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

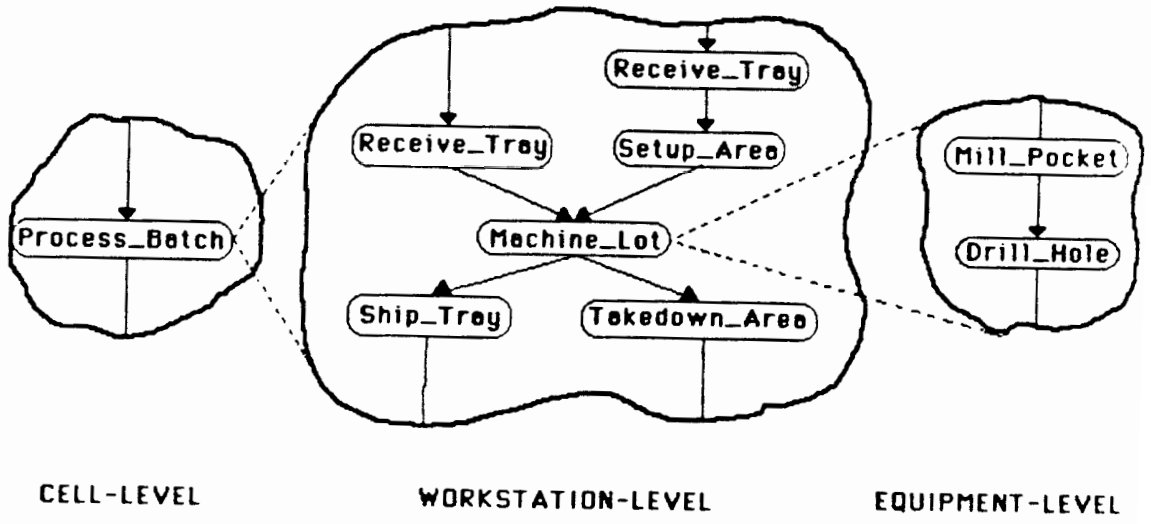

Figure 1. Example of command decomposition using hierarchical process plans

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.