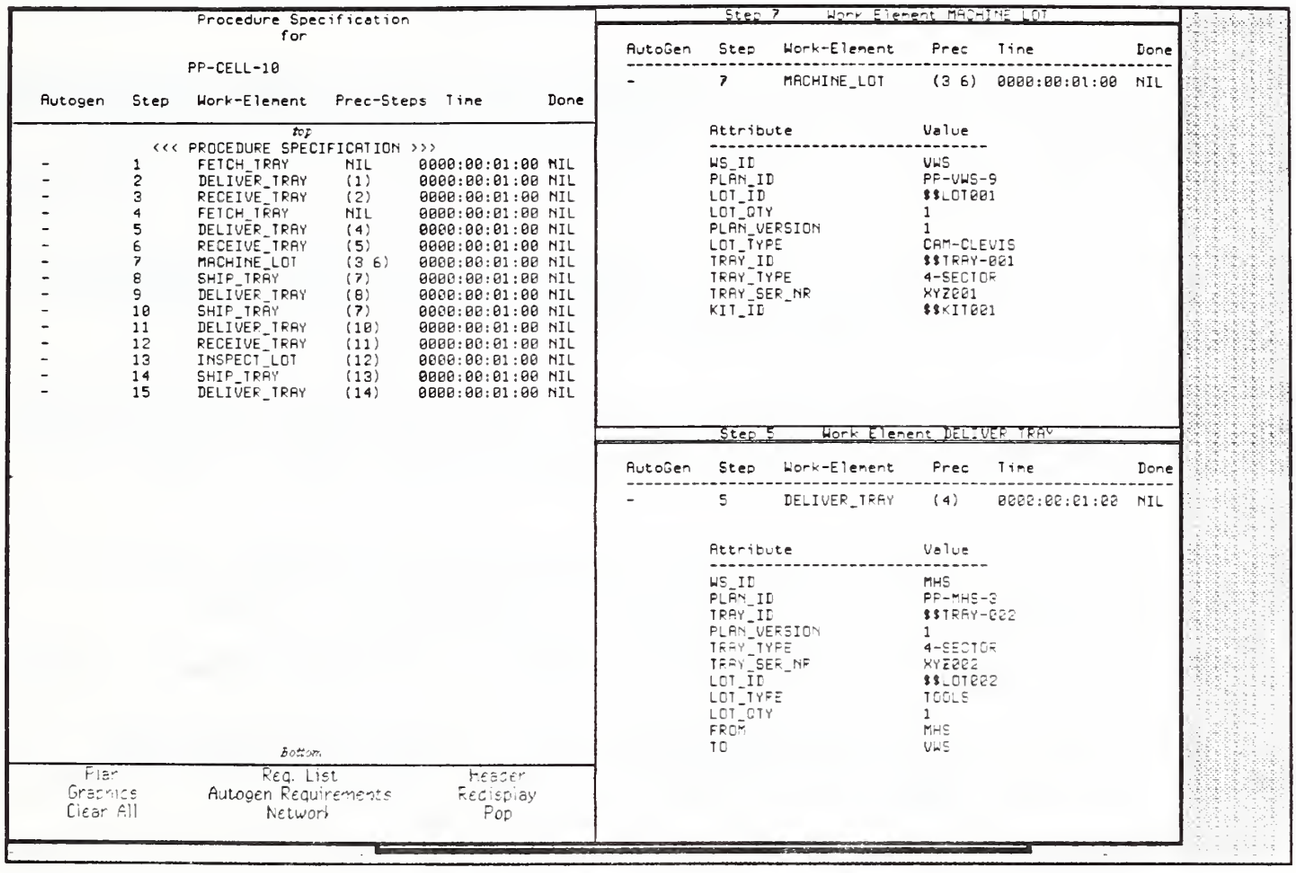


Figure VI-7. Procedure Specification Editor

[1] Ray, S., "NIST - AMRF Process Planning System - User's Guide", NIST internal report, 1989.

[2] Hopp, T.H., "AMRF Database Report Format: Part Model", NBSIR 87-3672, National Bureau of Standards, Gaithersburg, MD, September 1987.

[3] Chang, T. and Wysk, R. "An Introduction to Automated Process Planning Systems," Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

[4] Hummel, K., Brooks, S., "A Structure for the Representation of Manufacturing Features in the XCUT Process Planning System", Proceeding of the 1986 Winter ASME Conference, Anahiem, California, December 1986.

[5] Nau, D.S. and Chang, T.C., "Hierarchical Representation of Problem-Solving Knowledge in a Frame-Based Process Planning System", Journal of Intelligent Systems, Vol. 1 (1986).

[6] Libes, D. and Barkmeyer, E., "The Integrated Manufacturing Data Administration System (IMDAS) - An Overview", International Journal of Computer Integrated Manufacturing, Vol. 1, No. 1, January 1988.

[7] McLean, C.R., AMRF Systems Architecture Document, Draft Technical Report, National Bureau of Standards, (1986).

[8] Symbolics Reference Manual Volume 2a, Symbolics Inc. (1986).

[9] O'Halloran, D.R. and Reynolds, P.F., "A Model for AMRF Initialization, Restart, Reconfiguration, and Shutdown," NBS/GCR 88-546, May 23, 1986

This Appendix lists a selection of relevant published papers supporting the process planning project in the AMRF. The titles are:

"Interactive Process Planning in the AMRF" by Brown and McLean.

"A Knowledge Representation Scheme for Processes in an Automated Manufacturing Environment" by Ray.

"Research Issues in Process Planning at the National Bureau of Standards" by Brown and Ray.

"Hierarchical Abstraction of Problem-Solving Knowledge" by Nau.

Interactive Process Planning in the AMRF

Peter F. Brown
Charles R. McLean

Factory Automated Systems Division
Center for Manufacturing Engineering
National Bureau of Standards
Gaithersburg, MD 20899

Presented By:

Peter F. Brown
Charles R. McLean
ASME Winter Annual Meeting, Anaheim, California
December, 1986

Bibliographic Reference:

Brown, P. F., McLean, C. R., "Interactive Process Planning in the AMRF", Bound volume of the 1986 ASME Winter Annual Meeting, Anaheim, CA, December 1986

INTERACTIVE PROCESS PLANNING IN THE AMRF

Peter F. Brown
Charles R. McLean

Factory Automation Systems Division
Center for Manufacturing Engineering
National Bureau of Standards
Gaithersburg, MD 20899

ABSTRACT

As more intelligent automated control systems are introduced into discrete parts manufacturing facilities, it will become increasingly difficult to maintain the centralized process planning systems in use today. A new approach is required that will permit distributed manufacturing operations planning via a network of cooperating, intelligent, process engineering systems. There are a number of reasons why manufacturing process decisions should be made locally by planning modules that are fully aware of a controller's current or expected capabilities. Expert planning modules should be developed for each controller or class of controllers that are or will be used in manufacturing installations.

To accomplish this goal of distributed, intelligent planning modules, work has started with the development of a semi-automatic interactive process planning system. This system has several unique features. First, a hierarchical planning system has been developed for multi-level factory architecture. Second, all activities within the factory are described by work elements. A work element is an activity at some level of the factory for which there are well- defined constraints. Third, standard interfaces have been defined to allow the passing of information between planning modules and controllers. These interfaces are used for the organization of the data and not for the data itself. Fourth, a semi-intelligent editor for the manipulation of these process planning data structures. These tools include editors for defining work elements and manipulating the process planning data structures. A graphic network editor is used for defining the "Precedence Graph" of a process plan. All system editors are based on windows and menu selections.

Interfaces to factory-wide databases for retrieval of information, such as raw stock and tooling, and CAD/Solid Modeling databases are under development. This last interface will serve three purposes: 1) the input of the initial part geometry to be manufactured, 2) the verification of changes to part geometry by the process engineer, and 3) the storage of intermediate geometries to be passed to other factory systems (inspection, machine tools, robots, vision systems, etc.). This paper describes research efforts at the National Bureau of Standards (NBS) by the staff of the Distributed Automated Process Planning System (DAPP) project to define and test this information processing architecture in the machine shop environment of the Automated Manufacturing Research Facility (AMRF).

1. INTRODUCTION

In designing a process planning system for the AMRF, the primary issue was not whether the system should employ variant or generative techniques. The most important concern was to identify the fundamental architectural concepts that would best support process planning in a small batch manufacturing facility where all production operations are under direct computer control. Important research questions deal with the functional relationships and the data interfaces between manufacturing control and planning systems: 1) How should planned tasks be specified to controllers? 2) How should alternatives be described? and 3) What formats should be used to pass data between the planning system and the controllers? The AMRF project involves developing a testbed for factory automation research to define and test the system interfaces between modules like process planning, geometric modeling, manufacturing control, data administration, network communications, and other factory subsystems. Within the AMRF, process planning is designed to be one of the primary programming tools of the factory. This paper describes the efforts of the AMRF process planning project to define robust interfaces to support both the future development of interactive process engineering tools and automated intelligent process planning systems.

Current Philosophies in Process Planning

There are two basic types of process planning systems in use today: variant and generative. Variant planning systems are based on a library of standard plans for different part families that a process engineer retrieves and edits, creating "variants" of basic plans. Generative planning systems employ expert system concepts, they reason using embedded knowledge and problem solving techniques to develop new plans. For a more detailed discussion of the state of the art of computer-aided process planning systems, see Chang and Wysk [1].

Variant systems typically rely on group technology classification and database management systems for their implementation. Standard process plans are developed for each family of parts produced and are stored in the database. When a new part enters the system, it is first classified by part family. The part classification code is used as a key to select a copy of the appropriate default plan from the database. This copy is then modified to reflect the specific processing required due to the unique characteristics of the new part. If a plan does not exist for the part's family, then a new default plan is created by an experienced process engineer and stored in the database system.

The technology that is required to implement this type of process planning system is readily available on main frame as well as personal computer systems. Indeed, almost all of today's commercial process planning systems employ variant techniques. With this approach most knowledge resides in the mind of the process engineer, the computer serves mainly as an organizing tool. Although intelligent generative systems are often more desirable, there are significant benefits that can be obtained from the variant approach. The development of a variant system forces an organization to study and classify the activities that it can perform in order to understand the part families it can produce in its shop. This exercise, in turn, reveals the kinds of equipment and labor skills that the shop really needs. But, there are some limitations in the variant approach. It can often be impractical if the shop produces small batches of widely varying parts. More time has to be spent defining new part families

and modifying default plans. Furthermore, it does not capture the real knowledge or expertise of process engineers. The generative approach to process planning does address these issues.

The main thrust in process planning research today is in the area of generative systems, for some examples see [15,17]. In these systems, artificial intelligence is used to automatically create a plan for a new part. An expert problem solving system uses an internal process knowledge base and part specific data to generate new plans. This approach requires that a full product definition or part model is encoded in the system in a form that is accessible by the expert system software. This model should include geometry and topology, a tolerance model, and information about the functionality of a part. The knowledge base contains information gathered from process engineers on the how and why of making process decisions for various types of parts. Decisions are often keyed to the different types of features that are typically produced on parts. With this approach, the knowledge base becomes a repository of knowledge gained from the many years of experience of many process engineers. It also permits the separation of the process knowledge from the part data, facilitating data driven automation.

To date, fully generative process planning has proved to be an elusive goal, but there are some signs of progress. The biggest problems have included the representation of features (pocket, slots, holes), processes (drill hole versus bore hole), and sequencing information (make pocket before hole). Furthermore, the outputs produced by process planning systems are non-standard. That is, the organization of data into forms or structures such as routing and operations sheets differs from system to system. As a general rule, plans are meant to be interpreted by human readers, rather than by automated control systems. In the future, it will be essential that process planning interact more closely with automated control systems. Major questions with respect to the inputs and outputs of process planning systems must be resolved before fully computer-integrated intelligent manufacturing systems become a reality.

The AMRF process planning project is tackling questions that concern the fundamental role of process planning in automated manufacturing facilities. Important areas to be addressed include: 1) the definition and parameterization of activities or processes, 2) the development of standard definitions for both design and manufacturing features, and 3) the establishment of a data representation scheme that can be used to organize and exchange information between planning, control and other factory systems. The AMRF process planning project has developed a number of workable solutions in these areas.

AMRF Process Planning Concepts

A primary goal of Automated Manufacturing Research Facility (AMRF) which has been established at the National Bureau of Standards (NBS) is to develop a small batch manufacturing system to support research and experimentation in automated metrology and interface standards for the factory of the future [2,3,4,5,6]. Since process planning is expected to become one of the primary tools for programming automated factories, its system interfaces are of great interest. Unfortunately, the conventional views and implementations of process planning systems are inadequate to support such a factory. The research approach at NBS focuses on identifying basic concepts that would support the integration of

process planning directly with the software and hardware of manufacturing process control systems.

Presently, the AMRF is comprised of six manufacturing workstations which perform both production and support functions. Each of the three machining workstations has a numerically controlled machine tool, a robot manipulator, flexible part fixturing systems and local storage for tools and materials [7,8]. Another station, cleaning and deburring, has two robots, cleaning equipment, and buffing wheels. The inspection workstation contains a robot, a coordinate measuring machine, and surface roughness characterization device. The last workstation level system, the material handling system [9], consists of two automatically guided vehicles (AGV), trays for parts and tooling, a storage and retrieval system, tray roller tables in the workstations, and a tender area for manual support activities. Finally, all workstations have a controller consisting of one or more small computer systems and associated software.

Other major factory systems found within the AMRF include: a cell control system, user interfaces for design and modeling, process planning and off-line programming systems, a data administration system and a communications network. The major difference between the systems found in the AMRF and in conventional advanced manufacturing systems, is the number of different systems vendors involved. Manufacturing subsystems were consciously chosen from many different vendors to shed light on the "plug compatibility" problems that would be faced by industrial system integrators.

A major effort is underway within the AMRF to integrate the factory systems, identified above, into a single automated manufacturing environment. This integration will be accomplished using some of the hierarchical task decomposition techniques and real-time sensory interactive control concepts originally outlined by the robotics project at NBS [5,18]. With this approach, all control modules are arranged in a hierarchy. Each controller takes commands from only one higher level system, but it may direct several others at the next lower level. Long range goals enter the system at the highest level and are decomposed into subgoals to be executed at that level or passed down as commands to the next lower level. Status information, based on real-time sensory data collection, is generated at each level and is passed up as feedback to the next higher level. The preparation of planning data, that will enable these hierarchical control systems to achieve their goals, is the primary role of process planning.

The AMRF process plan data structures are intended to be generic so that they can be used in a variety of manufacturing organizations from small shops to large factories. Process plan data structures have been defined, using formal language specification techniques, that can be transmitted electronically between planning and control computers. Although the formats are quite readable, they could easily be enhanced by print formatting routines to be made more suitable for human interpretation and execution.

By defining standard process planning data structures, an organization will be able to develop planning systems in a modular fashion. An interactive plan editing system can be developed initially. Later expert planning modules can be added without a change to basic data formats or execution system architectures. Another important benefit of standard data structures is that it permits the implementation of planning systems by multiple independent

developers. It will also allow for the design of intelligent control systems that will be able to accept these standard process plans. By taking this approach, many organizations may be able to participate in the development of planning and control systems. Each developer could focus his efforts on developing specialized intelligent planning capabilities, building upon the programming work of others.

Our approach has focused on first defining process planning data structures that could eventually be used to construct a distributed generative process planning system. Such a system would involve the dynamic interaction between intelligent planning and control systems at each level within the AMRF. A number of interface issues between the planning and control systems must be resolved. Some of the interface issues that fall within the realm of the process planning project include: 1) the development of a feature-based representation of part geometry to be used as an input to process planning, 2) the specification of a plan syntax to be used as a neutral file format for transferring plans out to target control systems, and 3) the definition of basic work elements, i.e. generic or specific manufacturing activities that each control system is capable of executing.

The work element is the basic procedural entity in the AMRF planning and control system. The work element is a function or activity which is carried out by a manufacturing control system at a particular level in the factory hierarchy. A work element has a name, a set of parameters, a duration, and a list of precedent steps numbers. The numbers identify the steps in the plan that must be performed prior to this one. Work elements are parameterized and organized into procedure specifications within process plans. The parameters of complex work elements, usually performed by higher level systems, refer to lower level process plans. These process plans specify the decomposition of the complex activity into simpler work elements supported at the next lower level in the hierarchy. Within controllers, work elements are implemented as subroutines that carry out error checking, database transactions, as well as physical changes to the manufacturing environment.

Generic Data Interfaces

A major goal of the AMRF project is the identification of generic functions and data structures for advanced manufacturing systems that could be used as a basis for the development of industry-wide interface standards. Generic interfaces, which are relevant to process planning, have already been defined and implemented within the AMRF to support interaction between a diverse set of applications processes. A communications mailbox protocol has gives AMRF applications processes access to each other over the communications network [10]. A control command-status protocol [10] has been developed which provides a means by which supervisory controllers can assign production work orders to subordinates and receive feedback status. A work order management system, described in [10], has been implemented in which process plans are used to specify the decomposition of complex jobs into simpler tasks [10]. A level independent neutral process plan file format has been developed for transferring this data between planning and control systems [12]. Process plans are deposited in a common database for later retrieval and execution by automated manufacturing control systems. A generic interface to the common database [13] has been created to give control systems ready access to required data, such as: command and status messages, work orders, process plans, control programs, geometry descriptions and other reference data.

Although there are many differences between the automated control systems found at each major level in the AMRF hierarchy, they all seem to have some functions and responsibilities that are characteristic of project managers. Hence, project management concepts have provided a foundation for defining the behavior of planning and control systems within the AMRF. Project managers, regardless of their level within an organization, tend to perform some generic planning and execution functions. Typical functions include: 1) work decomposition or problem reduction -the breakdown of complex activities into a interdependent network of simpler ones that can be routinely carried out by subordinates, 2) resource management - the identification, acquisition and allocation of required resources, and 3) estimation or prediction - the analysis necessary determine project cost, time and quality trade-offs.

Project managers often use network scheduling tools such as critical path method (CPM) or program evaluation and review technique (PERT) to define, sequence and monitor project activities. A detailed discussion of PERT, CPM, and other project management methodologies can be found in [11]. The data that is typically required by these systems includes: activity specifications and precedence relationships, resource requirements, time and cost estimates. With the exception of cost estimates, the process plan file structure is designed to convey this information to control systems.

A process plan is comprised of four major sections: 1) Descriptive Header - contains static index and summary data, 2) Parameters - lists all variables for which real values must be substituted at execution time, 3) Requirements List - identifies all resources to be used during the execution of the plan, and 4) Procedure Specification - describes all work elements, their precedence relationships, and their attributes and specific value bindings. The next sections are devoted to a discussion of the process plan format.

Procedure Specification

The Procedure Specification is probably the most important section of the process plan, it describes not only all of the activities or work elements to be performed, but gives information about their order of execution. This information can be represented as a precedence graph. Figure 1 shows the precedence graph for the machining operation to be performed on a part. This graph allows the process engineer to explicitly state that some steps may be done in any particular order, allowing for parallel activities, while other have a strict sequence or precedence relationships. The nodes of a precedence graph are the work elements. The graph structure used to represent process plans permits the specification of alternate activity sequences. Intelligent control systems can use this information to continue the manufacture of a part when some forms of error conditions arise. The control system can search through the precedence graph to see what other nodes or step can be performed while notifying a supervisor of any unresolvable problems, a major step in integrating sensory feedback with intelligent manufacturing planning and control systems.

The precedence graph in figure 1 represents the process plan for the part shown in figure 2. Figure 2a shows a part for which a process plan is to be written; the part is broken down into a feature graph [14,15], which defines features (such as pockets, grooves, holes) and the access. The access defines which features block or cover other features. Once the features are determined one can define the procedure specification as to how to produce the

part. Figure 1 shows the order in which we wish to produce the work elements (nodes of the graph): INIT, CHAMFER_OUT, POCKET, GROOVE, CHAMFER_IN, HOLE, and CLOSE. These work elements correspond to the features defined in the feature graph. The precedence relations, as drawn in the precedence graph, can be interpreted to mean that after initializing the machine (INIT), the next step could be either CHAMFER_OUT, POCKET or GROOVE, in any order (in the feature graph these features do not interact, so they could be produced in any sequence). But before either the CHAMFER_IN, or the HOLES, can be produced, the POCKET operation had to be performed. It is important to point out here that the holes could have been produced before the pocket, but the process engineer decided that it would be best to produce the holes after the pocket. Thus the precedence concept is used to limit or structure the machining sequence. This graph can now be linearized by various constraints, such as minimizing tool changes, tolerance stack-up, etc.

This is the highest level that the process engineer will deal with a single part, in terms of the manufacturing features. These features will then be decomposed into a set of machining activities that are best suited for the constraints on a feature (such as its tolerance attributes). Using the previous example the hole feature may be produced by a simple twist-drilling operation. If the hole feature required tight positional and roundness tolerances, several machining steps might be needed, such as: center-drill, twist- drill, and reaming. Using the process planning editor, the process engineer will first define the part features. Then using an expert process selection module, the features will be decomposed into a set of machining process steps. The output of the expert system is in the process plan format [12]. This will allow the process engineer to modify the individual processes, as well as to monitor the specified processes.

The procedure specification contains the information about the sequence of the operations to be performed. Work element parameters reference hardware systems and software data objects used in the performance of a particular process. This information is consolidated into the Requirements List section of the process plan.

Requirements Lists

The requirements list section contains a list of all the hardware and software needed to execute the procedure specification we have just described. This structure has a similar function to that of a bill of materials. When a plan is executed in the AMRF, a controller can check to see that all items listed in the requirements list are available before executing the procedural steps of a process plan. This section of a process plan could also be used by the scheduler to determine that all items are available before the plan is even released for production. The requirements lists also identifies all other process plans referenced in the procedure specification.

In an effort to make the process engineers job easier, the generation of the requirements list can be done automatically. When the process engineer defines a work-element, there is a procedure for identifying items that will be added to the requirements list. Upon completion of the procedure specification, the system supervisor will query each node to ascertain what items it requires to perform its task, and these items are then added to the requirements list. Currently, there is only minimal checking for duplication of items. After the system has generated the requirements list, it is available for editing or viewing by the

process engineer.

In the example given in Figure 1, the requirements lists would contain a list of tools, process plans, control programs (N/C, robot, or inspection), fixtures, robot grippers, etc. Figure 3 shows the major fields of each entity in the requirements list. The fields are a label, a descriptive name, a set of attribute-value pairs, and pointers to any sub-elements of an item. This pointer item is used to describe assemblies or complex items.

Parameters Section

The current implementation of the process planning system is an interactive system, process plans are prepared off-line. The parameters section allows the engineer to specify in a symbolic way that a particular item is to be used, but does not actually specify a serial number (i.e. specify plan variables). In a simple example, the process engineer wishes to specify that a 1/2" 2 fluted endmill should be used for a milling operation. At the time of plan creation the process engineer could identify this tool as $$tool- 001 (the syntax of a process plan has all parameters preceded by $$), and when the plan is being executed $$tool-001 will be replaced by the actual serial number of the physical tool. In this way the process planner can specify completely how a job should be done without overly constraining the execution. It permits the passing of information that is useful to the work-element software, but is not known at planning time, rather only at run-time.

Header Section

The header section contains certain bookkeeping information used to index or catalog the plan. As the planning system discussed in this paper is dynamic, not all of the entities listed here are fixed. Several of the fields in the header are used by the data administration system as keys for retrieval of plans. These fields are PLAN-ID, PLAN-VERSION, PLAN-TYPE, and PLAN-NAME. In addition there are other header fields that will be used to keep track of important information such as PROCESS-ENGINEER, PART- NUMBER, GT-CODE, ENGINEERING-DRAWING-#, etc (See figure 3).

In order to be consistent and save on the duplication of work needed to read process plans, all levels of process plans use the same internal format. It is important to reiterate here that these planning structures are used to organize the data, not to limit the specific data that appears in the process plan. As long as one accepts the process planning data structures and their associated formats, the user can define or associate any kind of functionality to work elements that he/she wishes. The next section will discuss the file format developed to exchange information between planning systems and other systems (controllers, databases, etc.).

2. FILE FORMAT

As part of interface standards work, a method has been developed for exchanging process plans between various systems. Using formal language specification techniques (Backus-Naur), a grammar has been defined for the process plan data structures. Using this grammar, an ASCII file containing the process plan can be generated (for an example see figure 3). This file can be passed between various computer systems, translated back into a control systems internal representation of activities to be performed. It is then used to

sequence the part through manufacturing. To test our specifications we have developed and written parsers in several languages to construct the appropriate data structures. The file exchange format is quite human readable as we make liberal use of formatting when writing the ASCII file.

3. PROCESS PLAN HIERARCHY

The same basic structure is used for process plans at all levels of the factory. In the AMRF, currently only the lowest three levels of control are operational: Cell, Workstation and Equipment. The names that have been given to the classes of plans at these levels are, respectively: Routing Slips, Operation Sheets, and Instructions Sets (see Figure 4). The role of the plans at each level is described in subsequent sections.

Cell Routing Slips

Cell routing slips are used to coordinate the movement and processing of materials, parts, tools, and other needed items between and at workstations. A brief example will best illustrate this idea, (see figure 5). In the example the cell control system is told to deliver a tray of parts and one of tooling to the vertical workstation; the vertical workstation is told to receive the two trays, then to setup the tooling area, machine the lot of parts, takedown the tooling area, ship out the trays and, then to finally have the material handling system deliver the trays to some other AMRF system. Each one of the nodes in the graph represent a work element. The node MACHINE_LOT will decompose into an activity at the next level in the AMRF, which is the workstation level. The process planning data package at this level is known as the Operation Sheet.

Workstation Operation Sheets

The next lower level of factory control is the workstation level. Process plans at this level are used to coordinate equipment level activities. The MACHINE_LOT work element, from the previous example, can be decomposed into an entire operation sheet. Figure 6 shows a simple sequence of MOVE_PART, MACHINE_PART, and MOVE_PART, which would be repeated for the number of parts that are in the lot. MOVE_PART, which is used for loading a part into a fixture, involves the coordination of the robot, machine tool, and fixturing system. Finally, this level of the process plan is decomposed into a sequence of tasks for the equipment to perform. The decomposition of the MACHINE_PART work element provides a good example of the next lower level, the Instruction Set.

Equipment Instruction Sets

The bottom level of the process planning hierarchy is the Instruction Set. This is a detailed sequence of operation for an equipment level to perform. The example given here is the MACHINE_PART work element that is carried out by the vertical milling machine. Primitive work elements appearing in the Instruction Set describes features such as pocket and holes that are to be produced. Figure 1 shows the machinable features and machine tool work elements that make up the part. The example has shown how high level tasks can be broken down into sucessfully smaller and smaller tasks using the process planning data structures.

4. THE AMRF PLANNING SYSTEM

The specification of a data flow model and neutral data interchange formats was a major step in the development of the AMRF interactive process planning system. The model of data flow between planning and control assumes that each controller in the hierarchy retrieves plans that have been placed in the common database by the process planning system (see Figure 3). The primary role of the process planning system, in this model, is to provide interactive tools for generating and storing process plans for new production parts which are later executed by the control systems. Specified inputs to the planning system include: process planning work orders assigned by facility control, initial and final part geometry specifications, definitions of controller work element capabilities, various kinds of reference data, and the plan editing decisions of a skilled process engineer. Outputs from the planning system include: work order status information for facility control, graphics displays for the engineering user, process plans which define manufacturing sequences for each control system involved in production operations, and part model specifications for each new intermediate geometry. This section describes the architecture and operation of the current implementation of the AMRF process planning system.

Lisp, Flavors, and the Lisp Machine

The process planning system was written entirely in Lisp using an object-oriented programming environment. The development plan called for first building an interactive process planning system around the work elements and the interfaces to computer control systems previously described. The more long range goals call for the development of expert planning modules. A decision was made to develop the system using a Lisp machine so that it would be easy to upgrade to more expert planning modules. Lisp, the primary language of the AI community for the development of such systems, has a number of advantages. It is an interactive language so changes can be tested almost immediately, instead of the edit-compile-debug cycle of more conventional languages. Tools for constructing friendly user interfaces, are provided that are mouse and menu- driven. To aid in software development, there are window and menu- based debugger and inspection tools. A language sensitive editor and an object oriented programming environment are also provided. As discussed earlier in the paper, the planning system's most fundamental concept is the work element, which is analogous to the operator concept from artificial intelligence problem solving systems. Each work element has a set of constraints, and when evaluated, makes a specified state change in the system. For a more detailed discussion of the subject see [16]. These work elements, when linked together, form a process plan which describes how to make a part. This representation can be supported in a robust way by the use of an object-oriented environment known as Flavors. In this environment one defines objects, giving them certain behavior. To activate an object a message is sent to that object asking it to invoke a method, (a procedure which changes the object data and/or initiates other messages).

Flavors, Object-Oriented Programming and the Planning System

Earlier we described how a precedence graph is used by the planning system to represent how manufacturing operations are to be performed. Internally, a directed graph is used to represent the precedence relationships. Each node in the graph represents a work-

element, whose precedence relationship is defined between a node's parents and its children. In object-oriented programming, each node can be given a behavior, such as how to act as a node in the network. The current implementation of object nodes includes how to add or delete oneself within a network, display oneself, modify ones attributes and values, etc.

In the requirements lists section of the paper, the automatic generation of the requirements was described. This serves as a good way to illustrate the benefits of object oriented programming. In the methods of a work element is a list of requirements that must be specified to complete this task. When the process engineer asks the system to automatically generate the requirements list, an internal supervisor sends the requirements message to each work- element. They, in turn, respond by sending there requirements to the requirements-list supervisor. This supervisor then sorts the requirements list checking for duplicates, etc. In this way we were able to develop software that closely matches the way people conceptually handle such problems.

The implementation scheme employed also allows for modular system design in the construction of the user interface. Within the planning editors there are three major modules: 1) the internal data representation (a precedence graph), 2) a supervisor, and a 3) user interface manager. The process engineer interacts with the interface manager. The IM is a set of windows that display the current process plan. Items are mouse sensitive (when the mouse is moved around the screen these items are (highlighted). The user points at an item, clicks a mouse button, and a message is sent to the active window highlighting the chosen item. The addition of a new node to the precedence graph is a good example. The interface manager displays a list of all the valid work elements that the user can choose from. One of these items is then selected. A message is then sent to a supervisor describing the transaction to take place (i.e. add a new work element named "Do it" after the fourth step in the process plan). The supervisor then sends the appropriate messages to the internal data structures. Each process plan is an individual entity in the Lisp machine, so the supervisor keeps track of what is the active plan (the one actually being edited) and then it knows where the update message is to be sent. There is no limit on the total number of process plans that can be in the Lisp machine, but currently only one is active for editing. When the user wishes to change to a different plan, the user interface sends a message to the supervisor asking for the plan. The supervisor then gets from the planning system internals the list of applicable work elements on this level and the actual plan.

Through a series of transactions, the window is updated with this new plan, and the work-element that can be added to the plan are then displayed. This scheme was modularized in order that various modules could be distributed to different computers to implement a more distributed planning system.

5. SUMMARY

A process planning system has been implemented to demonstrate the concepts described in this paper. This system is currently being tested and evaluated within the AMRF. This last section of the paper will describe some of the major tools we have implemented, current work in progress, and finally the future direction we will be taking.

Current Implementation

The current implementation of the process planning system has several unique tools for the development of process plans. As described earlier, the work element is the fundamental object within the process planning system. The dynamic nature of the AMRF require an easy to use, flexible tool for creating, editing and viewing the work elements. This is the work element and requirements database. This data base allows AMRF projects to define the work elements and requirements for their workstations and equipment. The tool is graphically oriented, showing the hierarchy of the facility. Figure 7 illustrates the tool. The three levels of the AMRF are shown including the equipment associated with each workstation. The work element choices for the vertical mill are displayed in the menu. The last tool is a work element template editor. This tool is provided for control system implementors to define the work elements their systems can execute.

The output of this system is to generate lisp code to be used by the planning system. The second major set of tools is for the editing of actual process plans. These tools take shape in three areas; a top level for storage and retrieval of process plans for particular parts, a graphical editor for the development of the precedence graph, and a more text oriented tool for editing the individual values of a work elements attributes. The top level tool for keeping track of process plans allows the user to read plans in from files, define new process plans, edit existing plans or browse through all existing process plans. The structure of the system currently uses part names as the highest level. Each part can be opened to see its routing slips, operation sheets, and instruction sets. Each item is indented a certain amount to signify its level within the AMRF hierarchy. This tools resembles a hierarchical file system in a computer. Once the process engineer selects the plan to be edited there are two tools that can be used to develop plans. Figure 1 shows the tool referred to as a graphic-net editor. This tool is used at the more conceptual level of planning to lay out the high level of tasks to be performed. Another tool is used to fill in all of the attributes of a work element. This tool has a different set of windows for each of the major process planning internal data structures, procedure specification, requirements lists, and header section. When the process engineer is finished editing the process plan, it is written to a file using the file specification discussed earlier in the paper.

System Interfaces Under Development

There are currently three major subsystems under development, the interface to a CAD/Solid modeling system, the data administration system (DAS) and the work order management system. All of these interfaces will be used to obtain information about the parts to be produced. The solid modeling interface will serve as a tool to get information about the part geometries to be produced including the dimensional and tolerancing information. It will also be useful for the process engineer to verify the changes to the part geometry, and to do high level interference checking against fixture, and other forbidden volumes. The interface to the DAS will be used to store process plans and to search for process plans within the DAS. It will also be used by the planning system to obtain information about the production capacity of the entire facility (obtain tooling reports, etc.). The interface to the work order management system will provide a tool by the facility level system to give planning jobs to the process planning system and for the planning system to report back on the status of process plans.

Future Work and Conclusions

This first implementation of the planning system has provided quite a learning experience for the staff of the process planning project. The use of Flavors, the lisp machine, development of a file exchange specification, and prototype expert planning systems has given insight into a number of strengths and weaknesses of our current approach. These lessons have pointed out a number of areas that need work in future implementations. A more robust internal representation is needed to allow multiple relationships to exists between work elements in the precedence graph. We have been exploring using commercially available expert system shells such as Knowledge Craft, or KEE. These tools would provide some form of portability across a variety of computer systems. A second major thrust is to design the planning system to be more closely tied in with control systems, so that the planning module gives tasks to a controller while exploring possible alternative paths. A final area of major interest is the development of expert planning modules. Currently we are testing a interface to SIPS [17] for the work of transforming features to process steps. We have been enhancing the knowledge base to reflect the process capabilities of the AMRF, and are using the system as a learning tool for the development of future expert planning modules.

In closing, we have developed a scheme for implementing an intelligent process planning system, and for interface this system to control system within the AMRF. With this information processing architecture, we will be laying the ground work for the next generation of manufacturing facilities.

Acknowledgment

The author acknowledges the efforts of the following people who contributed to the design and implementation of the AMRF process planning system: Steve Ray, Diana Gordon, Brian Drummond, and Mark Unger.

REFERENCES

(1) Chang, T. and Wysk, R., "An Introduction to Automated Process Planning Systems," Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

(2) Simpson, J.A., Hocken, R.J. and Albus, J.S., "The Automated Manufacturing Research Facility of the National Bureau of Standards", Journal of Manufacturing Engineering, 1, Vol. #1, (1982).

(3) Hocken, R. and Nanzetta, P., "Research in Automated Manufacturing at NBS", Manufacturing Engineering, 91, Vol. #4, (1983).

(4) Nanzetta, P., "Update: NBS Research Facility Addresses Problems in Setups for Small Batch Manufacturing", Industrial Engineering, June (1984).

(5) Furlani, C. et al., "The Automated Manufacturing Research Facility of the National Bureau of Standards", Proc. of the Summer Simulation Conference, Vancouver, BC, Canada, July 11-13, 1983.

(6) McLean, C., Mitchell, M. and Barkemeyer, E., "A Computing Architecture for Small Batch Manufacturing", IEEE Spectrum, May 1983.

(7) McLean, C.R., "The Vertical Machining Workstation of the AMRF: Software Integration", Proceedings of the 1986 Winter ASME Conference, Anahiem, California December 1986 (Submitted).

(8) Magrab, E., "The Vertical Machining Workstation of the AMRF: Equipment Integration", Proceedings of the 1986 Winter ASME Conference, Anahiem, California December 1986 (Submitted).

(9) McLean, Charles R., and Wenger, Carl E., "The AMRF Material Handling System Architecture," Proceedings of the Fifth Annual Control Engineering Conference, Rosemont, IL, May 1986.

(10) McLean, C.R., "AMRF System Architecture Document", Draft Technical Report, National Bureau of Standards, January 1986. (11) West, J.D., Levy, F.K., Management Guide to PERT/CPM,Prentice Hall, Englewood Cliffs, New Jersey, 1977.

(11) West, J.D., Levy, F.K., Management Guide to PERT/CPM, Prentice Hall, Englewood Cliffs, New Jersey, 1977.

(12) Brown, P., Gordon, D., "Process Plan Data Exchange Format", Draft Technical Document, July 1986.

(13) Barkemeyer, E., Mitchell, M. (NBS), Mikkilineni, K., Su, S., Lam, H. (Univ of Florida), "An Architecture for Disributed Data Management in Computer Integrated Manufacturing", NBSIR86-3312, January 1986.

(14) Henderson, M.R., "Extraction of Feature Information from Three Dimensional CAD Data", PhD. Thesis, Department of Mechanical Engineering, Purdue University.

(15) Hummel, K., Brooks, S.,"A Structure for the Representation of Manufacturing Features in the XCUT Process Planning System", Proceedings of th 1986 Winter ASME Conference, Anahiem, California, December 1986 (Submitted).

(16) Ray, S. "A Knowledge Representation Scheme for Processes in an Automated Manufacturing Environment", Man and Cybernetics Conference, October 1986 (submitted).

(17) Nau, D.S. and Chang, T.C., "Hierarchical Representation of Problem-Solving Knowledge in a Frame-Based Process Planning System", Journal of Intelligent Systems, Vol. 1, (1986).

(18) Haynes, L., Wavering, A., "Real-Time Control System Software", Proceedings of IEEE International Conference on Robotics and Automation, Vol 3, 1896.

The NBS Automated Manufacturing Research Facility is partially supported by the Navy Manufacturing Technology Program.

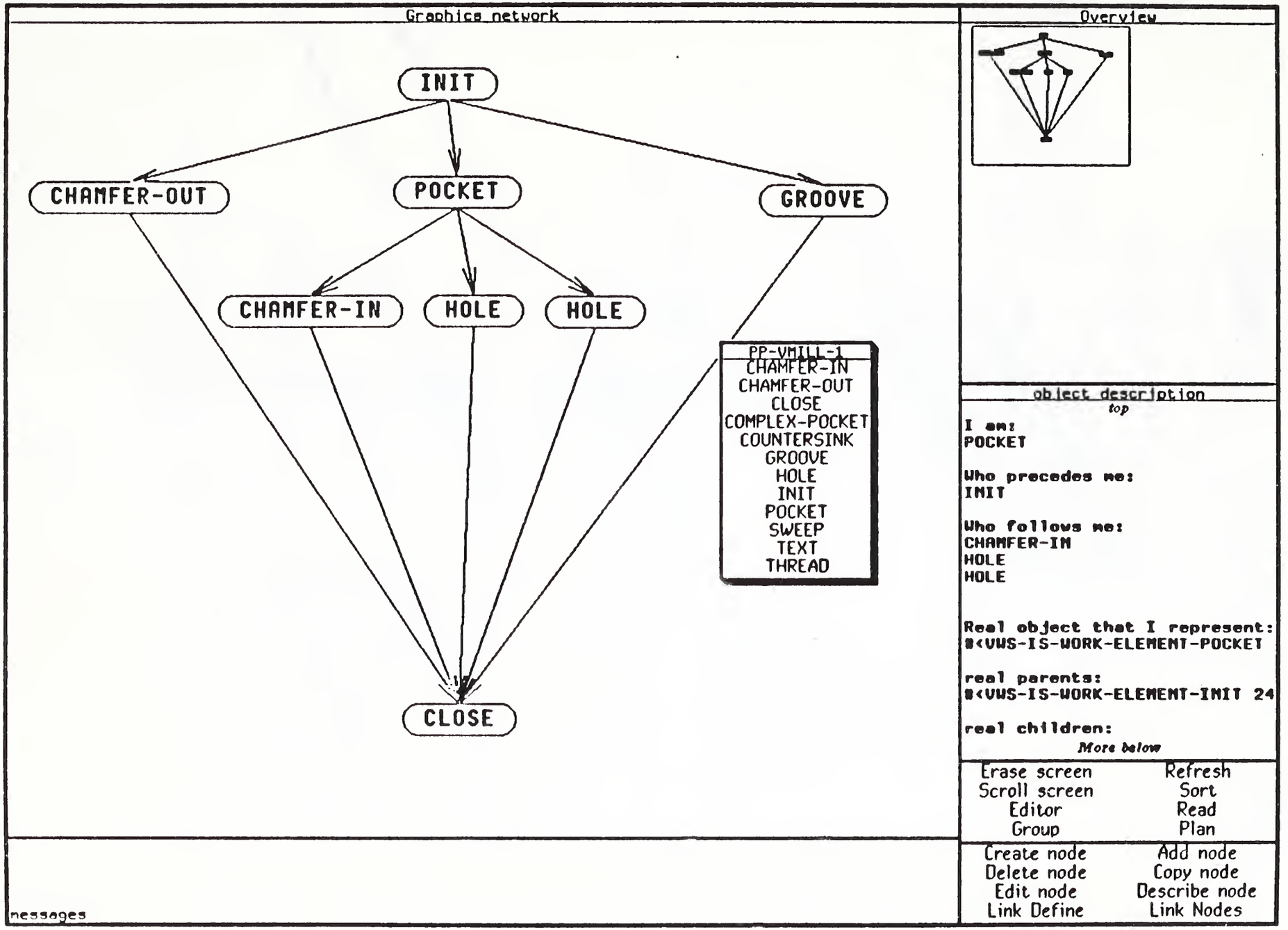
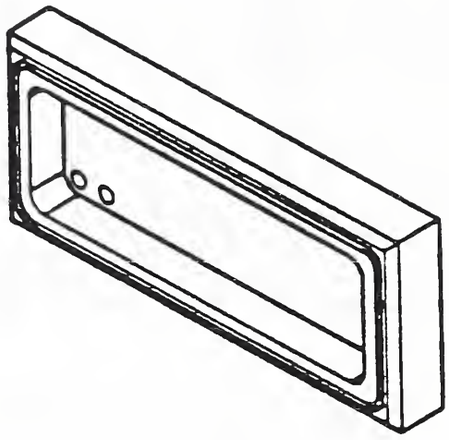


Figure 1. The Graphic Network editor showing an Instruction Set for the Vertical milling machine.

Feature Graph

a)



b)

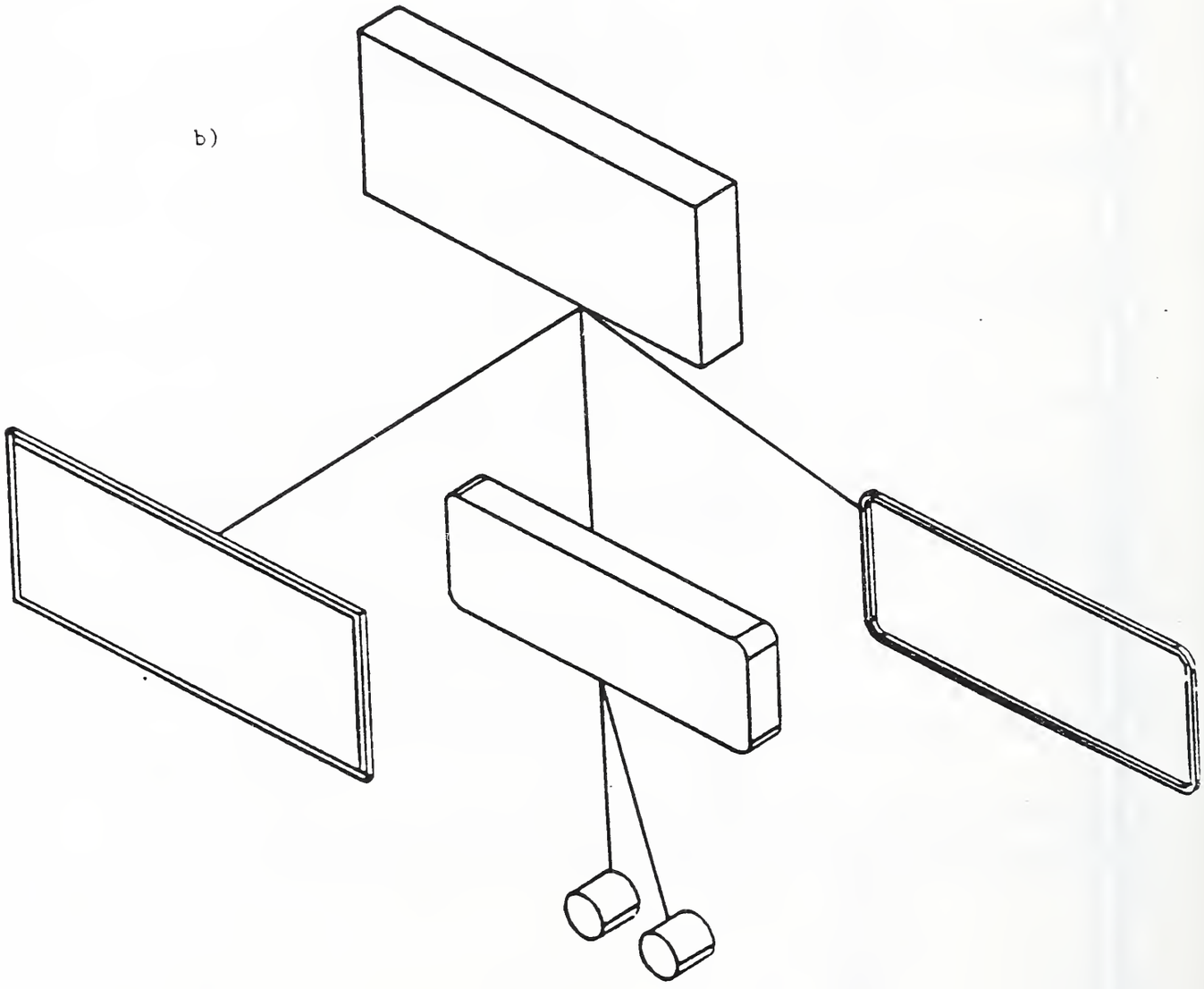


Figure 2. A feature access graph for a simple part, (a) the original part, (b) machinable features and their relationships.

```
--PROCESS_PLAN--
--HEADER_SECTION--

   PLAN-ID            := PP-VMILL-1;
   PLAN-VERSION       := 1;
   PLAN-TYPE          := INSTRUCTION-SET;
   PLAN-NAME          := "FILTER-HOUSING";
   PART-NUMBER        := 31;

--END_HEADER_SECTION--
--PARAMETERS_SECTION--

  $$TOOL-SET001 : TOOL-SET;
  $$TOOL001     : TOOL;

--END_PARAMETERS_SECTION--
--REQUIREMENTS_SECTION--

<<1>>  TOOL-SET
           ( TOOL-SET-ID          => $$TOOL-SET001,
             COMPONENTS           => (2,3,4) );

<<2>>  TOOL
           ( CHANGER-SLOT         => 2,
             TOOL-TYPE            => END-MILL,
             TOOL-ID              => $$TOOL001,
             DIAMETER             => 0.5
             COMPONENT-OF         => (1) );

--END_REQUIREMENTS_SECTION--
--PROCEDURE_SECTION--

<<1>>  INIT
           ( PROG-ID              => NC-1,
             PROG-NAME            => "filter-housing",
             BLOCK-NAME           => BLOCK1,
             SYSTEM               => VMILL,
             TYPE                 => PRIMITIVE,
             PREC-STEPS           => (),
             TIME                 => 0000:00:00:56 );

<<2>>  CHAMFER-OUT
           ( CHANGER-SLOT         => 6,
             FEATURE              => BLOCK,
             Z-SURF               => 0,
             BLOCK-NAME           => BLOCK1,
             SYSTEM               => VMILL,
             TYPE                 => PRIMITIVE,
             PREC-STEPS           => (1),
             TIME                 => 0000:00:01:24 );

<<3>>  HOLE
           ( CHANGER-SLOT         => 7,
             CENTER-X             => 0.725,
             CENTER-Y             => 0.725,
             DEPTH                => 0.25,
             Z-SURF               => -0.625,
             BLOCK-NAME           => BLOCK1,
             SYSTEM               => VMILL,
             TYPE                 => PRIMITIVE,
             PREC-STEPS           => (4),
             TIME                 => 0000:00:00:50 );

--END_PROCEDURE-SECTION--
--END_PROCESS_PLAN--
```

Figure 3. Sample of process plan file format (representative data is shown, some data has been omitted due to space limitations).

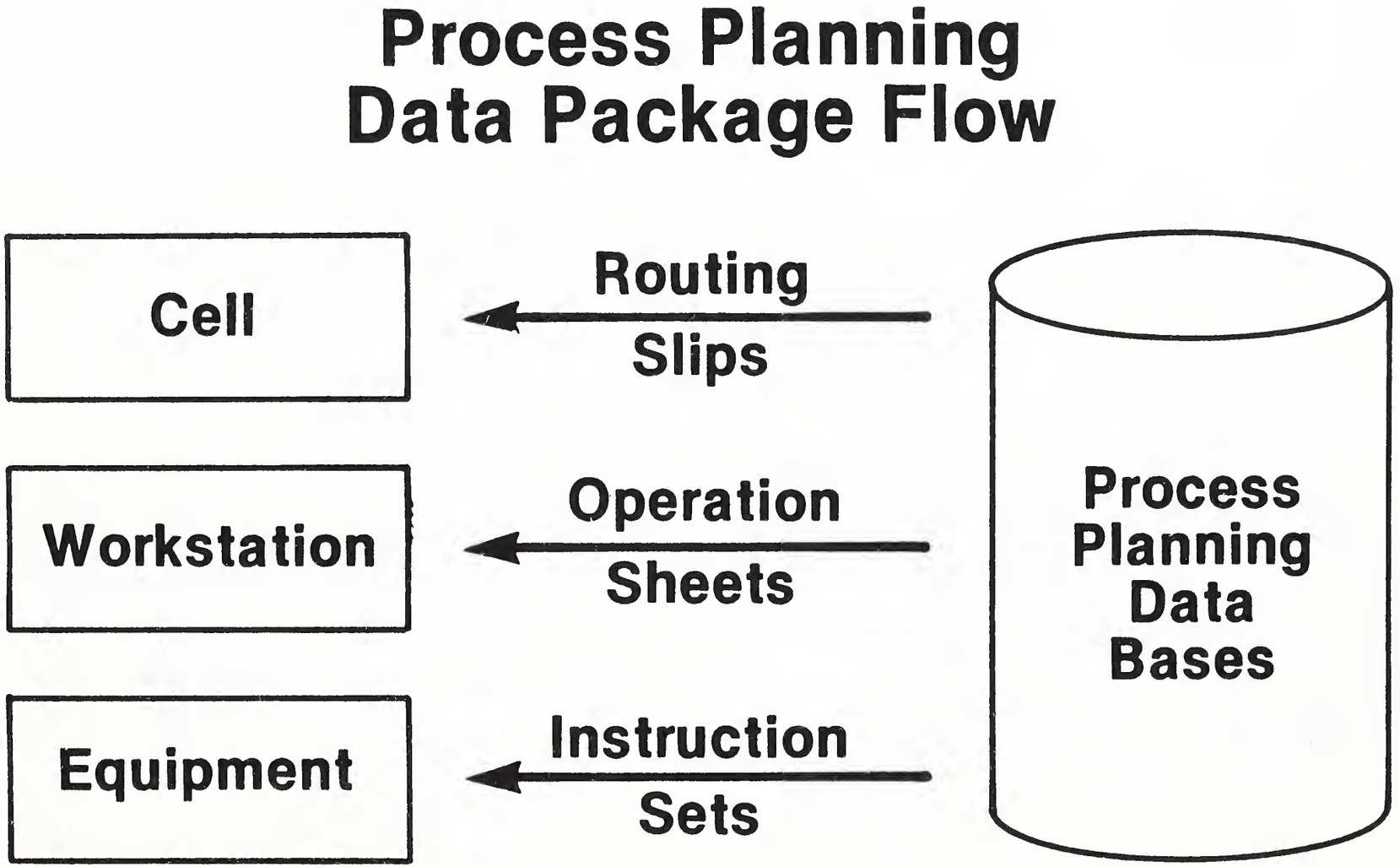


Figure 4. Illustration of data flow from process planning data bases to control systems.

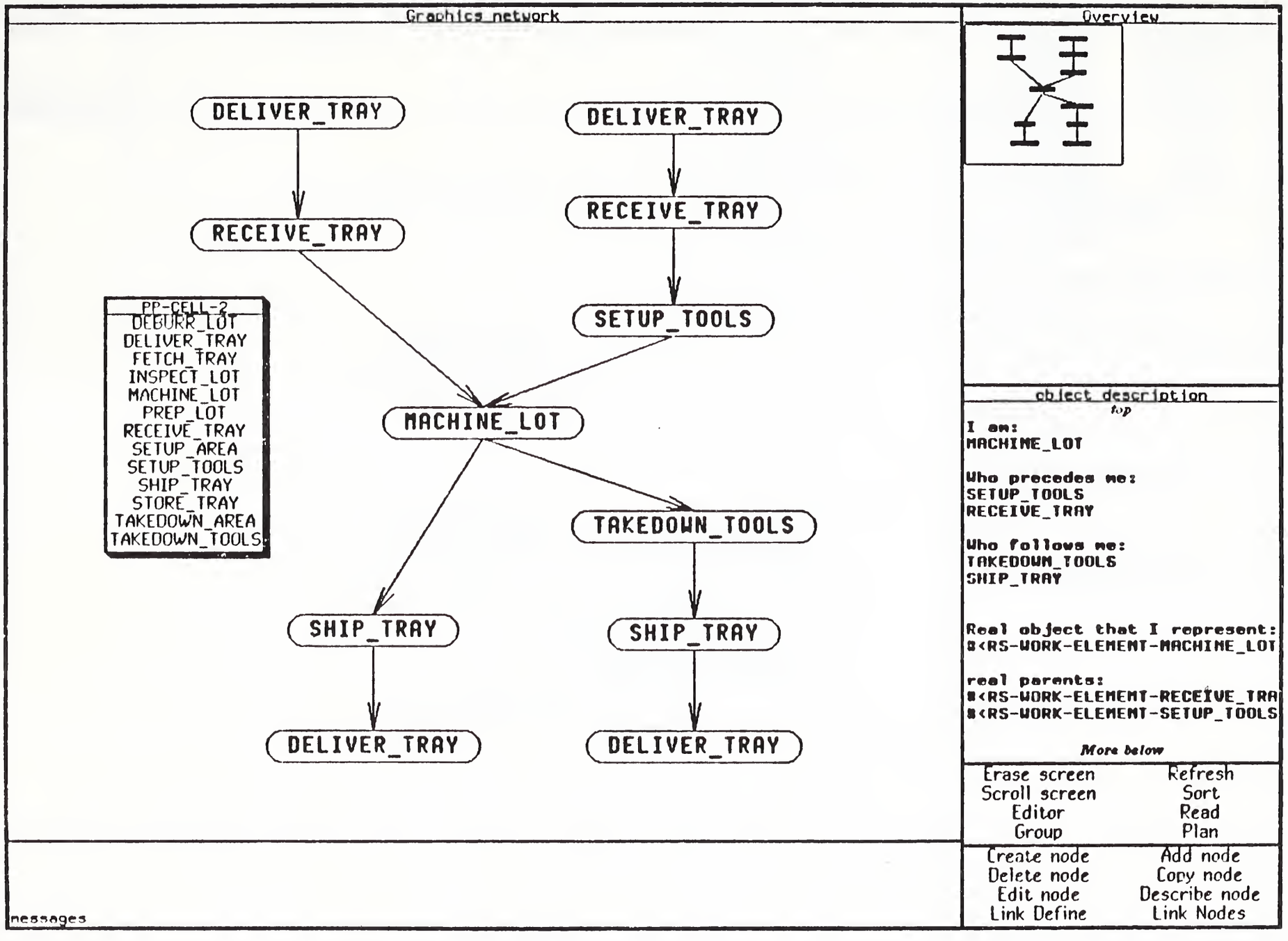


Figure 5. Cell Routing Slip precedence graph displayed on the Interactive process planning system.

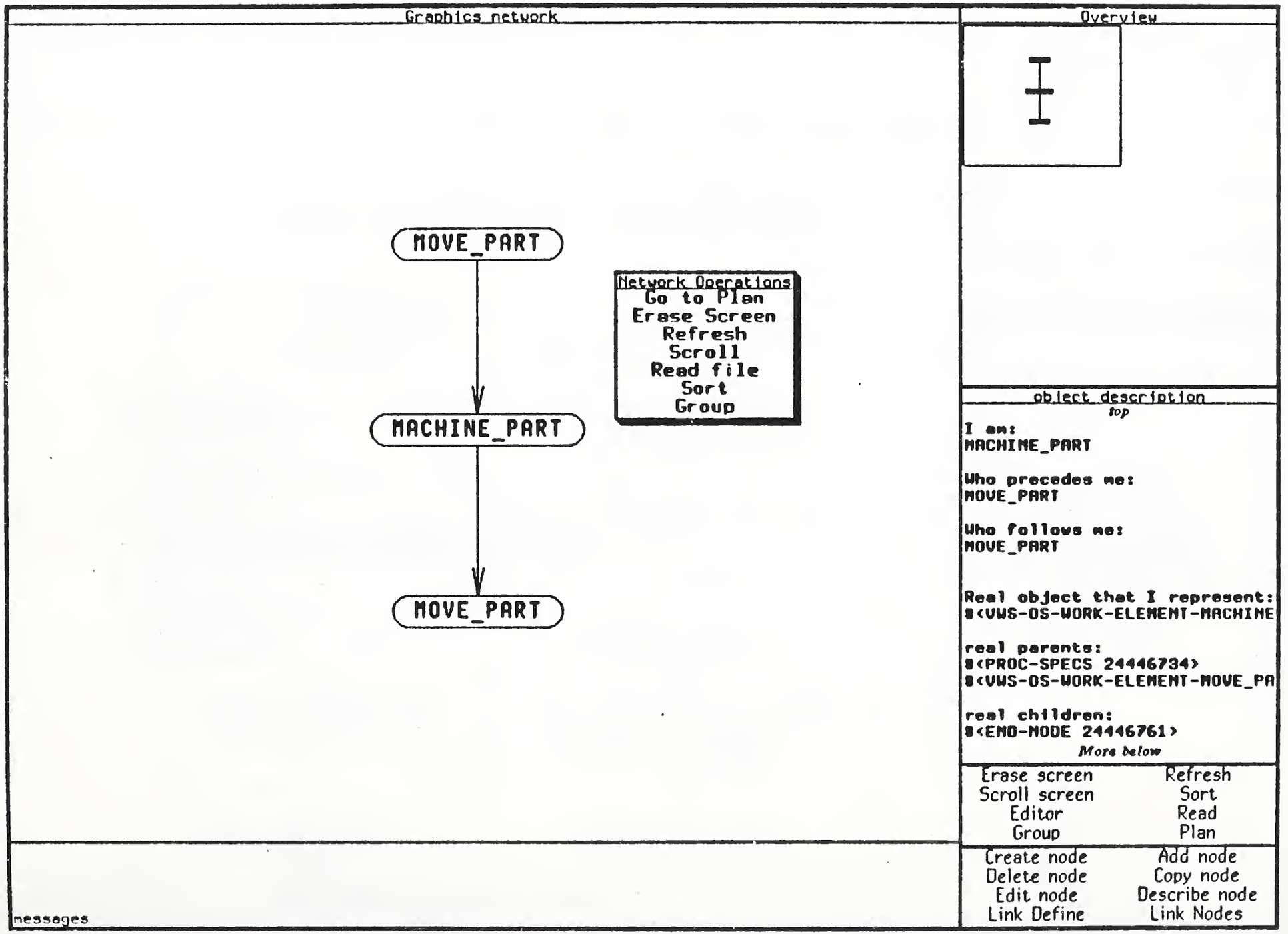


Figure 6. Abbreviated workstation Operation Sheet, further decomposition of machine_part is shown in figure 1.

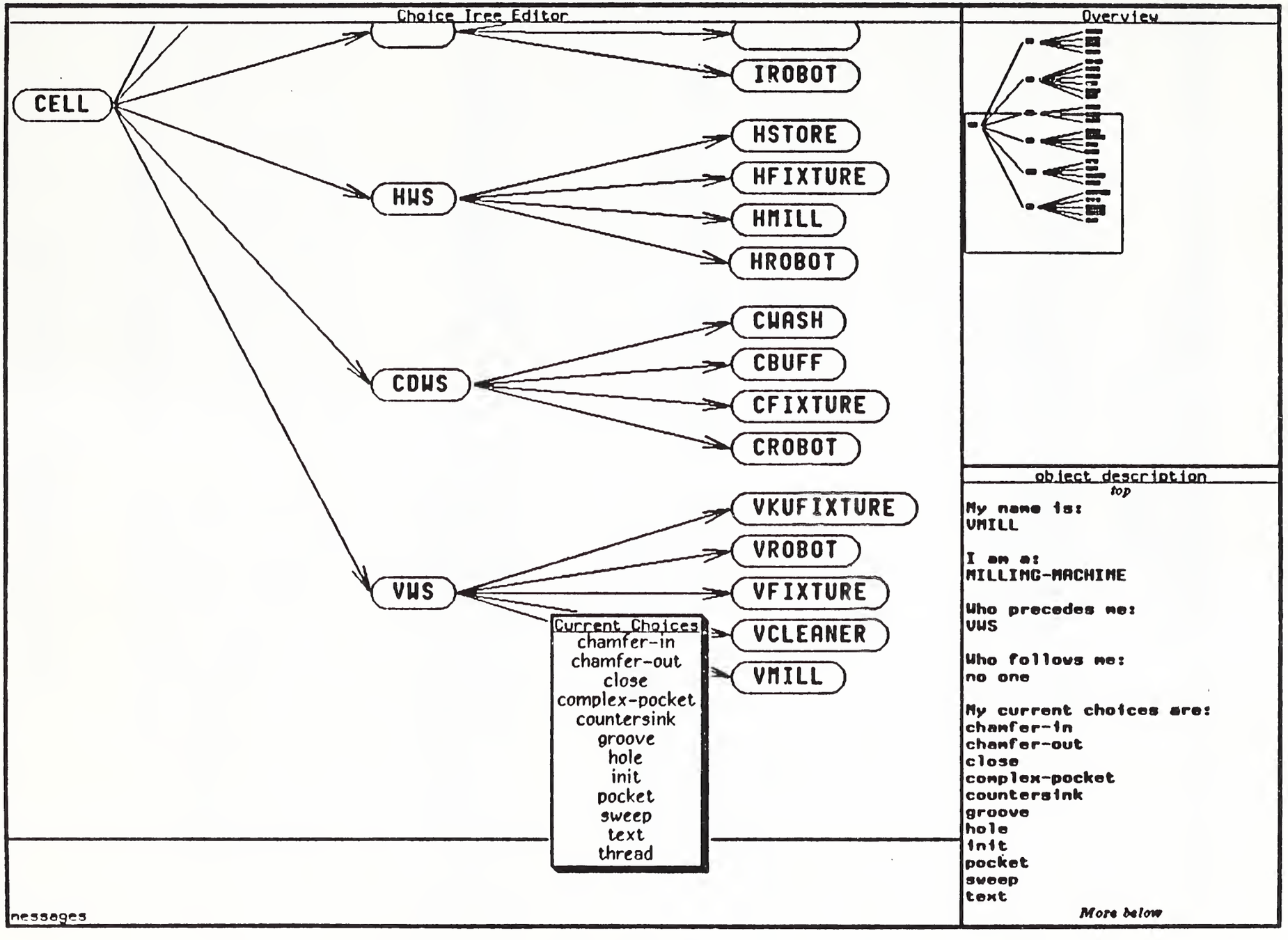


Figure 7. The work element data base, shown are the work elements for the vertical milling machine.

A Knowledge Representation Scheme for Processes in an Automated Manufacturing Environment

Steven R. Ray

Factory Automation Systems Division
Center for Manufacturing Engineering
National Bureau of Standards
Gaithersburg, MD 20899

Presented By:

Steven R. Ray
International Conference on Systems, Manufacturing, and Cybernetics
Pierremont Plaza Hotel
Atlanta, Georgia

Bibliographic Reference:

Ray, S. R., "A Knowledge Representaion Scheme for Processes in an Automated Manufacturing Environment", 1986 IEEE International Conference on Systems, Manufacturing, and Cybernetics", October 14-17, 1986, Atlanta, Georgia.

Reprinted from PROCEEDINGS OF THE IEEE INTERNATIONAL
CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS,
Atlanta, Georgia, October 14-17, 1986

A Knowledge Representation Scheme for Processes in an Automated Manufacturing Environment

Steven R. Ray

Factory Automation Systems Division
Center for Manufacturing Engineering
National Bureau of Standards
Gaithersburg, MD 20899

Abstract

A key factor in applying advanced programming concepts to an industrial manufacturing environment is the establishment of a language to specify the process steps involved. In the Automated Manufacturing Research Facility at NBS, these process steps are described in terms of "work elements." Work elements are specified in process plans which are passed to controllers throughout the facility. This paper describes the properties which were considered in the definition of work elements from the perspective of automated process planning and the control system implementation at NBS.

The control system is based upon a philosophy of hierarchical control, where high level goals are decomposed through a succession of levels, each producing sequences of simpler goals to the next lower level, with the lowest level generating drive signals to robots, grippers and other actuators. To support this scheme, the work elements define the activities that can be carried out at each level of the hierarchy.

The work elements are implemented with different software at each stage in the manufacturing sequence: process planning, communication, and execution. Work elements have been implemented within process planning as programming objects which have slots to describe their function. The information also exists in ASCII "flat file" format, which can be communicated through a common database. Finally, work elements are subroutines which are executed on shop floor controllers. The work elements are instantiated in one form or another as a production job goes from planning to the database, and subsequently executes at a controller. By anticipating the use of the work elements in expert systems and advanced programs, the introduction of intelligence into the manufacturing environment is greatly simplified.

1. Introduction

The Automated Manufacturing Research Facility (AMRF) was designed as a test bed to develop, test and evaluate potential standards in the automated manufacturing industry, (1,2,3,4,5). Since components of manufacturing shops are generally purchased from different vendors, the required compatibility among machines in fully automated shops will depend heavily on interface standards. At the AMRF, a wide variety of "off the shelf" components have been integrated into a single coordinated system, using well-defined communication protocols.

In order to easily introduce advanced programming techniques on the shop floor, automated control standards must also be defined. These control standards should draw upon concepts from artificial intelligence (AI) research, (6). This is particularly important in the area of process planning, which concerns the specification of operations to be performed in order to produce a desired part. Process planning involves much abstract reasoning, and therefore can use many of the approaches adopted in expert systems and other artificial intelligence applications. A standard process representation must incorporate AI concepts to allow these approaches to be easily implemented. An important part of the AMRF representation scheme has been designed with this in mind, and is called a "work element". An interactive process planning system currently uses the work element scheme to generate plans which are executed to produce real parts, (7).

2. Overview

This paper describes the design considerations for a work element, and its use in an operational manufacturing facility. Section 3 introduces some concepts commonly used in artificial intelligence research. Section 4 discusses the definition and characteristics of a work element. Section 5 presents the implementation of a work element for a machine tool in the AMRF. Section 6 extends the work element definition to the hierarchical control system used in the AMRF. Section 7 identifies future directions of research for process representation. Section 8 presents a summary of the paper.

3. Artificial Intelligence Concepts

State Space

A useful concept in artificial intelligence research is the state space, or search space representation, (6,8). The state of the world as known by a computer program is one state in the defined state space. Each change in the program's perception of the world is represented as a transition from one state to another in the state space. The state transitions can thus be considered as links between states, and the state space can be represented as a network.

Operators

Work elements can be thought of as operators in a state space. Whenever a work element is invoked, a state transition takes place. A process plan corresponds to a sequence of operators applied to an initial state, resulting in a goal state. Thus, automated process planning corresponds to a search of state space for the goal state; from the search procedure the state transition operators can be found. The challenge in applying AI to automated manufacturing is to represent the manufacturing task in the framework of AI concepts such as the ones described here, so that the problem can be handled using current AI techniques.

4. Work Element Characteristics

A manufacturing sequence can be conveniently expressed in terms of results-oriented processes, (9,10). These processes can be thought of as transitions in state space. A process plan would thus be a collection of state transitions, from a state containing the part blank, to a state containing the finished part.

A work element is the representation of a state transition, and serves as the language for manufacturing sequences. Process planning consists of selecting and parameterizing the appropriate work elements to traverse the state space from the initial to the goal state, (7). While process planning is widely practiced, it has not been standardized. Robots, machine tools and other controllers all use different control schemes and programming languages. For maximum efficiency, a uniform control structure should exist throughout a facility, using process plans based upon work elements. It is within the work elements that information specific to one controller should be stored.

There are several roles for a work element in the manufacturing environment, corresponding to planning, communication, and execution. In the AMRF, a work element is translated from one form to another to meet the needs of each role.

Planning

In the planning role, a work element exists as part of the procedure specification of a process plan. A process plan represents a set of state transitions and can be represented as a precedence graph. The implementation of a planning work element includes knowledge of any manufacturing steps (work elements) which must precede it. For example, a machining operation must always be preceded by a fixturing operation to hold the part in place. This knowledge can be contained within a work element such as "Drill_hole" which will verify the existence of a fixturing work element each time it is invoked. In addition, a work element must contain all the parameters to completely and unambiguously specify the manufacturing step. It is also useful to maintain a list of all hardware and software requirements needed to accomplish the step. Finally, the work element should have information needed for optimization calculations such as expected duration and cost.

Communication

A work element must also be able to transfer instructions from the planning environment to the execution environment. The overriding requirement for this is compatibility among systems; therefore the work element should exist in as simple and universal a form as possible. Some neutral, machine independent format must be used which can be understood by all controllers in a facility and which contains the essential information from the work element in the process planning role.

Execution

Work elements are executed by calls to subroutines resident within the factory controllers and are invoked with parameters to suit the particular application. Some of these parameters are bound in the planning stage; others by the controllers themselves. In general, the subroutines include verification and error handling routines. A more detailed discussion of the execution of work elements is outside the scope of this paper; the reader is referred to (11,12).

5. Work Element Implementation

Planning

In the AMRF, the planning work elements have been implemented using a frame-based or object-oriented approach. The major benefits of object-oriented programming are:

1) Local variables (slots) can be assigned to each object. In the case of process planning, each object represents a work element.

2) Distinct behavior can be defined for each object (methods).

3) The behavior and slots can be inherited from classes of objects.

The slots are used to store information pertaining to a particular frame in the form of attribute-value pairs. Table 1 shows some of the slots used in the process planning system for the work elements. The slot "Autogen-nodes" is used to identify any work elements which must precede the one being defined. Thus, the planning system can automatically insert planning steps which may be missing by referring to this slot value. These automatically generated steps will be work elements as well, with the "Gen" slot set to "T", indicating that it was inserted automatically. "Autogen-rqmts" meets a related need by specifying any pieces of equipment or computer code which will be needed to execute the work element being defined. The specified requirements can also be automatically added to a running list of requirements which accompanies each procedure specification. Both of these slot values help to make the job of process planning more convenient for the process engineer and to avoid careless oversights. The slots "Parents" and "Children" are used to construct a precedence graph for the process plan. These are simply the pointers to the previous and subsequent steps in the plan, respectively. "Complete" is used to signal when a particular work element has been fully defined. The process plan is not complete until all the component work elements have this flag set to "T".

Attribute	Value	Comment
System	-	Set to the cell, workstation or equipment meant to execute the step
Time	-	Set to the estimated execution time for this step
Autogen-nodes	-	A list of prerequisite work elements
Autogen-rqmts	-	A list of hardware and software requirements necessary for successful execution of this work element
Parents	-	Pointers to previous step(s) in the plan
Children	-	Pointers to subsequent step(s) in the plan
Complete	T or F	Flag denoting work element is completely specified
Gen	T or F	Flag denoting whether work element was automatically generated
Type	Complex Primitive Macro	Defines the type of work element
Plan_id	-	References the plan used to expand this step at a lower level

Table 1. Minimal set of attributes in a frame-based work element

User-defined slots	
SYSTEM	TYPE
PLAN_ID	CHANGER_SLOT
CENTER_X	CENTER_Y
DEPTH	Z_SURF
BLOCK_NAME	
Type-checking slots	
SYSTEM-DATATYPE	TYPE-DATATYPE
PLAN_ID-DATATYPE	CHANGER_SLOT-DATATYPE
CENTER_X-DATATYPE	CENTER_Y-DATATYPE
DEPTH-DATATYPE	Z_SURF-DATATYPE
BLOCK_NAME-DATATYPE	
Valid choice slots	
SYSTEM-CHOICES	TYPE-CHOICES
PLAN_ID-CHOICES	CHANGER_SLOT-CHOICES
CENTER_X-CHOICES	CENTER_Y-CHOICES
DEPTH-CHOICES	Z_SURF-CHOICES
BLOCK_NAME-CHOICES	
Others	
ATTRIBUTES	AUTOGEN-NODES
AUTOGEN-HEADER	AUTOGEN-RQMTS
TIME	COMPLETE
GEN	COMMENT
PRINT-NAME	PRINTPARENTS
CHILDREN	VALUE
VISITED	PARENTS

Table 2. Slots in the frame-based version of "Drill_hole"

Table 2 shows the slots in an example planning work element which is used to drill a hole at a machine tool. In addition to the slots identified in Table 1, there are attributes necessary for the execution of the hole-drilling process, including hole depth, diameter (using "Changer_slot"), and location. In constructing the internal implementation of this and other work elements, the planning system automatically included some supplemental slot definitions. For example, "Changer_slot-datatype" and "Changer_slot-choices" were also defined. The "Changer_slot-datatype" slot provides for type-checking, which is the simplest form of verification of user input. The "Changer_slot-choices" slot, if set to something other than "nil", provides the process engineer with a set of valid choices when specifying the "Changer_slot" slot value. The choices are derived from contextual information, thus presenting the process engineer with whatever choices are reasonable at that particular time. This again reduces the possibility of operator error when using the planning system.

The second major performance enhancement when using object oriented programming techniques is that of message passing. Each programming object is defined as having slots, described above, plus specific functionality, defined using "methods". A method is a function definition for a particular programming object. It is these methods which give the objects their different behavioral characteristics. Thus, an object can be instructed how to insert itself into a process plan, or how to find out all the previous steps in a plan. By defining a set of methods, process planning becomes a matter of sending messages to the appropriate work *element* objects to insert themselves into a plan *and* to *communicate* with the other objects in the plan. Rather than trying to explicitly keep track of all the steps required, the objects update themselves about the planning hierarchy and their relationships. This makes the maintenance of a valid plan much easier, since the objects can have methods defined to check for prerequisite work elements, hardware and software requirements, as well as contextual information determining their behavior. By delegating the responsibility of maintaining a plan to the work elements themselves (plus some other programming objects) it is no longer necessary to have a single master program which anticipates all possible errors.

Perhaps the most significant improvement offered by frame based systems is the concept of inheritance. A given frame can be defined as inheriting all the attributes and behavior of one or more "parent" frames. This can be easily applied to the implementation of work elements by first defining the taxonomy of process steps. The simplest way is to design a process tree, with the root being "all process steps". Beneath this could be classifications such as "hole processes", "surface processes", etc. At yet lower levels, one could have sub-categories such as "hole creation processes" and "hole improvement processes". Finally, the leaves of the process tree would be the individual work elements, such as "twist_drill_hole", "center_drill_hole" or "ream_hole". By constructing such a process taxonomy, one can take advantage of the concepts of inheritance. The class of "hole creation processes" inherits all the slots and methods of "hole processes", and adds to them, slots and methods specific to hole creation operations. The class of "drilling processes" inherits the slots and methods of "hole creation processes", plus whatever is important to drilling operations. In this way, a new process can be defined, inserted into the process taxonomy, and it immediately acquires a set of relevant behavioral characteristics. Further, by classifying work elements in this way, automatic process selection becomes easier to implement, by traversing the tree, making a decision at each branch. An implementation of this concept is currently being integrated into the AMRF process planning system, based upon a previously developed tool called SIPS (Semi Intelligent Process Selection), (13).

Communication

In keeping with the need for simplicity, the work element in its communication role does not take advantage of any advanced programming concepts. As shown in the example of Table 3, it is in human readable, ASCII text form, called a "flat-file" or neutral data exchange format, (14). It consists of nothing more than a work element name and a collection of attribute-value pairs, which correspond to some of the slots of the planning work element. This stripped-down version of work element appears in all communications of process plans between the process planning system, the AMRF databases, and the various controllers, (14).

```
<< 1 >> DRILL_HOLE
                    ( SYSTEM => VWS ,
                      TYPE => PRIMITIVE ,
                      PLAN_ID => PP-VWS-72 ,
                      CHANGER_SLOT => 6 ,
                      CENTER_X => 4.50 ,
                      CENTER_Y => 2.25 ,
                      DEPTH => 0.5 ,
                      Z_SURF => 0.0 ,
                      BLOCK_NAME => BLOCK1 ,
                      PREC_STEPS => ( ) ,
                      TIME => 0000:00:01:00 ) ;
```

Table 3. Communication version of work element "Drill_hole"

6. Hierarchical Systems

To support the hierarchical control scheme being used in the AMRF, work elements have been defined for each level. Process planning is also carried out in a hierarchical fashion, representing the first step toward truly distributed, automated process planning. This is a distinct change of approach from process planning methods traditionally used, (15). The lowest level in the hierarchy, called the Equipment level consists of industrial devices, such as robots, automatic carts, and milling machines. The second level is called the Workstation level and consists of a physical grouping of equipment level devices. For example, a milling machine, a lathe and the robot used to service them might make up a workstation. Each workstation has a controller which is implemented on a microcomputer. The third level in the hierarchy is called the Cell level. Essentially, a cell is the collection of workstations needed to accomplish some production job. Activities within the cell are coordinated by a cell controller. Levels still to be added include a shop level which will coordinate and optimize the activities of the cells, and a facility level which would be used to supervise several shops, such as an assembly shop and a manufacturing shop. In addition, the facility level coordinates various "front office" support functions. Current plans for the AMRF are for a manufacturing shop, using metal removal for part production.

Task decomposition is fundamental to hierarchical control. A high level goal is decomposed into sets of simpler goals for the next lower level. To maintain flexibility in a programmable factory, the method of decomposition should be defined by the data and not built into the facility itself. To perform new production jobs, i.e. producing new parts, one needs only to provide new process plans, without any further programming, assuming the existing work element definitions are sufficient.

To provide the capability for flexible task decomposition, each work element has an attribute called "Type" which determines how it is handled by a controller, (see Table 1). "Type" can have one of three values:

1) Primitive

This is the simplest case, where a work element corresponds directly to an executable subroutine, as in the "drill_hole" example.

2) Complex

This instructs the controller to decompose the given command into simpler commands to be executed at the next lower level in the control hierarchy. It does this by retrieving the process plan identified by the attribute "Plan_id", which contains the decomposition of the given command. For a detailed discussion of the decomposition and execution process, see (12).

3) Macro

Here the controller expands the given command into a set of commands to be executed at the same level in the hierarchy. Again, this is done by retrieving another process plan from the distributed database.

As an example of how task decomposition would work, suppose the Cell level of control received a "Process_Batch" command, (see Figure 1). This is a complex work element, with a pointer to a process plan defining its decomposition. The Cell controller would retrieve the referenced plan, which would contain a set of work elements to be executed by workstation level controllers. Examples would include commands for a milling workstation to receive a lot of parts, receive some tools, and machine the lot of parts. "Machine_lot" is also a complex work element, and upon receiving the Machine_lot command, the milling workstation would retrieve the referenced plan containing work elements to be executed by equipment level controllers. These would include work elements such as "drill_hole" which was discussed earlier, which is a primitive work element.

One of the main advantages of this form of hierarchical control is that it allows a truly modular, distributed implementation. The parallelism which results greatly increases the execution speed of a complex control system. This approach also supports distributing the process planning function. When a command is received, a controller could either retrieve a previously generated plan stored in a database, or it could call a local planning function, using the current control state data to generate a new plan to be executed at that particular level in the hierarchy. Ultimately, every controller in the hierarchy could have a resident planning module which would perform local task planning in real time, as commands are received. Finally, a hierarchical control implementation allows error handling to be treated hierarchically also. An error detected at some low

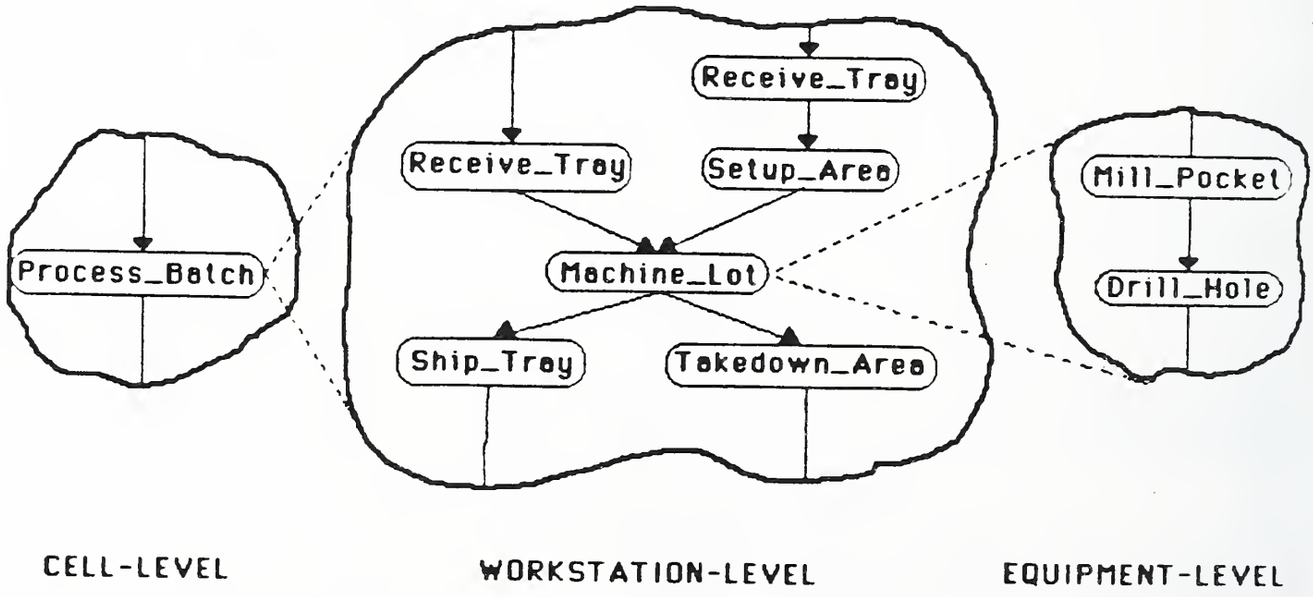


Figure 1. Example of command decomposition using hierarchical process plans

level need not be reported to the highest level of control if a local recovery technique is available. This avoids the situation of an entire factory shutting down whenever the slightest problem occurs anywhere in the control system. Rather, the error is reported only to the first level of control able to handle the situation, allowing other processes to continue uninterrupted.

7. Future Implementations of Work Elements

It seems inevitable that future automated process planning systems will rely heavily on ideas from artificial intelligence research. With the exception of the SIPS system, the current implementation of process planning in the AMRF uses traditional Lisp and object-oriented programming techniques. Features such as "autogen nodes" and "autogen rqmts" which automatically insert process steps and requirements into process plans are the first steps toward the development of intelligent planning modules which would function automatically. The next version of work elements may be implemented as intelligent modules with their own sets of rules to determine how they will be decomposed into simpler commands, (16). Thus, when a work element is considered for inclusion in a process plan, it would be told to plan its own parameterization, using the current planning context. This scenario leads to the possibility of truly distributed, real-time process planning, rather than offline development and storage of plans for later execution. The separation between planning and execution will become blurred as the modules acquire more local intelligence and real-time interactive capability.

8. Conclusions

A scheme for representing process steps in an automated manufacturing environment has been described. Called a work element, this representation plays distinct roles in process planning, plan communication and control system execution. By defining all task decomposition in terms of work elements, programming new production and support operations becomes simply a matter of supplying new process plans, or adding new work elements to the library. The structure of the work elements allows intelligent problem solving techniques to be easily implemented, such as "smart" work elements: self-contained rule-based systems which dynamically change their behavior depending on context. By designing the fundamental knowledge representation scheme in such a generic and flexible way, and by adopting standards based upon such designs, it should be easier to maintain compatibility in the future with existing systems of today.

9. References

(1) Simpson, J.A., Hocken, R.J. and Albus, J.S., "The Automated Manufacturing Research Facility of the National Bureau of Standards", *Journal of Manufacturing Engineering*, 1, #1, 18, (1982).

(2) Hocken, R. and Nanzetta, P., "Research in Automated Manufacturing at NBS", *Manufacturing Engineering*, 91, #4, 68, (1983).

(3) Nanzetta, P., "Update: NBS Research Facility Addresses Problems in Setups for Small Batch Manufacturing", *Industrial Engineering*, 68, June (1984).

(4) Furlani, C. et al., "The Automated Manufacturing Research Facility of the National Bureau of Standards", *Proc of the Summer Simulation Conference*, Vancouver, BC, Canada, July 11-13, 1983.

(5) McLean, C., Mitchell, M. and Barkemeyer, E., "A Computing Architecture for Small Batch Manufacturing", *IEEE Spectrum*, 59, May 1983.

(6) McLean, C.R., "An Architecture for Intelligent Manufacturing Control", *Proc. of Summer 1985 ASME Conference*, Boston, Massachusetts, August 1985.

(7) Brown, P.F. and McLean, C.R., "Interactive Process Planning in the AMRF", submitted for ASME Symposium, December 1986.

(8) Winston, P., *Artificial Intelligence*, Addison-Wesley, Reading, Massachussetts, 1984.

(9) Jones, A.T. and McLean, C.R., "A Production Control Module for the AMRF", *Proc. of Summer 1985 ASME Conference*, Boston, Massachusetts, August 1985.

(10) Hummel, K., "An Expert Machine Tool Planner", presented at the International Computers in Engineering Conference and Exhibition, Boston, Massachusetts, August, 1985.

(11) Kramer, T. and Jun, J., "Software for an Automated Machining Workstation", *Proc. of Third Biennial Intl. Machine Tool Technical Conf.*, Chicago, Illinois, September 1986.

(12) McLean, C. and Wenger, C.E., "The AMRF Material Handling System Architecture", *Proc. of Fifth Annual Control Engineering Conference*, Rosemont, Illinois, May 1986.

(13) Nau, D.S. and Chang, T.C., "Hierarchical Representation of Problem-Solving Knowledge in a Frame-Based Process Planning System", *Journal of Intelligent Systems*, Vol. 1, (1986).

(14) McLean, C.R., AMRF System Architecture Document, Draft Technical Report, National Bureau of Standards, (1986).

(15) Chang, T. and Wysk, R., *An Introduction to Automated Process Planning Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

(16) Drummond, B.S., "An Intelligent Notification System for Complex Physical Processes", Masters Thesis, Computer Science Department, George Washington University, (1986).

The NBS Automated Manufacturing Research Facility is partially supported by the Navy Manufacturing Technology Program.

Research Issues In Process Planning at The National Bureau of Standards

Peter F. Brown
Steven R. Ray

Factory Automation Systems Division
Center for Manufacturing Engineering
National Bureau of Standards
Gaithersburg, MD 20899

Presented By:

Peter F. Brown
The 19th CIRP International Seminar on Manufacturing Systems
Pennsylvania State University, University Park, Pennsylvania
May 31-June 1, 1987

Bibliographic Reference:

Brown, P. F., Ray, S. R., "Research Issues in Process Planning at the National Bureau of Standards", Proceedings of the 19th CIRP International Seminar on Manufacturing Systems, University Park, Pennsylvania, May/June, 1987.

Research Issues in Process Planning at the National Bureau of Standards

by

Peter F. Brown
Steven R. Ray

Factory Automation Systems Division
Center for Manufacturing Engineering
National Bureau of Standards

ABSTRACT. Several years ago, the Automated Manufacturing Research Facility (AMRF) project was established at the Gaithersburg site of the National Bureau of Standards (NBS). This facility is unique in several ways: first, all manufacturing activities are under direct computer control; second, all manufacturing data preparation systems and control systems are linked through a complex data administration and communication system; third, all manufacturing operations are carried out by robots and machine tools with a minimum of human intervention. This last constraint requires that all manufacturing data be complete and unambiguous. It was necessary to develop a process planning system which was capable of supporting the particular requirements and manufacturing capabilities of the AMRF. This paper describes the research agenda of NBS and its cooperative efforts over the past few years in the area of Automated Process Planning. Results include: the development of a neutral representation for process plans and a part model; the development of an interactive planning system which supports all controllers in the AMRF hierarchy; the use of expert systems for process and tool selection; automatic speed and feed calculation; and development of a system for automatic part fixturing. The next phase of development involves the introduction of distributed intelligent planning modules. By following a systematic procedure of defining clear interface specifications and establishing a framework for modular software development, progress is being made on the complex problem of process planning in an automated manufacturing environment.

INTRODUCTION. With the rising importance of national industrial competitiveness, the need for technological improvements in the manufacturing arena is becoming acute. It is clear that the source of many of these improvements will be the field of automation. Manufacturing automation can speed product turnaround, reduce the need for retooling, and lead to a more efficient allocation of resources. Automation can be effective for small batch manufacturing and in spare parts production. While this is a desirable goal, many small shops cannot afford to fully automate. By using clearly defined interfaces, a shop can support both manual and automated operations. Pursuing a research agenda for fully automating a factory should yield useful results for manual, semi-automated and fully automated facilities.

There are a number of obstacles to the implementation of a fully integrated automated manufacturing facility. One major gap is the lack of smooth information flow between Computer Aided Design (CAD) systems and Computer Aided Manufacturing (CAM) systems. Traditionally, these two functions have been treated as completely separate activities. There is no feedback from CAM to CAD to reflect the manufacturability of a particular design. There are very few commercial/production systems which actually integrate CAD with CAM. An example of one which does accomplish this for a limited part family is the General Dynamics Advanced Manufacturing System [McMahon87]. After the design of a wing-spar, the design is checked for manufacturability. Potential problem areas are identified to the designer who can then make the appropriate modifications to improve the manufacturing process. The implementation of this system required major modifications to their existing CAD and CAM facilities, and a significant outlay of human and financial resources.

Extending these ideas to general part families is a much more difficult task. A brief example of the information flow illustrates a number of problem areas. A client requests some product to be manufactured and provides a loose set of requirements. This request is translated into a local representation, usually a part drawing. This local representation includes some simple translation of functional attributes into specific tolerance information. The information is loosely organized as notes or text on the part drawing. This step can be done manually or with the current technology of Computer Aided Drafting. The step is complete when the client and designer agree that the drawing adequately represents the client's needs. The problem with this approach is that the information is not represented in a computer database form. This implies that a human will have to interpret the notes sometime later in the process, leading to ambiguities. More importantly, this approach does not allow any feedback to the designer as to the manufacturability of the design.

The next step in the process is to bridge the link to the CAM systems. This is called process planning. Process planning transforms the design information into some local process specification structure used by the manufacturing organization. This step includes defining a group of machinable features and their associated processing steps, selecting target machine tools to be

used to process the part, generating tool and fixturing orders, and any other information needed to actually produce the part. The CAM system then expands each process step into more detailed instructions including robot or machine tool N/C programs, tool offsets, etc. It is at this point that important information is generated which should be communicated back to the designer. The important point is to produce a product at minimum cost while retaining the desired quality and functionality.

Thus, the step called process planning is the transformation of information from the CAD representation to the CAM representation. The transformation rules that humans apply are not well understood even by those who use them. Clearly this makes it difficult to encode those rules in process planning systems. It is only when these rules can be represented in automatic systems that any feedback can be given during the design process. To accomplish this, a more powerful product representation is needed. This representation must serve the needs of the designer who is striving for functionality, as well as the manufacturing engineer who wants high quality at low cost.

Key research issues are the development of a complete product definition that captures the design and functional aspects of the part, the understanding and development of the transformation rules discussed above, and finally the development of models of the constraining mechanisms that affect those transformation rules. The key standards issue is the development of a standard process plan representation. A standard representation permits the independent development of planning modules and reduces the integration problem. The process planning project has addressed a number of these issues internally and in collaboration with other organizations. Process planning is one part of the larger AMRF project whose goal is to study the problem of information flow in an automated facility, and to develop and test system interfaces for this information flow.

OVERVIEW. This paper addresses the key research efforts and issues supporting the integration of automated process planning in the Automated Manufacturing Research Facility (AMRF) at the National Bureau of Standards. Section 3 describes the AMRF facility in terms of its goals, architecture and implementation. Section 4 discusses the role of process planning within the AMRF, and identifies some of the underlying issues which must be addressed before integrating a planning system. Section 5 details the research activities supporting process planning conducted at, or in collaboration with, NBS. Section 6 outlines a strategy for future work, and Section 7 summarizes the paper.

THE AMRF. The AMRF was established in 1981 to serve as a testbed facility to support research in measurement techniques and computer interface standards that are required for automated machining of parts in small lot sizes. One of the primary thrusts of the project was to establish clear interface specifications and modular structures to allow plug-compatibility between systems. This allows both a flexible manufacturing environment and offers the capability of incremental automation in existing facilities. Results of this work are already contributing to the formulation of standards for a generic factory model, low level robot interfaces, process plan file structures, N/C machine tool interfaces, communication standards, IGES (Initial Graphical Exchange Specification) and PDES (Product Definition Exchange Specification). Currently, a PDES-like format is used to communicate the part geometry and functionality. As the formal definition of PDES is developed, we intend to maintain compatibility.

(1) The Role of NBS. The National Bureau of Standards plays a unique role in manufacturing automation. It serves as a common ground where both academic and industrial research issues can be explored. Industrial research efforts often suffer from the constraints imposed upon them by a plant in full production. The cost of taking down a production line to experiment with new automation concepts is prohibitive. This results in a conservative approach to implementing new technologies in a plant. Universities, while free to take great risks with new ideas, rarely have the resources to carry out large scale experiments involving many industrial robots and controllers. This is primarily due to the large investment in capital equipment that is required. Furthermore, it is difficult to remain aware of the problems currently facing production facilities without either working at such a facility, or working with personnel from the facility. The AMRF addresses many of these problems. Experiments can be carried out on a realistic scale without the loss of production. The AMRF provides a forum where industrial and academic researchers can work and discuss their various perspectives. Finally, by keeping information in the public domain, results of work performed at NBS can be made available to the entire manufacturing community.

(2) AMRF Architecure. The AMRF is built around the concept of hierarchical control, where high level commands are decomposed into sequences of simpler commands at the next lower level in the hierarchy, which in turn are decomposed at yet lower levels (Figure 1). Well-defined protocols have been established to allow command and status information to flow up and down the hierarchy. The bulk of data transfer (such as process plans and part models) occurs laterally with a distributed data administration system. A mechanism has been implemented to allow any controller in the AMRF to request or store information in a generic way, regardless of which database is being used to hold that information. The adoption of such an architecture avoids many potential information bottlenecks. Further, by adopting a hierarchical approach, the complexity of a task is reduced to a manageable level for any node in the hierarchy. More details on the AMRF can be found in [Simpson82, Furlani83, Hocken83, McLean83, McLean85, Nanzetta84].

PROCESS PLANNING IN THE AMRF. The process planning system in the AMRF was designed to accomplish many goals. One major goal of the planning effort was to establish a neutral format for a process plan at any level in the control hierarchy. This format had to be simple enough to be easily parsed by the least capable computers in the facility, yet flexible enough to convey complex process plans containing multiple branches. A second goal of the planning system was to serve as a general programming tool for the facility. Since all workstation controllers in the

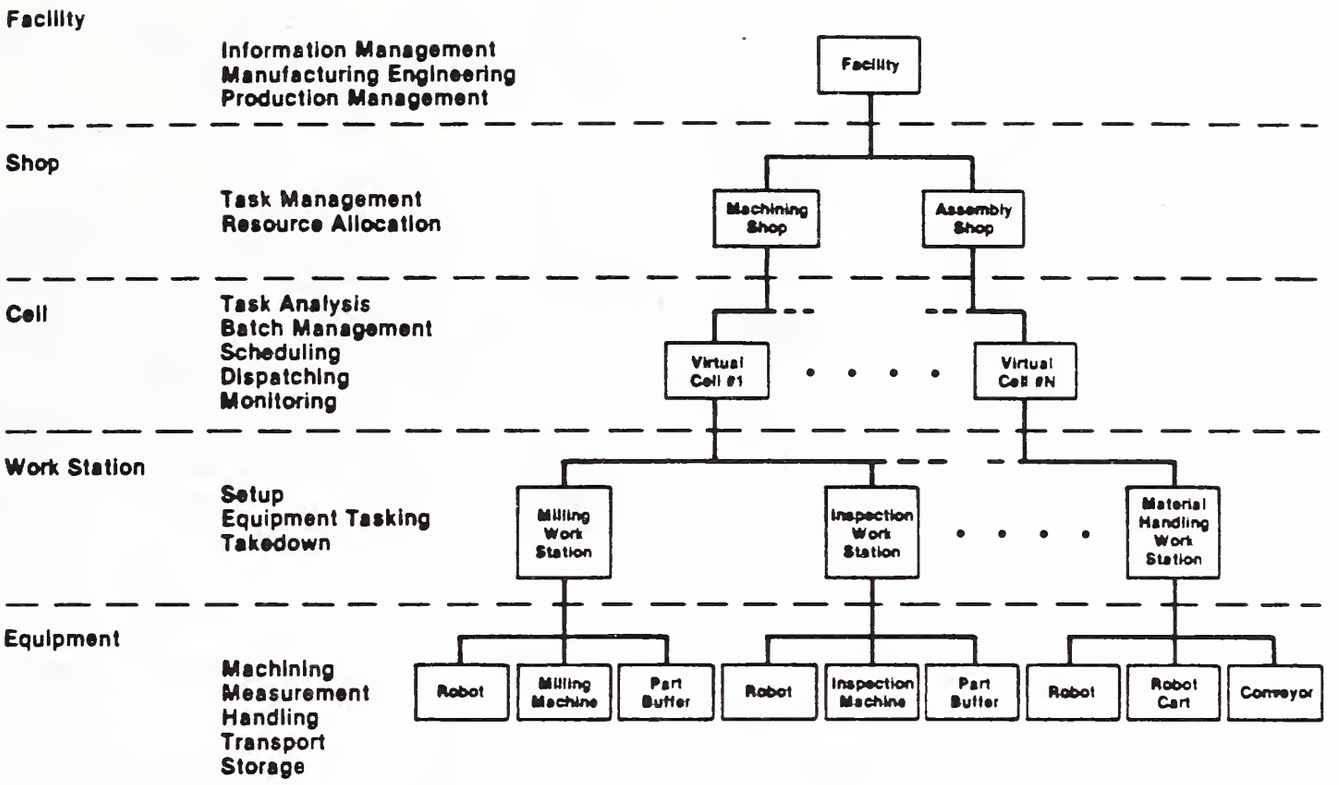


Figure 1. The AMRF COntrol Hierarchy.

facility are designed to interpret and execute process plans in the same format, the process planning system can generate command sequences for activities involving any combination of devices on the factory floor. The planning system supports all three levels of the hierarchy currently implemented: the cell, workstation, and equipment level (Figure 2).

Before these goals could be tackled in a systematic way, a number of issues had to be addressed, for example: What representation scheme should be used for a process plan, both within the planning system computer, and at execution time on the factory floor? How should an individual step within a process plan be represented? How should the hardware and software requirements for a process plan be stored? How is system integration and interface specification to be accomplished? How should the system handle command, status and database transactions, which are common to all systems in the facility? The research program in process planning was formulated with the above questions in mind. The approach used to address these issues, detailed in the following section, was to work on many of the immediate problems within NBS, while supporting and working in collaboration with others on some of the more long term questions. In-house work therefore focussed on representation and interface issues, with outside projects addressing expert system approaches, geometric feature manipulation, automatic fixturing, and other topics.

RESEARCH TOPICS SUPPORTING PROCESS PLANNING IN THE AMRF. A technology evaluation was carried out early in the project to determine the current state of the art of both production and research process planning systems. The goal was to determine if the technology used in these systems could be used in a facility such as the AMRF, i.e. one with direct computer control of all factory operations. It was found that variant planning systems suffered from severe drawbacks in generality and extendability, and no system addressed all the necessary issues. It was further decided that a number of central items had to be developed which simply did not yet exist. These included:

- A standard representation of process plans based on programming language theory from computer science.

- A standard representation of activities on the shop floor. A representation was derived based on knowledge representation techniques from artificial intelligence.

- A product representation (rather than just a part drawing) as output from a design system. This representation is used to drive the planning system.

- A methodology to allow the generation of alternate functional views of the product data as needed by various factory systems.

- A methodology relating these features to the automatic generation of machine specific code.

This section describes the research performed at NBS and elsewhere in collaboration with the AMRF, dealing with issues such as those outlined above. The interactive planning framework built to support the AMRF is also reported.

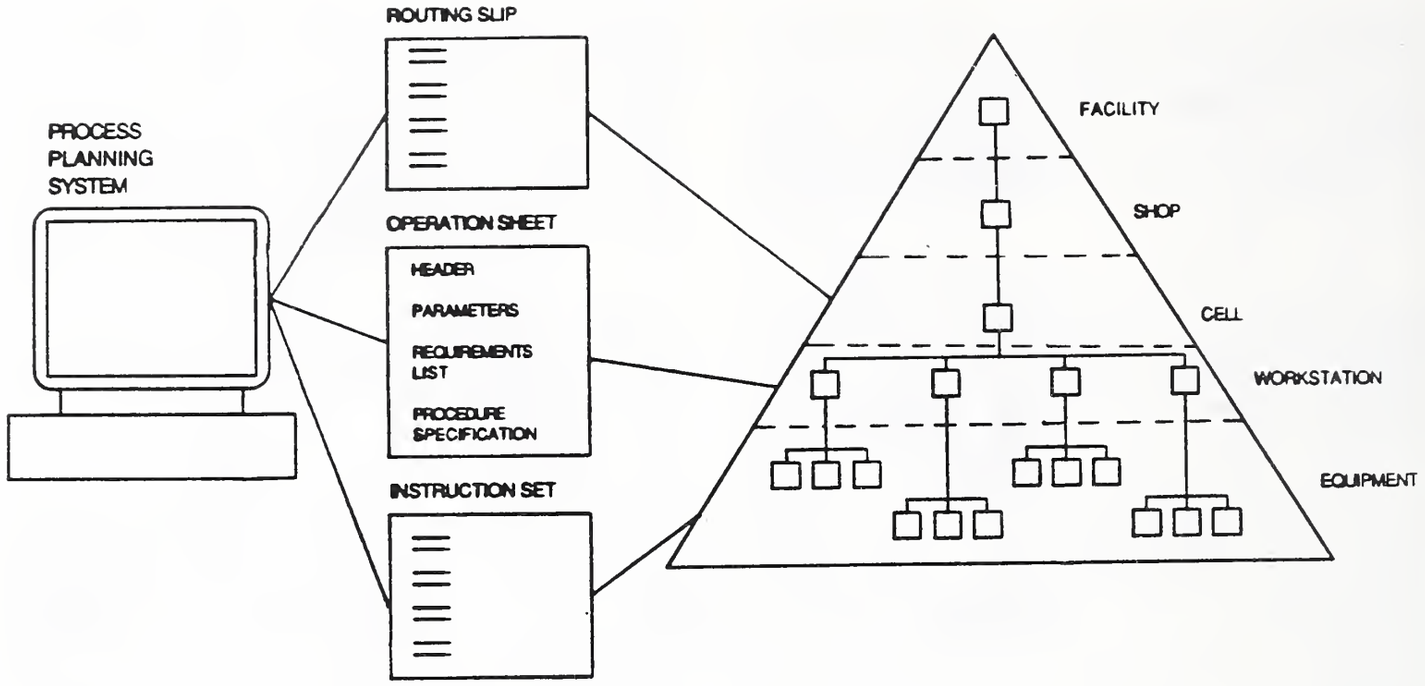


Figure 2. Process planning data packets and corresponding control levels.

(1) Assessment of Computer-Aided Process Planning. Two key collaborators working with NBS on the early phase of research into computer aided process planning were Dr. Ted Chang and Dr. Dana Nau. An NBS grant to Dr. Richard Wysk at Virginia Polytechnic Institute entitled "Advances in Computer-Aided Process Planning", [Chang83] provided a useful survey of existing planning systems and current concepts. The outcome of this work served as the basis of the book "An Introduction to Automated Process Planning Systems" [Chang85]. At the same time, Dr. Nau was at NBS as a guest researcher who became interested in the applicability of artificial intelligence to process planning. The result of his work was "Expert Computer Systems and Their Applicability to Automated Manufacturing" [Nau82]. Many of our current concepts on process planning came out of this early collaboration.

(2) A Machine Tool Planner for Automated Process Planning. A core task in the transformation of design data into a process plan is the task of process selection, followed by machine code generation. Typically, this means starting with the specification of a design and determining the processing step or steps needed to produce it. In collaboration with the University of Kansas, a graduate research project began at NBS [Hummel85] to investigate possible means of performing such a task automatically. One of the outcomes of the investigation was the decomposition of the task into three parts. The three parts or phases are called: feature planning, operation planning and machine planning. During each of these phases "constraint posting" is used, constraint posting consists of the formulation, propagation and satisfaction of constraints which describe the interactions between various sub-problems. The constraints can, for example, include causal relationships between machining operations, or restrictions on resources. The first step (feature planning) takes a list of manufacturing features as input. If no processing knowledge exists for a given feature it is decomposed into a list of simpler features, by means of pointers embedded in the feature definition. This could lead to the generation of precedence constraints based on the sub-features produced. The next step, operation planning, involves the selection of machining operations to produce each of the "elemental" features identified in the previous phase. The machining operation specifies various parameters, such as feed rate and cutter speed. Finally, the machine planning step turns these operations into groups of APT-like program segments.

The Kansas implementation uses a production rule approach, modeled after conventions of YAPS [Allen83], to represent the rules needed in each of the planning sub-tasks. The system is written in Franz Lisp (tm) on a Sun Microsystems workstation, specifically for a Bridgeport CNC vertical milling machine. It has successfully produced plans for a limited set of pocket and hole making operations. Mr. Hummel has continued this work at the Bendix Corporation. Concepts such as meta-rules to control the search, and an optimum search tree generator have been implemented. A simple geometric reasoning capability was also added to aid in the feature decomposition problem. Much was learned about the representation of machinable features and the need for better geometric reasoning capabilities and constraint propagation methods.

(3) Automated Process and Tool Selection. Several years ago, an independent effort was initiated at the University of Maryland by Dr. Dana Nau to investigate novel approaches to the application of artificial intelligence to process planning. This work was funded in part by NBS. Dr. Nau developed a prototype reasoning system in Prolog called SIPP (Semi Intelligent Process Planner). This was soon followed by a version implemented in Franz Lisp , then re-coded in Zetalisp on a

Symbolics Lisp machine. Dr. Nau realized that a core task in the planning problem was that of selecting a process, given an isolated manufacturing feature. The latest version focused on this problem, and was named SIPS (Semi Intelligent Process Selector). SIPS is a frame-based reasoning system which was designed around the concept of "hierarchical knowledge clustering", [Nau87].

There are several advantages to the SIPS approach as compared to traditional production rule systems. First, conditions which are common to several processes can be evaluated in a parent node. Thus, only the conditions which distinguish one process from another "sibling" process need be evaluated by any of the child nodes. The second major difference is the concept of the cost of a process. Ideally, one would like a process selector to generate a plan with the lowest cost. In production rule systems, priorities can be assigned to rules which rank them by cost, but generally the priorities must be assigned beforehand. In SIPS, the order of the search is determined by the cost estimate for each process, which is calculated during the reasoning process. Thus, in situations where the cost is feature dependent, SIPS offers a convenient way to rank the candidate processes. Finally, SIPS provides a representation of both procedural and declarative knowledge in a conceptual frame.

The SIPS system is currently integrated into the interactive process planning framework of the AMRF. It can be invoked when editing process plans at the equipment level of the hierarchy. In operation, the process engineer specifies the part to be machined in terms of design or manufacturing features meaningful to SIPS, ordered in a feature graph. Each feature can then be passed to SIPS, which will replace that feature in the graph with the process, or sequence of processes recommended to produce it. It is then the task of the engineer to consolidate the collection of processes needed for all the features into an optimized sequence of operations. The optimization of this last step is currently being investigated. Enhancements to SIPS are currently being supported, through cooperative research efforts between NBS, Dr. Nau and researchers from Texas Instruments. These efforts involve the enhancement of: 1) the overall problem solving paradigm, 2) the inferencing strategies used, 3) the knowledge representations employed, and 4) the domain specific knowledge bases.

(4) Automated Fixturing - University of Kansas. The Department of Mechanical Engineering at Kansas University has been working with NBS under a grant for several years on computer integrated manufacturing. One research issue has been in the area of automated part fixturing, [Carlyle86]. This process is almost always performed by a machinist because of the complex nature of the problem. Researchers at Kansas believed a properly designed modular fixturing system could be assembled by a robot. By constraining the range of solutions using modular fixtures, progress could be made in developing an automated approach to part fixturing.

Work proceeded along three main branches: to develop fixturing hardware to be controlled by computer, a fixture planner, and a robot planner. The fixturing hardware was designed to be a baseplate type of assembly, with a matrix of conical holes. Each hole accepts an endstop or a clamp. Further, the clamp can then be driven hydraulically under computer control to open or close. To support the hardware, a fixture planner was also developed, called "Baseplatetool", [Unger86]. This system graphically displays the baseplate on a computer screen, and allows a process engineer to specify the arrangement of stops and clamps needed for a fixturing operation. The system uses a two dimensional modeler for the purposes of speed, unlike an earlier version which used a solid modeler. An important feature of the system is the use of a separate database to store all facility-dependent information. This includes the layout of the baseplate itself, the clamp designs, the parts to be fixtured, the locators used, the size of the locator holes, etc. In this way, Baseplatetool can be quickly adapted for use with any hole-based fixturing system. The interface uses mouse input. Great efforts were made to allow the engineer to remain at the conceptual level when designing a fixture. The third development was a robot planner to allow robotic assembly of fixtures. This system takes the fixture design generated using Baseplatetool, and produces a process plan to be used by a robot in the assembly of the fixture components.

To integrate the work on automated fixturing with the ongoing research at NBS, a postprocessor was written for the robot planner. This produces a process plan in the neutral AMRF format for the robotic assembly of a fixture designed with the tool. The fact that the fixturing hardware and software was fully integrated with the AMRF within a week of its arrival at NBS serves as a testament to the power of machine independent interfaces.

(5) AMRF Process Planning System. The process planning system consists of two primary sections: a configuration tool and editing tools, (Figure 3). The configuration tool is used to specify the organization of the equipment on the factory floor. Thus, it allows a user of the planning system to construct a representation of the facility. This representation contains the cells, the machining and support workstations, and all of the associated processing equipment.

An internal database is used to keep track of the activities or functions that each factory floor system can perform. The database maintains the specification of an activity, its associated constraints and other information. These activities are called work elements, [Ray86]. The work element concept is derived from the idea of an operator in state space. Thus, the application of a work element results in a transition within a control system from one state to another. From the perspective of the planning system, every control module in the factory is treated the same way, whether it controls equipment (such as a machine tool controller) or directs other control modules (such as the cell or workstation controllers).

The second tool is the one used to actually create, edit, or view process plans. The plans created with this tool are in terms of the entities and work elements defined in the configuration tool. There is a network interface to external databases where process plans can be stored, and other information such as part models and inventory data can be accessed. Once the user has selected a process plan for editing, the information can be displayed in two alternate forms. One

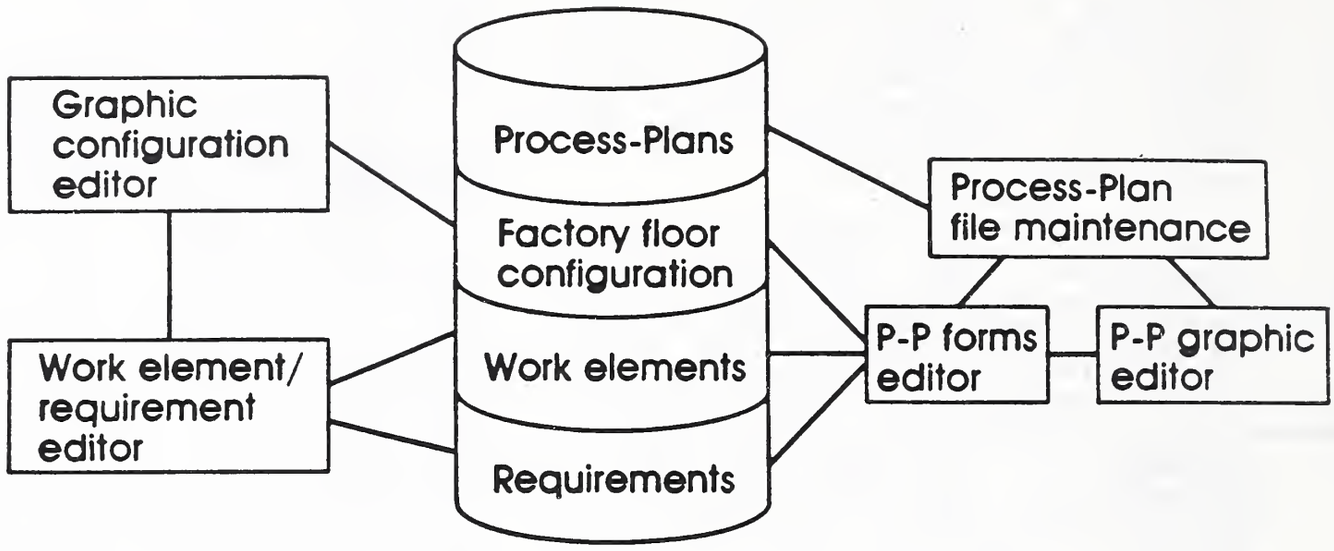


Figure 3. The AMRF Process Planning System.

display uses a text or form layout, while the second uses a graphical representation based on the precedence information within the plan. Both tools show the same information, but the graphical tool provides easier viewing of the overall plan while the textual display gives the user more detailed information.

A major effort supporting the integration of the planning system within the AMRF was the development of a neutral process plan format. This format is an ASCII based language specification that is used throughout the AMRF. A process plan is comprised of four major sections:

1) Descriptive Header - contains static index and summary data.
2) Parameters - lists all variables for which real values must be substituted at execution time.
3) Requirements List - identifies all resources to be used during the execution of the plan.
4) Procedure Specification - describes all work elements, their precedence relationships, their attributes and specific value bindings.

Further details of the interactive process planning system can be found in [Brown86].

Another critical interface developed within the AMRF is a part model or product specification format. This part model consists of the part geometry and topology (based on a boundary representation) and part functionality, [Hopp87, Tu87]. The functionality section allows the specification of datums, datum reference frames and tolerance information. In addition to this information, a mechanism has been developed for the specification of features. These features can refer to any information within the part model, including other features. This format provides a mechanism which allows multiple uses of the part model (such as design, process planning, vision, and inspection). An application system use the same underlying part specification, but develops different views of this information.

In summary, the current planning system supports the neutral process plan format and the part model format. Process plan procedures are described in terms of work elements. The system also has the capability to invoke an external expert module to perform automated process selection. The neutral process plans are readable by all controllers within the AMRF. Some of the equipment controllers then execute predefined N/C programs. The vertical machining workstation can dynamically generate N/C code from a process plan and feature description, [Kramer86].

STRATEGY FOR FUTURE WORK. The major goal during the first several years of the AMRF was the design, construction and integration of the present facility. That goal has been reached and the system was demonstrated during the public test run in December of 1986. The next phase of research is to conduct experiments using the current facility. One important research area is the development of distributed planning and control systems.

(1) Perspective of Current Work. The current implementation of the process planning system supports the architecture of the AMRF. This system is interactive, i.e. it requires human decision making throughout the development of a process plan. The system was designed to allow modular extensions for intelligent problem solving. The SIPS system has been integrated and other expert modules can be added in a straightforward manner. This is possible because of the fundamental work already done in designing the interfaces to the AMRF.

One of the key outcomes of the work done to date has been the rethinking of the role of process planning in an automated factory. Also, the importance of clear, well defined interfaces cannot be over-emphasized. The development of standard interfaces has been of great help in speeding the software

development. A great deal of work still needs to be done to define interactions between control systems and planning systems and refine the features used in the product specification.

With a framework in place which supports process planning in a fully automated environment, work can now proceed on the integration of artificial intelligence technology into the system. By proceeding in this way, we hope to keep our efforts focused on those areas most needing attention.

(2) Role of expert systems and artificial intelligence. It is clear that expert systems have a vital role to play in the manufacturing environment. Many portions of the manufacturing decision making process are based on heuristic rather than algorithmic knowledge. Some key areas are ripe for consideration for future expert systems, such as resource allocation, machine selection, tool selection, etc. Tying all of these systems together into a series of cooperative expert systems still remains one of the most important challenges. At the same time, however, the need to better integrate conventional programming tools with the current system has become apparent. Many relatively straightforward tasks still need to be performed, such as data base interfaces and speed/feed calculations. Tasks which do lend themselves to expert system solutions may still be best accomplished with computer-assisted tools which interact with a human engineer. The computer-assisted tools will probably have the largest immediate impact in the manufacturing arena.

(3) Distributed, Real-Time Planning. A distributed architecture offers the greatest chance of success for the implementation of a flexible planning system which can react in real time to unforseen situations. The AMRF hierarchical control architecture is a convenient testbed in which to develop these planning concepts. The hierarchical approach means that a complex problem can be broken down into a number of solvable sub-problems [Sacerdoti77]. By distributing the problem among a number of processors, more computational resources can be applied to the problem in parallel. Further, the modular construction allows the system to be easily modified to reflect changing factory configurations. Figure 4 shows the allocation of planning responsibility among a level hierarchy. Figure 5 represents a hypothetical scenario for information flow between two levels within the hierarchy. Each node has both a planning and control module. What follows is one example of how a distributed planning and control system could function.

- The control level Z passes down a command for a job to be performed.

- The A planner might already have a stored template describing the appropriate course of action, or it could develop a set of tasks necessary to execute the command.

- Planner A asks the subordinate level planners about the feasibility of sub-tasks X,Y,Z...

- Planner B responds with a "YES" and returns a process plan that includes an estimate of the time, cost and resources required.

- Planner C supports a similar piece of equipment and also returns "YES" with a lower cost, but a much longer time estimate.

- This information is then used by the planner or controller at level A to decide which plan would be best to use, and to combine and optimize the various sub-tasks.

At a given time, module C could be a better choice, but some time later, if delivery time became critical, module B would be a better choice. Further, if module C should break down during execution, the planner could simply recommend module B as an alternative. It is important that a planning module first produce a rough estimate as to whether it can handle a job, and then during execution help provide error recovery. This second step could be performed by continually generating contingency plans, whenever the module is otherwise idle.

There are of course numerous ways that a cooperative planning and control architecture could be designed, this represents just one approach. It is our belief that the architecture of the AMRF, and the interfaces that have been defined will allow the implementation and testing of these ideas in a convenient and robust fashion.

Planning Levels	Planning Functions
Level 1	GT-Cell Classification
Level 2	Machinable Feature Classification
	Plan Optimization
Level 3	Process Selection
	Tool Selection

Figure 4. The decomposition of planning functions within a hierarchical planning system.

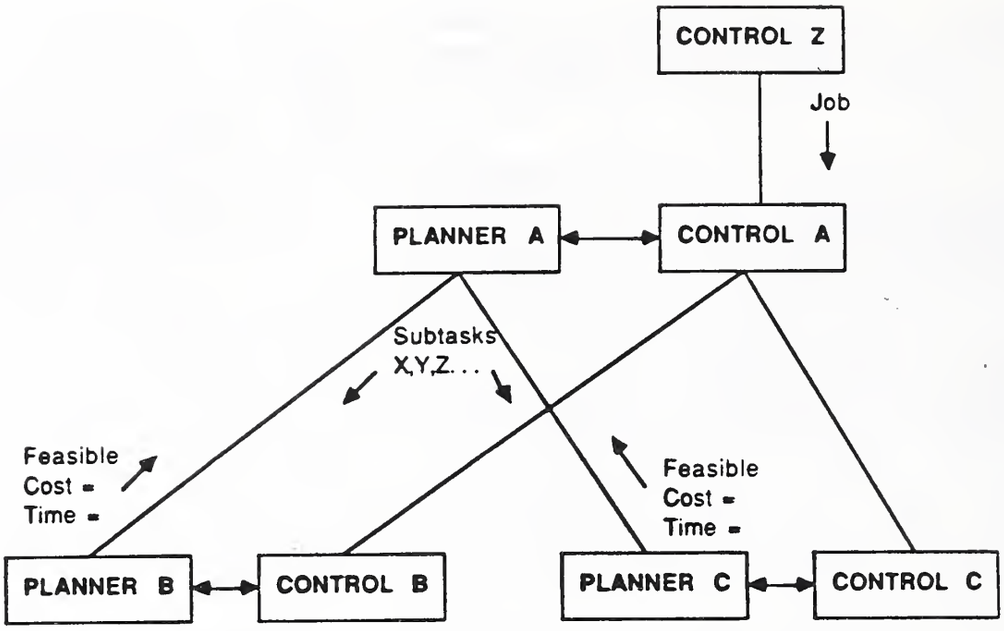


Figure 5. Flow of planning information within a distributed hierarchical planning system.

(4) Portability. The current process planning system was written in Zetalisp running on a Symbolics computer system. We still feel Lisp is the best environment for this type of software because it is widely available, it supports object oriented programming, windowing facilities, flexible data typing and an interactive programming environment. All of these features greatly enhance the productivity and flexibility of a software developer. But issues have emerged concerning the differing needs of software development environments and application delivery systems. Since we started the process planning system, general interest in artificial intelligence environments has greatly increased. The Lisp environment on conventional computers has improved significantly. Personal computers have now become serious Lisp programming tools.

We are beginning to define the environment for the distributed planning system. We are looking into a Lisp environment which contains portable, public-domain software. This software should include object-oriented and windowing facilities. Our goal is to be able to implement a system which will run on a variety of host machines.

(5) Design by Features.
In traditional design, the functionality of a part is never explicitly stated. The designer transforms the functionality into geometry and tolerance specifications. Subsequently, there is no good way to provide feedback to the designer on issues such as cost, manufacturability and performance. An important development which should radically change this situation is the concept of design by features. Since both designers and process engineers conceptualize in terms of features, a feature representation is a natural vehicle for part description, [Dixon86, Hummel86]. We believe that a relationship can be established between design and manufacturing features. Once this relationship is known, a mechanism can be developed to provide the feedback to the designer. Default parameters can also be attached to these features, making the design and manufacturing tasks more consistent. In this way, the risk of over or under-constraining a design is reduced. Finally, these features can be related directly to geometry to aid in the analysis of the functionality of a part, such as strength, heat transfer characteristics, etc. This approach underscores the fact that manufacturing concerns are as important as functionality in order to produce economical, high quality products. All aspects of a part, including design, analysis, manufacturing and inspection should be weighed against one another.

CONCLUSIONS. The Automated Manufacturing Research Facility at the National Bureau of Standards is pursuing a systematic approach to the development of process planning systems for future automated factories. Early work focused on representation issues. Results include a neutral process plan format, a part model format, and the concept of a work element. Building on this framework, an interactive planning system was designed and implemented. The system provides planning service for all AMRF control systems. As work progressed, we learned more about how an intelligent planning system should interact with intelligent control systems. With the integration of expert planning modules, we are now ready to proceed toward the design of a distributed, hierarchical planning system.

The NBS Automated Manufacturing Research Facility is partially supported by the Navy Manufacturing Technology Program.

References

Allen, E.M., "YAPS: Yet Another Production System", University of Maryland TR-1146, (1983).

Brown, P. and McLean, C., "Interactive Process Planning in the AMRF", Proceedings of Winter 1986 ASME Conference, Anaheim, California, December 1986.

Chang, T-C., "Advances in Computer-Aided Process Planning", NBS Report NBS-GCR 83-441, Gaithersburg, MD, (1983).

Chang, T-C. and Wysk, R.A., "An Introduction to Automated Process Planning Systems", Prentice-Hall, Englewood Cliffs, NJ, (1985).

Carlyle, S.M., Barr, B.G., Faddis, T.N. and Umholtz, R., "Automated Fixturing System", Internal Report, Computer Integrated Manufacturing Laboratory, University of Kansas, (1986).

Dixon, J.R., "Artificial Intelligence and Design: A Mechanical Engineering View", Proceedings of AAAI-86, Vol. 2, Philadelphia, PA, 1986.

Furlani, C. et al., "The Automated Manufacturing Research Facility of the National Bureau of Standards", Proc. of the Summer Simulation Conference, Vancouver, BC, Canada, (1983).

Hocken, R. and Nanzetta, P., "Research in Automated Manufacturing at NBS", Manufacturing Engineering, 91, #4, 68, (1983).

Hopp, T., "AMRF Database Report Format: Part Model", NBS Internal Report, (in preparation), (1987).

Hummel, K., "An Expert Systems Based Machine Tool Planner for a Distributed Automated Process Planning System", Masters Thesis, University of Kansas, 1985.

Hummel, K. and Brooks, S., "Symbolic Representation of Manufacturing Features for an Automated Process Planning System", Proceedings of Winter 1986 ASME Conference, Anaheim, California, December 1986.

Kramer, T. and Jun, J., "Software for An Automated Machining Workstation", Proceedings of the 3rd Biennial International Machine Tool Technical Conference, September 1986.

McMahon, R.L. et al., "Manufacturing Technology for an Advanced Machining System, Sixth Semiannual Report", General Dynamics Report MT-87-004, Fort Worth, TX, (1987).

McLean, C., Mitchell, M. and Barkemeyer, E., "A Computing Architecture for Small Batch Manufacturing", IEEE Spectrum, 59, May 1983.

McLean, C., "An Architecture for Intelligent Manufacturing Control", Proceedings of Summer 1985 ASME Conference, Boston, Massachusetts, August 1985.

Nanzetta, P., "Update: NBS Research Facility Addresses Problems in Setups for Small Batch Manufacturing", Industrial Engineering, 68, June (1984).

Nau, D.S., "Expert Computer Systems and Their Applicability to Automated Manufacturing", NBS Report NBSIR 81-2466, Gaithersburg, MD, (1982).

Nau, D.S. and Luce, M., "Knowledge Representation and Reasoning Techniques for Process Planning: Extending SIPS to do Tool Selection", CIRP International Seminar on Manufacturing Systems, University Park, PA, 1987.

Nau, D.S. "Hierarchical Abstraction for Process Planning", to appear in Second Intl. Conference on Applications of Artificial Intelligence in Engineering, Boston, MA, 1987.

Ray, S., "A Knowledge Representation Scheme for Processes in an Automated Manufacturing Environment", Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Atlanta, Georgia, October 1986.

Sacerdoti, E.D., "A Structure for Plans and Behavior", Elsevier North-Holland, New York, NY (1977)

Simpson, J.A., Hocken, R.J. and Albus, J.S., "The Automated Manufacturing Research Facility of the National Bureau of Standards", Journal of Manufacturing Engineering, 1, #1, 18, (1982).

Tu, J. and Hopp, T., "Part Geometry Data in the AMRF", NBS Internal Report (in preparation), (1987).

Unger, M., "BaseplateTool", private communication, (1987).

Hierarchical Abstraction of Problem-Solving Knowledge

Dana S. Nau

Computer Science Department and
Institute for Advanced Computer Studies
University of Maryland

and

Factory Automated Systems Division
Center for Manufacturing Engineering
National Bureau of Standards
Gaithersburg, MD 20899

Presented by:

Dana S. Nau
ASME Winter Annual Meeting, Anaheim, California
December, 1986

Bibliographic Reference:

Nau, D. S., "Hierarchical Abstraction of Problem-Solving Knowledge", Bound volume of the 1986 ASME Winter Annual Meeting, Anaheim, CA, December 1986

Hierarchical Abstraction of Problem-Solving Knowledge

Dana S. Nau*
Computer Science Dept., and Institute for Advanced Computer Studies
University of Maryland

February 3, 1987

Abstract

In most frame-based reasoning systems, the data manipulated by the system is represented using frames, and the problem-solving knowledge used to manipulate this data consists of rules. However, rules are not always the best way to represent problem-solving knowledge.

This paper describes an alternative way to represent problem-solving knowledge called *hierarchical knowledge clustering*. Hierarchical knowledge clustering has been implemented in a system called SIPS (Semi-Intelligent Process Selector), which plans what machining processes to use in manufacturing metal parts. The paper describes the approach to knowledge representation and problem solving used in SIPS, and compares and contrasts this approach to other work.

Primary topic: Knowledge Representation.
Other related topics: Engineering Problem Solving, Expert Systems.

Author's address:

Dana S. Nau
Computer Science Dept.
University of Maryland
College Park, MD 20742
(301) 454-7932
dsn@mimsy.umd.edu

1 Introduction

In most frame-based reasoning systems, the information being manipulated is represented using frames, and the problem-solving knowledge that manipulates the frames consists of rules. But for some problem domains, rules may not be the most natural way to represent knowledge—and in addition, rule-based systems can require large amounts of computation during problem solving if the rule base is large.

This paper describes a way to address these problems using *hierarchical knowledge clustering*, a technique for hierarchical abstraction of problem-solving information. For some problem domains, this approach can be more natural and more efficient than rule-based problem solving.

*This work has been supported in part by the following sources: an NSF Presidential Young Investigator Award to Dana Nau, NSF Grant NSFD CDR-85-00108 to the University of Maryland Systems Research Center, IBM Research, General Motors Research Laboratories, Martin Marietta Laboratories, and the National Bureau of Standards.

R_1: IF goal(h) & $A(h)$ & $B(h)$
THEN assert twist-drilling(h)

R_2: IF goal(h) & $A(h)$ & $C(h)$ & $D(h)$
THEN remove goal(h); assert rough-boring(h); $g = f_1(h)$; assert goal(g)

R_3: IF goal(h) & $A(h)$ & $C(h)$ & $E(h)$
THEN remove goal(h); assert finish-boring(h); $g = f_2(h)$; assert goal(g)

Figure 1: A simple set of rules. A, B, C, D, and E are different sets of restrictions.

Hierarchical knowledge clustering has been implemented in a system called SIPS (Semi-Intelligent Process Selector) [18]. SIPS was developed to produce plans of action for the creation of metal parts using metal removal operations such as milling, drilling, reaming, etc. Each of these operations or *machining processes* creates a *feature* on the metal part, such as a hole, slot, pocket, etc. Given the specification for the final part, the task of deciding what sequence or sequences of machining processes to use in creating the part is known as *process selection*. To do process selection, SIPS starts with the specification of the part to be produced, and reasons about the intrinsic capabilities of each machining process.

SIPS has recently been interfaced to a solid modeling system at General Motors Research Laboratories. This interface allows the user to create part descriptions graphically, and have SIPS select suitable machining processes to create these parts. Also, SIPS has recently been extended to do not just process selection, but also tool selection and the determination of process parameters. The latest version of SIPS is being integrated into the Automated Manufacturing Research Facility (AMRF) project [2] at the National Bureau of Standards.

This paper gives an overview of SIPS. Section 2 explains the motivation for the hierarchical knowledge clustering technique, and Section 3 explains how this technique has been implemented in SIPS. Section 4 discusses the relationships between SIPS and work by others, and Section 5 contains concluding remarks.

2 Motivation

In most knowledge-based problem-solving systems, problem-solving knowledge consists of rules of the form "IF *conditions* THEN *action*". Even in frame systems, where the data (and possibly the knowledge base) are represented using frames, the knowledge base still usually consists of rules. However, there are several problems with using this approach for process selection.

Consider the problem of creating a hole h. There are many machining processes capable of creating holes, but to keep the example simple, suppose we consider only three processes: twist drilling, rough boring, and finish boring. Each of these processes has different restrictions how good a hole it can produce. If the restrictions for twist drilling are satisfied, twist drilling can produce h without requiring that anything else be done. However, rough boring (if its restrictions are satisfied) produces h by modifying a hole g which must already be present. Finish boring is similar to rough boring, except that it can satisfy stricter machining tolerances for h. One way to describe these processes would be rules similar to those shown in Figure 1.

One problem with these rules is the repititiousness of their preconditions: each rule tells what

R_4: IF **goal**(h) & $A(h)$
THEN remove **goal**(h); assert **hole-process**(h)

R_5: IF **hole-process**(h) & $B(h)$
THEN remove **goal**(h); assert **twist-drilling**(h)

R_6: IF **hole-process**(h) & $C(h)$
THEN remove **goal**(h); assert **hole-improve-process**(h)

R_7: IF **hole-improve-process**(h) & $D(h)$
THEN remove **goal**(h); assert **rough-boring**(h); $g = f_1(h)$; assert **goal**(g)

R_8: IF **hole-improve-process**(h) & $E(h)$
THEN remove **goal**(h); assert **finish-boring**(h); $g = f_2(h)$; assert **goal**(g)

Figure 2: A better set of rules.

distinguishes some machining process from every other machining process in the entire knowledge base. It would be more natural and (depending on the control strategy) probably more efficient to set up context in which hole processes are the only processes being considered, and then describe each hole process only in terms of what distinguishes it from the other hole processes. This approach would lead to rules such as those shown in Figure 2.

Another problem is how to select the appropriate rule when more than one rule is applicable. For example, suppose both **twist-drilling** and **hole-improve-process** are capable of creating h. Since **twist-drilling** is less costly, one would want to use R_5 instead of R_6, but the rules include no way to assure that this will happen.

This problem could be handled if one could attach priorities to the rules corresponding to the costs of the machining processes—and rule-based systems sometimes include ways to do this. But in this case, it is not so easy: the priorities are not available beforehand to put into the rules, but instead are functions of the various machining processes. For example, the cost of a hole improvement process should be computed as the minimum of the cost of rough boring and finish boring.

One way to handle this is to notice that the rules in Figure 2 correspond to the tree shown in Figure 3. By representing each node in the tree as a frame, one could represent the process costs as slots whose values could be computed as functions of other frames. Additional slots could represent various other relevant properties of the processes—feed rates, cutting speeds, location of the machine in the factory, etc.

If we represent the machining processes in this fashion, the next question is how to represent and invoke the IF and THEN parts of the rules. Although message passing is often used in frame systems, it would not work well here, because it would still a process even if a less costly process were applicable. In order to make sure that only the least-cost frames get activated, a global control strategy is needed to supervise the activation of the frames. The combination of the hierarchical representation with such a control strategy is called *hierarchical knowledge clustering*.

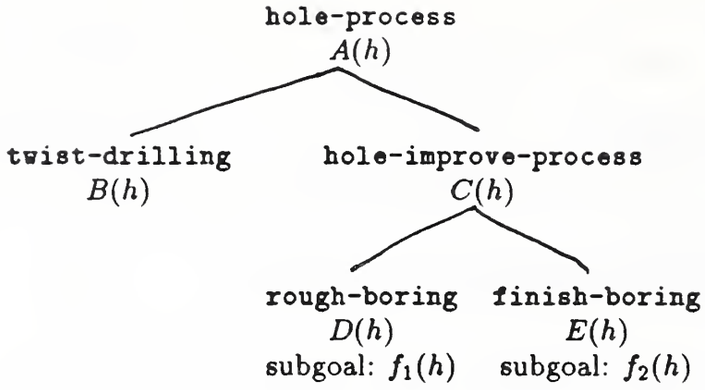


Figure 3: A tree corresponding to the rules in Figure 2.

3 Implementation

Hierarchical knowledge clustering has been implemented in a system called SIPS. SIPS includes a frame system which can be used to represent both static knowledge (e.g., representations of three-dimensional objects) and problem-solving knowledge (as discussed in Section 2).

Figure 4 shows a frame structure corresponding to the tree shown in Figure 3. This frame structure is much simpler than the knowledge base actually used in SIPS, but it illustrates how SIPS represents problem-solving knowledge.

The `relevant` slot in the `hole-process` frame specifies that a hole process is relevant for making a hole. This information is used to start SIPS's search when SIPS is told to find plan the creation of a hole.

The `cost` slot is intended to be a lower bound on the cost of performing a process. In the case of `hole-process`, this lower bound is computed by an attached procedure which takes the minimum of the `cost` slots of the child frames. `hole-improve-process` inherits this procedure from `hole-process`, so its cost will also be computed as the minimum of the costs of its children. Since the `twist-drilling`, `rough-boring`, and `finish-boring` frames represent single kinds of machining processes rather than classes of machining processes, the relative costs of these processes are put into their `cost` slots.

Similarly, `precost` is intended to be a lower bound on the cost of any other processes which might be required before doing the hole process. For `hole-process`, this bound is computed by an attached procedure which computes the minimum of the `precost` slots of the children. Since `twist-drilling` does not need to have any other processes occur before it, its `precost` slot contains the value 0. But a hole improvement process takes an existing hole g and transforms it into the desired hole—and since g must be created by some kind of hole process, the cost of creating g will be at least the minimum cost for a hole process. Thus, the `precost` slot for `hole-improve-process` is the value of `hole-process`'s `cost` slot. Both `rough-boring` and `finish-boring` inherit this value from `hole-improve-process`.

A process's `restrictions` slot tells what restrictions must be satisfied in order for that process to be a feasible way to achieve the desired goal. For `hole-process`, the restrictions are mainly geometric ones—for example, restrictions on the angle between the hole and the surface in which it is to be created. For the other processes in Figure 4, the restrictions are mainly restrictions on the hole dimensions and on the best machining tolerances achievable by the process (parallelism, roundness, true position, etc.).

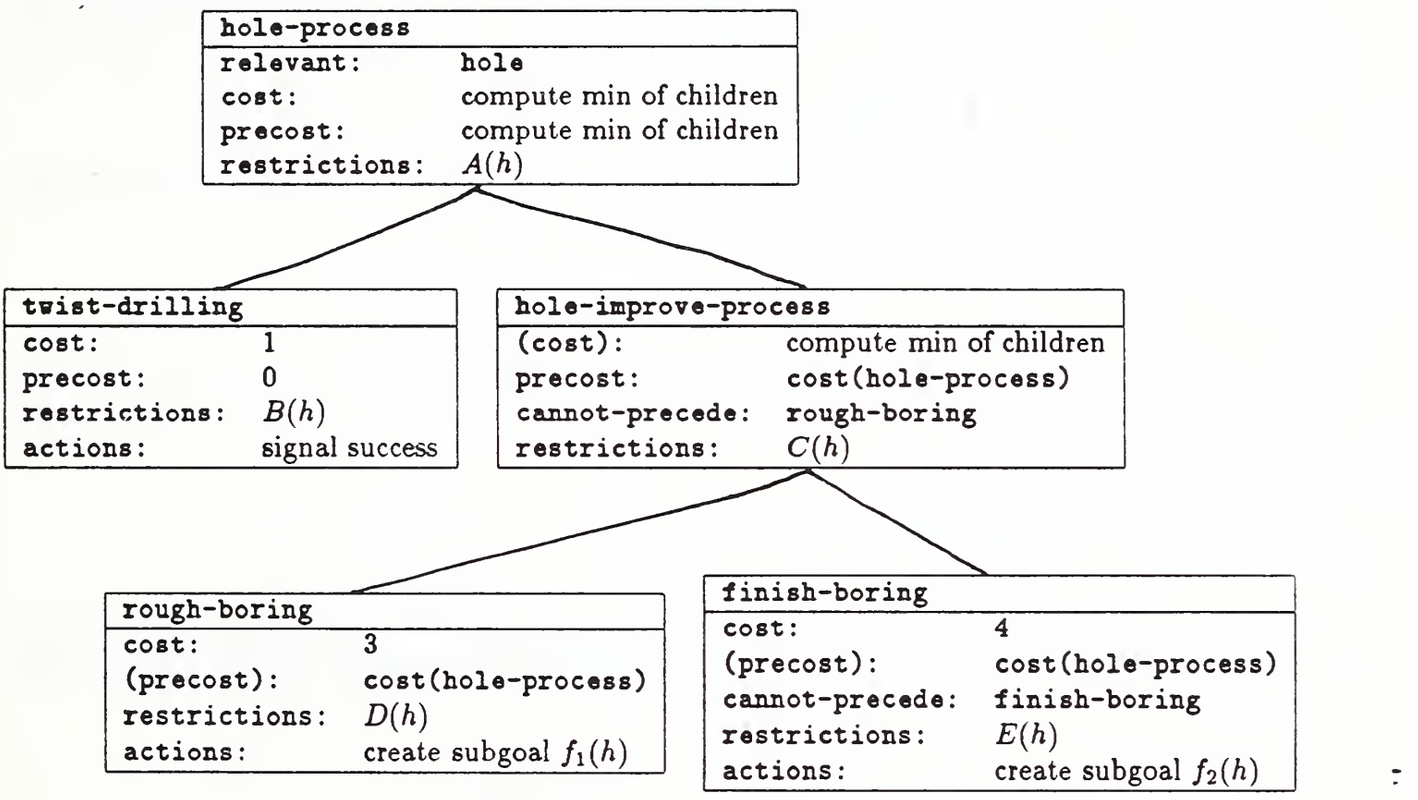


Figure 4: A frame structure corresponding to the tree shown in Figure 3. Parentheses around a slot name indicate that the slot is inherited from the parent frame.

The `cannot-precede` slots for `hole-improve-process` and `finish-boring` state that in no sensible process plan will these processes be followed by certain other machining processes. This slot is not really necessary for correct operation of SIPS, but it makes SIPS more efficient by decreasing the size of the search space.

SIPS does problem solving by searching backwards from the ultimate goal to be achieved. Therefore, the `actions` slot for a machining process must specify what SIPS needs to do *before* it can perform the machining process. For `twist-drilling`, nothing need be done beforehand—so `twist-drilling`'s `actions` slot states that twist drilling succeeds immediately. However, rough boring and finish boring produce a better hole from an existing hole—and SIPS needs to figure out how to make this hole. The `actions` statements for `rough-boring` and `finish-boring` set up the creation of this hole as a subgoal for SIPS.

Figure 5 shows part of the state space which can be generated from the set of frames shown in Figure 4. Each state in the state space is a (partial) plan for creating a hole h1. Whether or not this plan is feasible will depend on the nature of h1—except that the plans marked "infeasible" in Figure 5 can never be feasible, because of the `cannot-precede` slots in the knowledge base. When a plan is infeasible, its children will never be generated.

SIPS searches the state space using an adaptation of Branch and Bound. The lower bound function LB which guides this search is computed from the `cost` and `precost` slots of the machining processes. For example, for the plan labeled P in Figure 4,

$$LP(P) = \texttt{precost(hole-process)} + \texttt{cost(hole-process)} + \texttt{cost(finish-boring)}.$$

So that SIPS will avoid generating expensive plans when cheaper ones can be used, SIPS's search strategy is best-first.[1] Thus, the first solution found by SIPS is guaranteed to be the least costly one.

4 Relation to Other Work

This section discusses the relationships between SIPS and other work in three areas: automated process planning, planning with abstraction, and computational approaches for knowledge-based systems.

4.1 Process Planning

A number of computer systems exist which provide partial automation of process planning. In most existing systems, process planning is done by retrieving from a data base a process plan for another part similar to the desired part, and modifying this plan by hand to produce a process plan for the desired part. Examples of such systems are CAPP [12] and MIPLAN [24]. For more detailed descriptions of such systems, the reader is referred to [4] and [19].

Devising a complete process plan automatically using a part's specifications (e.g., a full technical drawing) is a very difficult problem. There are several systems which attempt to produce a process plan for the exact part desired—but most such systems are experimental and have limited capabilities. A few of the better-known systems include CPPP [8], APPAS [26], CADCAM [3,6], TIPPS [5], GARI [7] and TOM [13], and SIPP [16,17] (a predecessor to SIPS, implemented in Prolog). Except for SIPP, these systems use problem-solving approaches rather different from what is used in SIPS.

[1] Thus, SIPS's search procedure may also be thought of as an adaptation of A* [20], with LB as the heuristic function.

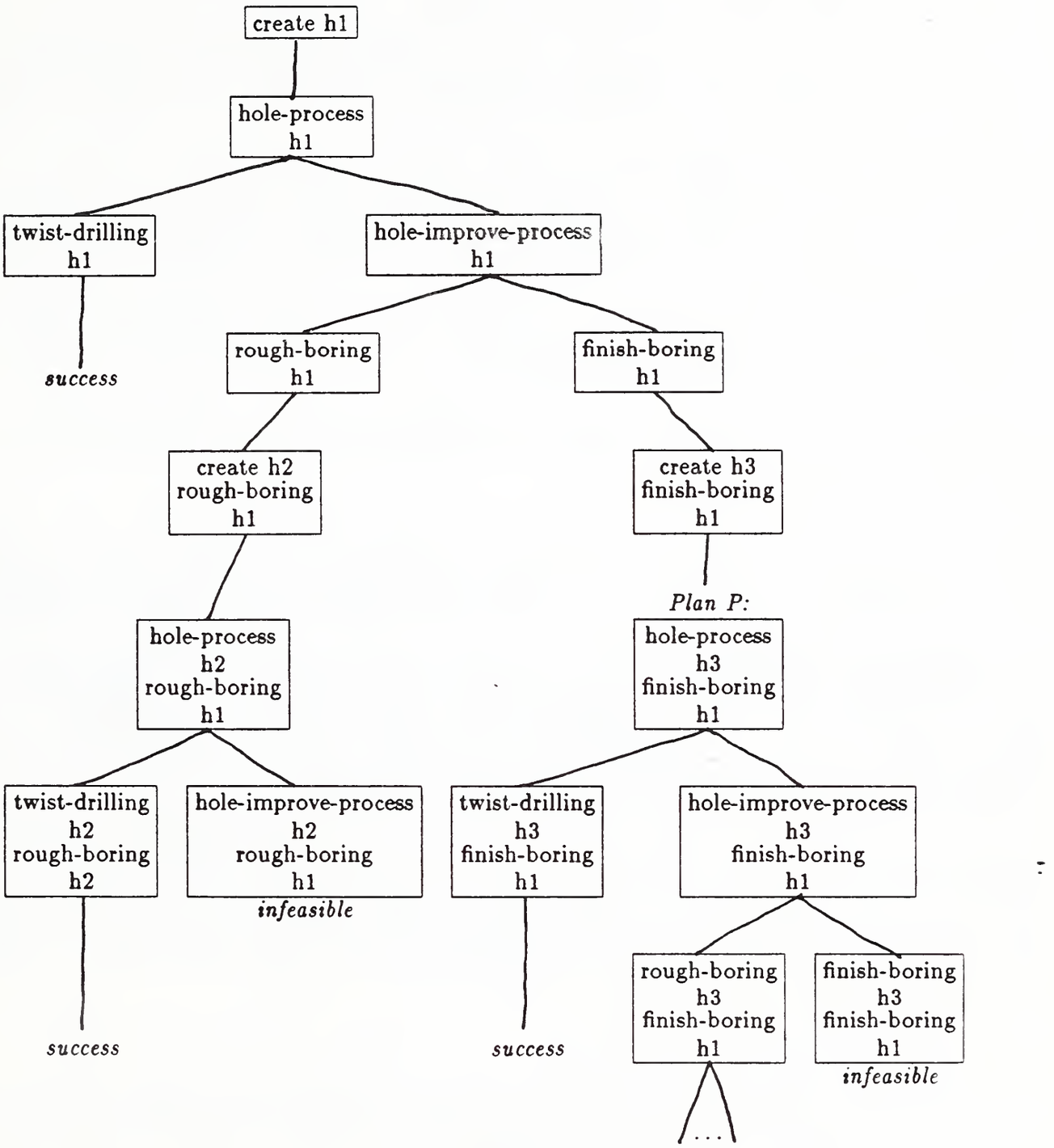


Figure 5: Part of a search space for creating a hole h1. Plan P is labeled for reference in the text.

4.2 Planning with Abstraction

Hierarchical knowledge clustering can be viewed as a way to do planning based on abstraction. For example, the `hole-process` frame in Figure 4 represents an abstract machining process which has two possible instantiations: `twist-drilling` and `hole-improve-process`.

Several types of abstraction have been explored in the literature on planning. One type of abstraction is that used in NOAH [23], in which an action A is an abstraction of actions A_1 and A_2 if A_1 and A_2 are each steps in the performance of A. This is rather different from the abstraction used in SIPS: in SIPS, A is an abstraction of actions A_1 and A_2 if A_1 and A_2 are alternate instantiations of A.

Another type of abstraction is that used in ABSTRIPS [20], in which a complete plan is constructed ignoring some of the preconditions of each action and the plan is then modified to meet the preconditions which were ignored. This type of abstraction is related to that used in SIPS in the following sense: an instantiation of an action A is an action A_1 which must satisfy the preconditions of A and also some additional preconditions, and both SIPS and ABSTRIPS refine a plan containing A by checking those preconditions of A_1 which differ from the preconditions of A. However, there are several important differences:

1. SIPS completely instantiates the last action in a plan before considering what actions should precede this action, whereas ABSTRIPS generates a complete (but possibly incorrect) plan and then tries to fix it up.

2. In SIPS, an abstract action has several possible alternate instantiations, but in ABSTRIPS, only one instantiation is possible. Thus in ABSTRIPS, the notion of considering alternate instantiations of an action and choosing the one of least estimated cost does not make sense.

Another type of abstraction which is quite close to that used in SIPS is proposed by Tenenberg [22]. This approach is similar to SIPS in the sense that each abstract action may have more than one possible instantiation. It is potentially more general than that used in SIPS, in the sense that the effects of actions are represented hierarchically, as well as their preconditions—but so far, Tenenberg's approach approach has not yet been implemented.

Several systems for diagnostic problem-solving make use of certain kinds of taxonomic hierarchies. Both MDX [14] and Centaur [10] use taxonomies of various diagnostic problems, in which knowledge about each class of problems is located at the node in the hierarchy which represents that class. These approaches yield some of the same benefits as SIPS in terms of representational clarity and efficiency of problem-solving. However, the details of how they represent and manipulate their knowledge are rather different from what SIPS does.

4.3 Computational Approaches

It is well known that rule-based systems having large rule bases can require substantial computational overhead. Suppose a rule-based system is trying to solve a problem in some problem domain D. Each time the system applies a rule, this changes the system's current state S—and in order to decide what rule to apply next, the system must determine which rules match S. If the system searched through its entire set of rules to find the ones matching S, the computational overhead would be tremendous.

Several approaches have been tried for alleviating this problem. One approach, which is used in KEE [9], is to provide facilities whereby the user can divide a set of rules R into smaller subsets $R_1, R_2, \ldots, R_n$, such that each subset is relevant for a different problem domain. Given a problem

to solve, the system starts out by determining which problem domain the problem is in. It then selects the rule set R_i for that domain, and then uses R_i exclusively from that point on, ignoring all the other rules. Since R_i is smaller than R, the problems with efficiency are lessened.

Hierarchical knowledge clustering can be thought of as as an extension of the above approach. It provides a way to tell, directly from the current state S, that only some subset R_S of the rules in R is relevant to S.[2] Thus, all rules not in R_S can temporarily be ignored. Since R_S is normally quite small, this provides improved efficiency.

Another approach to reducing the computational overhead of computing rule matches is the rete match algorithm used in OPS5 [11] and YAPS [1]. This algorithm provides a way to store partial rule matches in a network so that the system can determine whether a rule matches the current state without having to re-evaluate all of the rule's preconditions each time the current state changes. This makes the complexity of computing rule matches depend not on the size of R, but instead on the size of the set P_S of rules whose preconditions partially match S. If P_S is small, then the rete match procedure is efficient, but if P_S is large, the elaboration of partial matches may incur significant overhead.

Hierarchical knowledge clustering can be thought of as a way to control the elaboration of partial matches, by distributing the preconditions of a rule throughout the levels of a hierarchical structure and elaborating a partial match only if it looks promising. Thus, the approach used in SIPS may have potential for increasing the efficiency of the rete match procedure.

5 Concluding Remarks

SIPS currently runs in Franz Lisp on a Sun, and in Zeta Lisp on a Symbolics Lisp Machine and a TI Explorer. It can either read prepared data from a file, or (if some of this data is omitted) run interactively, asking the user for any needed information. Various user features have been implemented in SIPS. For example, if SIPS produces a plan for producing some feature, the user can later tell SIPS to go back and find other alternative plans for producing this feature.

For the process planning problem domain, hierarchical knowledge clustering appears to be more natural to use than a "flat" set of production rules. In the experience of a manufacturing engineer who has worked on SIPS's knowledge base, SIPS's style of knowledge representation has been easy to understand and use. Trying to represent SIPS's knowledge base as a rule-based system would make the rules very cumbersome.

A more sophisticated interface for SIPS is currently being developed. SIPS has been interfaced to a solid modeling system at General Motors Research Laboratories, so that the user can build up an object to be created by giving graphical specifications of its machinable features, and have SIPS select sequences of machining processes capable of creating those features. Further work on solid modeling for SIPS is currently underway [25].

More recently, SIPS has been extended to do not just process selection, but also tool selection and the determination of process parameters. This has been done by giving SIPS a knowledge base for tooling in addition to its knowledge base for process selection. Thus, the current knowledge base for SIPS consists of three hierarchies: a taxonomy of machinable features, a taxonomy of machining processes, and a taxonomy of cutting tools. Once SIPS finds a successful sequence of machining processes for a given machinable surface, it uses its knowledge about the characteristics of each cutting tool to decide, for each machining process, what cutting tool to use and what

[2] In particular, finding R_S corresponds either to retrieving the children of some frame or (when SIPS creates a subgoal) retrieving all frames relevant to the creation of a feature. In each case, only a few of SIPS's process frames are relevant—and which frames are relevant is determined easily from the frame system.

process parameters to use. The latest version of SIPS is being integrated into the Automated Manufacturing Research Facility (AMRF) project [2] at the National Bureau of Standards.

References

[1] E. M. Allen, "Yaps: Yet Another Production System," Tech. Report TR-1146, Computer Science Dept., Univ. of Maryland, College Park, MD, Feb. 1982.

[2] P. F. Brown and C. R. McLean, "Interactive Process Planning in the AMRF," *Symposium on Knowledge-Based Expert Systems for Manufacturing at ASME Winter Annual Meeting*, Anaheim, CA, Dec. 1986, pp. 245-262.

[3] T. C. Chang, "Interfacing CAD and CAM - A Study of Hole Design," M.S. Thesis, Virginia Polytechnic Institute, 1980.

[4] T. C. Chang and R. A. Wysk, *An Introduction to Automated Process Planning Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

[5] T.C. Chang and R. A. Wysk, "Integrating CAD and CAM through Automated Process Planning," *International Journal of Production Research* 22:5, 1985.

[6] T. C. Chang and R. A. Wysk, "An Integrated CAD/Automated Process Planning System," *AIIE Transactions* 13:3, Sept. 1981.

[7] Y. Descotte and J. C. Latombe, "GARI: A Problem Solver that Plans How to Machine Mechanical Parts," *Proc. Seventh International Joint Conf. Artif. Intel.*, Aug. 1981, pp. 766-772.

[8] M. S. Dunn, Jr., J. D. Bertelsen, C. H. Rothauser, W. S. Strickland, and A. C. Milsop, "Implementation of Computerized Production Process Planning," Report $R_8$1-945220-14, United Technologies Research Center, East Hartford, CT, June 1981.

[9] R. Fikes and T. Kehler, "The Role of Frame-Based Representation in Reasoning," Communications of the ACM **28:9**, Sept. 1985, pp. 904-920.

[10] P. Jackson, "Introduction to Expert Systems," Addison-Wesley, Wokingham, England, 1986, pp. 142-157.

[11] C. L. Forgy, "The OPS5 User's Manual," Tech. Report CMU-CS-81-135, Computer Sci. Dept., Carnegie-Mellon University, 1980.

[12] C. H. Link, "CAPP—CAM-I Automated Process Planning System," *Proc. 13th Numerical Control Society Annual Meeting and Technical Conference*, Cincinnati, March 1976.

[13] K. Matsushima, N. Okada, and T. Sata, "The Integration of CAD and CAM by Application of Artificial-Intelligence," *CIRP*, 1982, pp. 329-332.

[14] S. Mittal, B. Chandrasekaran, and J. Smith, "Overview of MDX–A System for Medical Diagnosis," *Proc. Third Annual Symposium on Computer Applications in Medical Care*, Washington, DC, Oct. 1979.

[15] D. S. Nau and T. C. Chang, "Prospects for Process Selection Using Artificial Intelligence," *Computers in Industry* 4, 1983, pp. 253-263.

[16] D. S. Nau and T. C. Chang, "A Knowledge-Based Approach to Generative Process Planning," *Production Engineering Conference at ASME Winter Annual Meeting*, Miami Beach, Nov. 1985, pp. 65-71.

[17] D. S. Nau and T. C. Chang, "Hierarchical Representation of Problem-Solving Knowledge in a Frame-Based Process Planning System," *Jour. Intelligent Systems* 1:1, 1986, pp. 29-44.

[18] D. S. Nau and M. Gray, "SIPS: An Application of Hierarchical Knowledge Clustering to Process Planning," *Symposium on Integrated and Intelligent Manufacturing at ASME Winter Annual Meeting*, Anaheim, CA, Dec. 1986, pp. 219-225.

[19] D. S. Nau, J. A. Reggia, M. W. Blanks, Y. Peng, and D. Sutton, "Prospects for Knowledge-Based Computing in Automated Process Planning and Shop Control," Tech. Report, Computer Science Dept., University of Maryland, College Park, Feb. 1984.

[20] N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Press, Palo Alto, 1980, pp. 350-354.

[21] C. Ramsey, J. A. Reggia, D. S. Nau, and A. Ferrentino, "A Comparative Analysis of Methods for Expert Systems," *Internat. Jour. Man-Machine Studies*, 1986.

[22] J. Tenenberg, "Planning with Abstraction," *Proc. AAAI-86*, Philadelphia, 1986, pp. 76-80.

[23] E. Sacerdoti, *A Structure for Plans and Behavior*, American Elsevier, New York, 1977.

[24] TNO, "Introduction to MIPLAN," Organization for Industrial Research, Inc., Waltham, MA, 1981.

[25] G. Vanecek and D. Nau, "Computing Geometric Boolean Operations by Input Directed Decomposition," in preparation, 1987.

[26] R. A. Wysk, "An Automated Process Planning and Selection Program: APPAS," Ph.D. Thesis, Purdue University, 1977.

BRIEF INTRODUCTION TO BACKUS-NAUR (BNF) NOTATION:

The following are symbols of BNF, and not of the language itself:

1. <xx> Denotes a non-terminal symbol whose name is "xx". A "non-terminal" is a symbol of the BNF notation which can be further decomposed into a set of non-terminals and/or terminals. Eventually, all symbols decompose into terminals. A "terminal" is a symbol or character of the object language. The "object language" consists of all symbols and characters that will appear in the actual file.

2. ::= The non-terminal to the left of this symbol is composed of all those elements that are to the right of this symbol (expresses decomposition).

3. <xx> <yy> This expresses concatenation of two non-terminals ("and"). The concatenation applies to whatever these non-terminals decompose to as well.

4. | This means "or" (any one of the specified elements may be chosen to place in this position).

5. { } Means zero or more occurrences of. For example, {<header_line>} means the same as <header_line_1> . . <header_line_n> .

6. [ ] Means optional.

7. xx . . yy This notation is for numeric or alphabetic ranges.

8. <???> Means as yet undefined.

Simple example of BNF:

```
<a> ::= <b> <c> !
<b> ::= hi
<c> ::= there
     becomes
hi there !
```

Comments:

1. Keywords, values, and parameters are to be 19 characters or less.

2. All letters are uppercase.

3. Any terminal (i.e. punctuation_mark, integer) may be preceded and followed by whitespace (defined below in BNF) unless otherwise specified.

4. The notation "xxH" in the following BNF represents the hexadecimal number specifying an ascii character. For instance, 2H means SPACE.

5. Any element may be omitted by delineating with the proper punctuation. For instance, in order to specify no precedence steps, two consecutive semi-colons may be used.

THE PROCESS PLAN SPECIFICATION IN BNF

```
<pp_file> ::=    --PROCESS_PLAN--
                      <parameters_section>
                      <header_section>
                      <rqmts_list>
                      <procedure_specification>
                 --END_PROCESS_PLAN--

<header_section> ::= --HEADER_SECTION--
                              {<header_line>}
                      --END_HEADER_SECTION

      <header_line> ::= <header_elem_name> =  <value> .

      <header_elem_name> ::= <keyword>

<parameters_section> ::= --PARAMETERS_SECTION--
                                 {<parm_line>}
                          --END_PARAMETERS_SECTION--

      <parm_line> ::=  <parm_name> ; <parm_type> ;
                     <parm_range> ;  <parm_default> .

      <parm_name> ::= $$<keyword>

      <parm_type> ::= <???>

      <parm_range> ::= <value> , <value>

      <parm_default> ::= <???>
```

```
<rqmts_list> ::= --REQUIREMENTS_SECTION--
                      {<rqmt_line>}
                  --END_REQUIREMENTS_SECTION

      <rqmt_line> ::= <rqmt_number> ; <rqmt_identifier> ;
                      <rqmt_type> ; <rqmt_description> ;
                      <rqmt_quantity> ; <parent_rqmts> .

      <rqmt_number> ::= <integer>

      <rqmt_identifier> ::= <keyword>

      <rqmt_type> ::= <keyword>

      <rqmt_description> ::= <???>

      <rqmt_quantity> ::= <integer>

      <parent_rqmts> ::= <rqmt_line_num_list>

      <rqmt_line_num_list> ::=  {<keyword> ,} <keyword>


<procedure_specification> ::= --PROCEDURE_SECTION
                                    {<procs_line>}
                               --END_PROCEDURE_SECTION--

      <procs_line> ::= <step_number> ; <work_descr> ;
                       <prec_steps> ; <duration> .

      <step_number> ::= <integer>

      <work_descr> ::= <work_element_name>
                       {, <keyword> = <value>}

      <work_element_name> ::= <keyword>

      <prec_steps> ::=  {<integer> ,} <integer>

      <duration> ::= " <days> : <hrs> : <min> : <sec> "
     (No whitespace allowed between characters)

      <days> ::= <digit> <digit> <digit> <digit>
          (No whitespace allowed between digits)

      <hrs> ::= 00 . . 23

      <min> ::= 00 . . 59
```

```
        <sec> ::= 00 . . 59



<keyword> ::= <keyword_prefix> {<uppercase_letter> | <digit> |
              $ | # | _ | - | % | & | + | ! | @ | *}
              (No whitespace allowed between characters)

<keyword_prefix> ::= <uppercase_letter> | $ | # | @

<value> ::= <number> | <keyword> | <string>

<string> ::= " {<ascii_printable_char>} "

<whitespace> ::= { CR | LF | SPACE | TAB | FORMFEED }

<upper_case_letter> ::= A . . Z     ( 41H . . 5AH )

<digit> ::= 0 . . 9       ( 30H . . 39H )

<integer> ::= <digit> | <digit> <digit>
              (No whitespace allowed between digits)

<ascii_printable_char> ::= SPACE . . ~ | TAB | FORM-FEED |CR | VT
          | LINE-FEED ( 20H . . 7EH | 12H | 10H | 9H | 0BH | 0AH )
    (Note: " , or 22H,  must be preceded by \ , or 5CH and also
  \, or 5CH, must be preceded by \ , or 5CH)

<file_keyword> ::= --PROCESS_PLAN-- | --END_PROCESS_PLAN--

<section_keyword> ::= --PARAMETERS_SECTION-- |
                      --END_PARAMETERS_SECTION-- |
                      --HEADER_SECTION-- |
                      --END_HEADER_SECTION-- |
                      --REQUIREMENTS_SECTION-- |
                      --END_REQUIREMENTS_SECTION-- |
                      --PROCEDURE_SECTION-- |
                      --END_PROCEDURE_SECTION--

<punctuation_mark> ::= ; | = | . | : | ,
                       ( 3BH | 3DH | 2EH | 3AH | 2CH )


<number> ::= <integer> | <integer>.<integer> |
 <integer>[.<integer>]E<exponent>
             (No whitespace allowed between characters)
```

<exponent> ::= [+]<integer> | -<integer>
(No whitespace allowed between characters)

```
--PROCESS_PLAN--

--HEADER_SECTION--

   PLAN-ID           := PP-CELL-1;
   PLAN-VERSION      := 1;
   PLAN-TYPE         := ROUTING-SLIP;
   PLAN-NAME         := "FILTER-HOUSING";

   PROCESS-ENGINEER  := "Peter Brown";
   PART-NUMBER       := 31;
   GT-CODE           := 0134673689;
   ENG-DRAW-#        := 123987;
   ENG-REVISION      := 2;

--END_HEADER_SECTION--


--PARAMETERS_SECTION--

   $$TRAY001       : PART-TRAY;
   $$TRAY002       : TOOL-TRAY;
   $$LOT001        : LOT;
   $$TOOL-SET001   : TOOL-SET;

--END_PARAMETERS_SECTION--


--REQUIREMENTS_SECTION--

<<1>>  PROCESS-PLAN
           ( PLAN-ID              => PP-MHS-1,
             PLAN-VERSION         => 1,
             PLAN-TYPE            => OPERATION-SHEET,
             PLAN-NAME            => "FILTER-HOUSING" );

<<2>>  PROCESS-PLAN
           ( PLAN-ID              => PP-MHS-2,
             PLAN-VERSION         => 1,
             PLAN-TYPE            => OPERATION-SHEET,
             PLAN-NAME            => "FILTER-HOUSING" );

<<3>>  PROCESS-PLAN
           ( PLAN-ID              => PP-MHS-3,
```

```
              PLAN-VERSION          => 1,
              PLAN-TYPE             => OPERATION-SHEET,
              PLAN-NAME             => "FILTER-HOUSING" );

<<4>>  PROCESS-PLAN
           ( PLAN-ID                => PP-MHS-4,
             PLAN-VERSION           => 1,
             PLAN-TYPE              => OPERATION-SHEET,
             PLAN-NAME              => "FILTER-HOUSING" );

<<5>>  PROCESS-PLAN
           ( PLAN-ID                => PP-VWS-1,
             PLAN-VERSION           => 1,
             PLAN-TYPE              => OPERATION-SHEET,
             PLAN-NAME              => "FILTER-HOUSING" );

<<6>>  PROCESS-PLAN
           ( PLAN-ID                => PP-VWS-2,
             PLAN-VERSION           => 1,
             PLAN-TYPE              => OPERATION-SHEET,
             PLAN-NAME              => "FILTER-HOUSING" );

<<7>>  PROCESS-PLAN
           ( PLAN-ID                => PP-VWS-3,
             PLAN-VERSION           => 1,
             PLAN-TYPE              => OPERATION-SHEET,
             PLAN-NAME              => "FILTER-HOUSING" );

<<8>>  WORKSTATION
           ( WORSTATION-ID          => VWS );

<<9>>  WORKSTATION
           ( WORKSTATION-ID         => MHS );

<<10>>  TRAY
           ( TRAY-TYPE              => SECTOR-4,
             TRAY-ID                => $$TRAY001 );

<<11>>  TRAY
           ( TRAY-TYPE              => SECTOR-4,
             TRAY-ID                => $$TRAY002 );

--END_REQUIREMENTS_SECTION--


--PROCEDURE_SECTION--

<<1>>  DELIVER-TRAY
           ( ORIGIN                 => MHS,
             DESTINATION            => VWS,
```

```
              TRAY-TYPE              => SECTOR-4,
              TRAY-ID                => $$TRAY001,
              SYSTEM                 => MHS,
              TYPE                   => COMPLEX,
              PLAN-ID                => PP-MHS-1,
              PREC-STEPS             => (),
              TIME                   => 0000:00:00:30 );

<<2>>  DELIVER-TRAY
           ( ORIGIN                  => MHS,
             DESTINATION             => VWS,
             TRAY-TYPE               => SECTOR-4,
             TRAY-ID                 => $$TRAY002,
             SYSTEM                  => MHS,
             TYPE                    => COMPLEX,
             PLAN-ID                 => PP-MHS-2,
             PREC-STEPS              => (),
             TIME                    => 0000:00:00:30 );


<<3>>  RECEIVE-TRAY
           ( TRAY-TYPE               => SECTOR-4,
             TRAY-ID                 => $$TRAY001,
             SYSTEM                  => VWS,
             TYPE                    => PRIMITIVE,
             PREC-STEPS              => (1),
             TIME                    => 0000:00:00:30 );

<<4>>  RECEIVE-TRAY
           ( TRAY-TYPE               => SECTOR-4,
             TRAY-ID                 => $$TRAY002,
             SYSTEM                  => VWS,
             TYPE                    => PRIMITIVE,
             PREC-STEPS              => (2),
             TIME                    => 0000:00:00:30 );

<<5>>  SETUP-AREA
           ( AREA-ID                 => TOOL-CHANGER,
             ITEMS                   => $$TOOL-SET001,
             SYSTEM                  => VWS,
             TYPE                    => COMPLEX,
             PLAN-ID                 => PP-VWS-1,
             PREC-STEPS              => (4),
             TIME                    => 0000:00:02:45 );


<<6>>  MACHINE-LOT
           ( LOT-ID                  => $$LOT001,
             LOT-TYPE                => FILTER-HOUSING,
```

```
            QUANTITY            => 4,
            TRAY-ID             => $$TRAY001,
            TOOL-SET            => $$TOOL-SET001,
            SYSTEM              => VWS,
            TYPE                => COMPLEX,
            PLAN-ID             => PP-VWS-2,
            PREC-STEPS          => (3,5),
            TIME                => 0000:00:28:15 );

<<7>>  TAKEDOWN-AREA
         ( AREA-ID              => TOOL-CHANGER,
           ITEMS                => $$TOOL-SET-1
           SYSTEM               => VWS,
           TYPE                 => COMPLEX,
           PLAN-ID              => PP-VWS-3,
           PREC-STEPS           => (6),
           TIME                 => 0000:00:02:30 );

<<8>>  SHIP-TRAY
         ( TRAY-TYPE            => SECTOR-4,
           TRAY-ID              => $$TRAY001,
           SYSTEM               => VWS,
           TYPE                 => PRIMITIVE,
           PREC-STEPS           => (6),
           TIME                 => 0000:00:00:30 );

<<9>>  SHIP-TRAY
         ( TRAY-TYPE            => SECTOR-4,
           TRAY-ID              => $$TRAY002,
           SYSTEM               => VWS,
           TYPE                 => PRIMITIVE,
           PREC-STEPS           => (7),
           TIME                 => 0000:00:00:30 );

<<10>> DELIVER-TRAY
         (  ORIGIN              => VWS,
           DESTINATION          => MHS,
           TRAY-TYPE            => SECTOR-4,
           TRAY-ID              => $$TRAY001,
           SYSTEM               => MHS,
           TYPE                 => COMPLEX,
           PLAN-ID              => PP-MHS-3,
           PREC-STEPS           => (8),
           TIME                 => 0000:00:00:30 );

<<11>> DELIVER-TRAY
         ( ORIGIN               => VWS,
           DESTINATION          => MHS,
           TRAY-TYPE            => SECTOR-4,
```

```
            TRAY-ID                 => $$TRAY002,
            SYSTEM                  => MHS,
            TYPE                    => COMPLEX,
            PLAN-ID                 => PP-MHS-4,
            PREC-STEPS              => (9),
            TIME                    => 0000:00:00:30 );

--END_PROCEDURE_SECTION--


--END_PROCESS_PLAN--
```

The process planning system operates on Symbolics 3600 series computers running the 6.1 version of the operating system. It is written in Zetalisp, and uses Symbolics Flavors. Future work will upgrade the entire system to run under Genera 7 on the Symbolics machines. To operate effectively, the host computer should have at least 4 Mbytes of memory running under Release 6.1, or 8 Mbytes running under Genera 7.

READER COMMENT FORM

Document Title Process Planning System Architecture

This document is one in a series of publications which document research done at the National Bureau of Standards' Automated Manufacturing Research Facility from 1981 through March, 1987.

You may use this form to comment on the technical content or organization of this document or to contribute suggested editorial changes.

Comments:

_____

_____

_____

_____

_____

_____

_____

If you wish a reply, give your name, company, and complete mailing
Address:

_____

_____

_____

What is your occupation ?

_____

NOTE: This form may not be used to order additional copies of this document or other documents in the series. Copies of AMRF documents are available from NTIS.

Please mail your comments to: AMRF Program Manager
National Bureau of Standards
Building 220, Room B-111
Gaithersburg, MD 20899

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

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO. NISTIR 88/3828	2. Performing Organ. Report No.	3. Publication Date MARCH 1989

4. TITLE AND SUBTITLE

NBS AMRF Process Planning System - System Architecture

5. AUTHOR(S)

Peter F. Brown & Steven R. Ray

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)*

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 purpose of this document is to provide a general description of design and implementation of the AMRF Process Planning System. The document should provide the reader with an understanding of the concepts behind the work in the process planning project as well as on the approach adopted. Details on system implementation are provided.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

automation, expert, hierarchical, manufacturing, planning, process planning, standards

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

113

15. Price

$13.95