Int J Adv Manuf Technol (2001) 18:410–421
© 2001 Springer-Verlag London Limited

The International Journal of
Advanced Manufacturing Technology

A Schema for Flexible Equipment Control in Manufacturing Systems

J. C. E. Ferreira[1], J. Steele[2], R. A. Wysk[2] and D. A. Pasi[3]

[1]Universidade Federal de Santa Catarina, Departamento de Engenharia Mecanica, GRUCON, Florianopolis, SC, Brazil, [2]Pennsylvania State University, Department of Industrial and Manufacturing Engineering, PA, USA, [3]JLG Industries, McConnellsburg, PA, USA

In this work a methodology is proposed for increasing the flexibility of the control software of Flexible Manufacturing Systems (FMSs). This greater flexibility is required due to factors such as uncertain product demand, uneven distribution of shop load, and machine or cutting tool unavailability. In the proposed framework the following modules were developed: (a) an automated process planning module which generates non-linear process plans for a given part, considering the shop floor resource availability; the non-linear plans include both material handling and material processing information; (b) a planning module that linearises the process plan aiming at minimising the total manufacturing time of the parts; (c) a NC program generation module, which generates the NC program for the chosen CNC machine(s). In order to increase the flexibility of the control software even more, a resource model was devised and implemented, which provides the necessary resource information for the above modules. Each of these modules is described within this paper, and details about the part and process plan representation necessary for this implementation are also given. A case study is presented in order to show the capability of the methodology.

Keywords: FMS control; Manufacturing features; NC program generation; Process planning; Resources

1. Introduction

One of the characteristics of today's market is the uncertain product demand, and because of that manufacturing industries need to have flexibility in operations in order to keep the production running in the case of a new product order. One possible solution to that problem is the use of Flexible Manufacturing Systems (FMSs), which are composed of CNC machine tools and material storage systems interconnected by material transfer and handling equipment. FMSs aim at manufacturing low to medium-sized batches of different parts with almost no downtime between orders.

However, the FMS control system software is usually hard-coded into the FMS, making it difficult to cater for changes in the FMS environment, such as the inclusion of a new machine, or a modification of the layout. This is a problem that reduces the flexibility of such systems, and may lead to longer downtimes and lower throughput. Thus it is important to develop methodologies for the implementation of reusable control software for FMS [1,2].

In order to provide a higher flexibility for FMS control software, the authors proposed and implemented a framework which is described within this paper. The modules that comprise this framework are shown in Fig. 1. The activities that comprise the schema are described in the following sections.

2. The Part AND/OR Graph

Previous work has been done aiming at representing both parts and processes. With regard to parts, IGES (Initial Graphics Exchange Specification) is a standard that aims at transferring

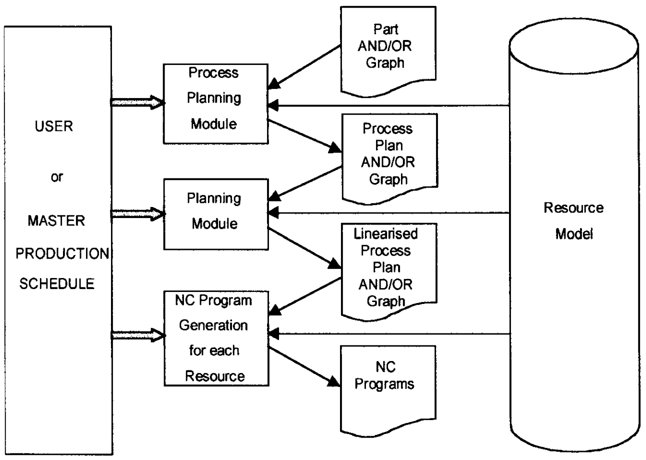


Fig. 1. The modules in the schema for flexible control.

Correspondence and offprint requests to: Dr. Joao C. E. Ferreira
Phone: +(55)(48) 331–9387; Fax: +(55)(48) 234–1519; E-mail: jcarlos @emc.ufsc.br

part information between different CAD systems, and STEP (Standard for Exchange of Product Model Data) [3] aims at representing and exchanging product information during its life cycle across the enterprise. Concerning processes, QTC (Quick Turnaround Cell) [4], ALPS (A Language for Process Specification) [5], and PSL (Process Specification Language) [6] have contributed towards the use of standards for process representation.

A non-linear type of representation is utilised in this work in order to provide more flexibility to the controller. In the case of a linear (traditional) process plan, disruptions caused by the non-availability of resources (e.g. cutting tools) often create problems with the planned operation sequence. In this case, the determination of replacement operations is traditionally done by a person by means of improvisation, and under pressure. This usually results in longer throughput times and higher costs. If the process plan already contained alternative plans, which is the case of a non-linear process plan, then a different feasible operation would already be available, and the execution of the plan would be much simpler if a disruption occurred.

There are different ways to represent process plans with alternatives, and some of these ways include:

- Petri-nets [7,8]
- AND/OR graphs [9]
- Tree structures [10]
- Directed graphs [5,11]

A detailed analysis of existing process representations is given in [12].

AND/OR graphs [13,14] are used in this work to represent both parts and process plans. These graphs are composed of nodes connected by arcs, and in the case of parts the nodes represent a manufacturing feature (e.g. a hole), whereas arcs represent AND/OR precedence, or a simple sequential precedence. A description of the symbols present in a part AND/OR graph is given in Table 1.

An example of a part and its AND/OR graph representation is given in Fig. 2. This AND/OR graph shows that all features in the part may be machined in any order, except for "hole5",

Table 1. Symbols present in a part AND/OR graph.

Name	Description	Symbol
Feature Node	A manufacturing feature	*Feature ID* / *Feature name*
Arc	Denotes precedence relation between nodes	$\longrightarrow$
AND Junction	All of the paths between the AND junctions will have to be executed	& b & e
OR Junction	Only one of the paths between the OR junctions will be executed, where "b" refers to "begin", and "e" refers to "end"	! b ! e

which has to be machined after "pocket1". The required information for each of these features, which includes their geometry and tolerances, is shown in Table 2. These attributes are described in [15].

The AND/OR graphs are stored in ASCII files, and the part information in these files is represented as shown in Fig. 3, having the part illustrated in Fig. 2 as an example. Such a product model can be generated from a CAD model using the algorithm presented by Lee and Kim [16].

The first portion of the AND/OR part file contains the necessary information about each feature (i.e. geometric data, tolerances and surface finish), whereas the "feature string" contains the graph structure, which corresponds to how the features are connected to one another. A nested postfix scheme is used to represent the graph structure [17], and this representation is defined by the following grammar:

T (terminal symbols) $\rightarrow$ *node_id*, "&", "!", "*"
N (non-terminal symbols) $\rightarrow$ *(S T)*
S (symbols) $\rightarrow$ *P* | *P S*
P (production set) $\rightarrow$ *node_id* | *N*

The semantics associated with each postfix symbol are as follows (see table 1):

(a) (*S &*): The ampersand (i.e. "&") implies AND relationship.
(b) (*S !*): The symbol "!" implies OR relationship.
(c) (*S* *): The symbol "*" implies that the elements of *S* are sequentially ordered. For example, if *N* is represented by (N_1 N_2 N_3 *), then N_1 precedes N_2, and N_2 precedes N_3.

3. The Process Plan AND/OR Graph

The process plan AND/OR graph contains alternative operations that may be performed on the workpiece. This graph also defines the sequence requirements for each specific plan. Alternatives allow the planning module to utilise current resource information when making decisions.

The same kind of representation presented for parts is used, but process information is stored instead of features. The nodes represent the tasks of part processing, material transport, material handling, or material storage. Each node contains detailed operation parameters, such as the cutting conditions and tool type for a machining operation. The symbols that are found in process plan AND/OR graphs are similar to those in part AND/OR graphs, and they are shown in Table 3.

4. The Process Planning Module

Process planning has been identified as a critical link between product design and production control. Thus, the results from process planning may be used to identify manufacturing constraints for the product design in addition to providing the plans for production simulation or control. Process planning research for discrete part manufacturing has focused on feature extraction from CAD product models [18] and encoding stan-

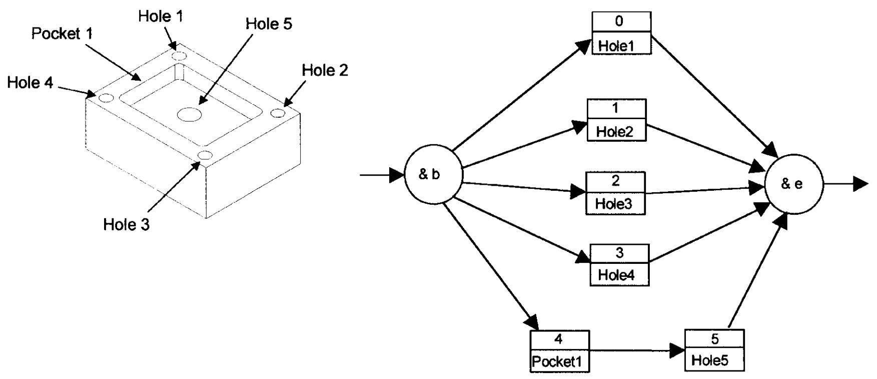


Fig. 2. A prismatic part and its representation through an AND/OR graph.

Table 2. Feature attributes necessary for process planning.

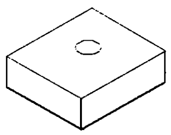
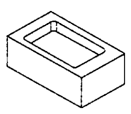
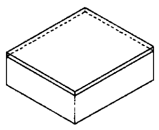
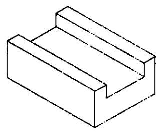
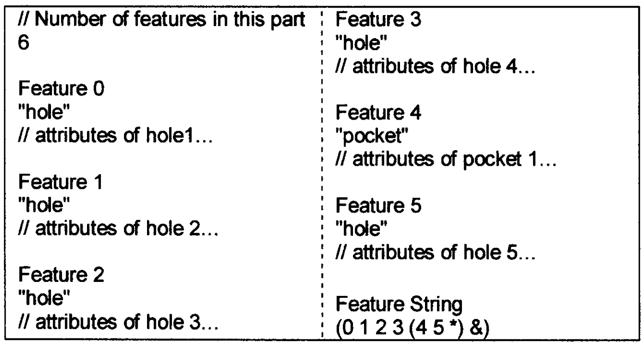
	Features			
	Hole	Pocket	Face	Slot
				
A T T R I B U T E	● feature type = "hole" ● depth ● diameter ● position ● position tolerance ● pos. diametric tolerance ● neg. diametric tolerance ● surface finish of hole	● feature type = "pocket" ● start position ● max. surface finish ● length ● pos. length tolerance ● neg. length tolerance ● width ● pos. width tolerance ● neg. width tolerance ● depth ● pos. depth tolerance ● neg. depth tolerance ● angle 1 (between the long side and the "x" axis) ● angle 2 (between the wide side and the "y" axis)	● feature type = "face" ● start position ● max. surface finish ● length ● pos. length tolerance ● neg. length tolerance ● width ● pos. width tolerance ● neg. width tolerance ● depth ● pos. depth tolerance ● neg. depth tolerance ● angle 1 (between the long side and the "x" axis) ● angle 2 (between the wide side and the "y" axis)	● feature type = "slot" ● start position ● max. surface finish ● length ● pos. length tolerance ● neg. length tolerance ● width ● pos. width tolerance ● neg. width tolerance ● depth ● pos. depth tolerance ● neg. depth tolerance ● angle 1 (between the wide side and the "x" axis) ● angle 2 (between the wide side and the "y" axis) ● end position

Table 3. Symbols present in a process plan AND/OR graph.

Name	Description	Symbol
Process Node	A manufacturing task	*Node ID* *Process name*
Arc	Denotes precedence relation between nodes	$\longrightarrow$
AND and OR Junctions	The same as in table 1	& b & e ! b ! e



```
// Number of features in this part
6

Feature 0
"hole"
// attributes of hole1...

Feature 1
"hole"
// attributes of hole 2...

Feature 2
"hole"
// attributes of hole 3...

Feature 3
"hole"
// attributes of hole 4...

Feature 4
"pocket"
// attributes of pocket 1...

Feature 5
"hole"
// attributes of hole 5...

Feature String
(0 1 2 3 (4 5 *) &)
```

Fig. 3. The contents of the file that corresponds to the part AND/OR graph.

dard machining practice rules into expert systems [19]. These rules map the features corresponding to generalised manufacturing operations to the usage of specific shop resources and manufacturing parameters such as cutting conditions. Problems with this approach are as follows:

(a) Material handling plans are not generated simultaneously with material processing. This may result in process plans that are impossible to execute or are scheduled inefficiently without consideration of prerequisite material handling tasks [20].
(b) The rules in expert systems do not explicitly link to representations of resources. For instance, a rule may refer to a machine tool using one name while the resource model uses another name. Thus, the process plan generation may not consider existing resources or may specify resources incorrectly.
(c) Standard practice manufacturing rules are reasonable in a high production flow shop but more flexible reasoning is desirable in flexible manufacturing to take advantage of the available tooling. Such flexible reasoning may be used to generate process plans with alternatives.

A process planning module was developed in order to address these problems, and it is grounded in procedural reasoning contained within behaviours of specific manufacturing resource objects. The information framework of this process planning methodology is illustrated in Fig. 4. These classes take advantage of object-oriented principles such as inheritance and ownership to efficiently represent the physical capabilities of manufacturing resources. Furthermore, these behaviours at different abstraction levels could be provided by different vendors and shared if resource standards such as those proposed by Jurrens et al. [21] were adopted. Instantiation of these classes is accomplished using a resource model described by Steele et al. [22]. Thus, the shop capability to produce parts is easily modelled using this approach.

Another important motivation for the implementation of this resource modelling approach is that there is a need to model the physical capability of resources to change the state of the part for flexible manufacturing where the products are not predefined. The state descriptions used to model these physical resource capabilities include the location of the part on the shop floor and features added to the raw material of the part.

Given this resource modelling capability, process plans can be generated, that quickly adapt existing resources in a shop to produce unanticipated products. This is accomplished by presenting a finished product model (i.e. the part AND/OR graph) as a goal state to a state-based search algorithm that uses operators from class behaviours of shop floor resource objects. These operators eliminate the differences between the state descriptions of the part. Our search algorithm uses a backward chaining approach to minimise the differences between an initial state description of the raw material part located in a raw material buffer and the goal state description of the finished part located in a finished parts buffer. In contrast to traditional search techniques that stop when one path to the goal is reached, this search algorithm explores the entire search space in order to find all alternative paths from the initial part state to the final state. This search space is limited by use of a problem reduction technique, which will be explained later in this paper. When the search space is entirely explored, another algorithm is used to trace through the resulting search graph in order to produce a process plan graph with alternative paths separated by OR branches.

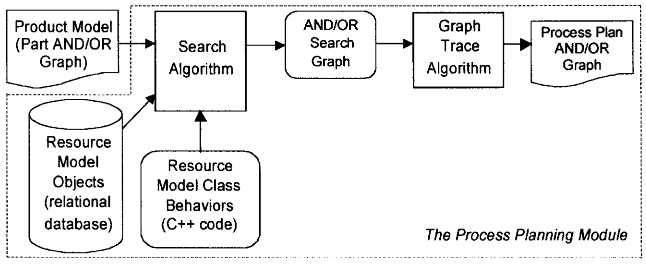

Product Model (Part AND/OR Graph)
Search Algorithm
AND/OR Search Graph
Graph Trace Algorithm
Process Plan AND/OR Graph
Resource Model Objects (relational database)
Resource Model Class Behaviors (C++ code)
The Process Planning Module

Fig. 4. Process Planning Data Flow.

4.1 The Resource Model

Wysk et al. [9] classify the assets of manufacturing systems using symbolic definitions from set theory. According to this classification, manufacturing assets include members of the equipment set (E), tool set (T), fixture set (F), and instruction set (I) that belong to the general set $R = \{E \cup T \cup F \cup I\}$. We group these assets according to an object-oriented and hierarchical shop floor model. According to Smith [23], these assets may be grouped hierarchically using the following concepts: *shop*, *work centre*, and *equipment*. Thus, in an object-oriented classification, resource classes include a facility class R that may own instances of the work centre class W. Each work centre class may own instances of the equipment class E, tool class T, fixture class F, and instruction class I. Objects in these classes that cannot be grouped in a work centre are directly owned by the facility object. Subclasses of class E include the classes MP, MH, MT, and BS. The MP class represents material processors in the facility, the MH class represents material handlers, the MT class represents material transporters, and the BS class represents buffer storage devices. These classes and their subclasses contain attributes designed to facilitate the activities illustrated in Fig. 4.

The material processors are the pieces of equipment that add value to a product by adding features to a raw material part. Material removal machines (machine tools) represent instances of this class. Different types of material processors are classified into subclasses of the MP class. For instance, milling machines represent one of these sub-classes, while turning centres represent another sub-class. Tools available for usage by material processors are modelled as tool objects that are owned by material processor objects.

The material handlers are equipment that load/unload products to/from equipment for processing, storage, or transportation while material transporters move products between stations. Typically, material handlers have the kinematic flexibility to change the orientation and position of the part to insert the part properly into equipment fixturing. Material handlers are typically industrial robots and their placement in the shop floor layout determines which buffers, machines and ports they may

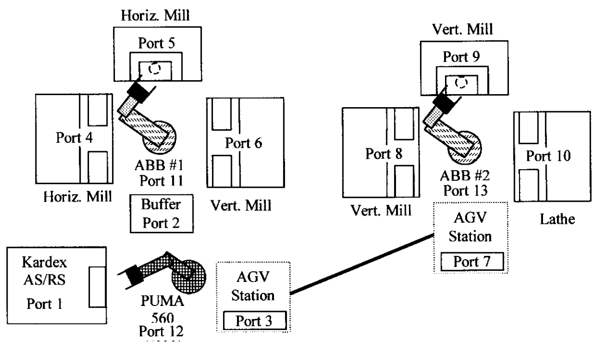


Fig. 5. Layout of the Computer Integrated Manufacturing Laboratory (CIM Lab).

access. Material transporters, on the other hand, are equipment that usually lack manipulation and simply move products from one location to another. Examples of material transporters are automated guided vehicles (AGVs) and conveyors.

Possible part movement within the facility with the use of these entities is specified with ports. A port represents a unique internal buffer belonging to a machine that owns one or more physical locations for part storage. Such machines are typically material processors or buffers. A port may also correspond to a home position of a MH entity. Thus, the capability of each MH object to load and unload parts to a home port is defined by its list of loadable and unloadable ports. The capability of each MT object to move parts is defined by its list of reachable ports. The collection of these lists defines the material handling connectivity graph of the facility [24]. Lastly, instances of the BS class either passively hold parts with a possible mechanical inversion or actively store the parts in an automated storage and retrieval system (AS/RS).

Figure 5 illustrates the layout of the Pennsylvania State University's CIM lab with instances of these different resource classes. This facility has six instances of MP that include horizontal and vertical CNC machining centres, and CNC lathes, each with their unique port number. Three MH devices are included in this facility, and their loadable and unloadable ports are shown in Table 4. Two BS devices are also included in this facility, which correspond to an AS/RS with port #1 and a static buffer with port #2.

The tool assets in a manufacturing facility are instances of the T class of effectors. These effectors are used to enable material processors to perform operations on parts. This class includes processing machine tooling such as twist drills and reamers. Note that there is explicit ownership and grouping in this definition because only specific classes of tools function with different material processors. Thus, the subclass for MP that describes lathe machines only may own instances of lathe tool classes. The data for the tooling attributes in the resource model was provided by United States Cutting Tool Institute [25] and Walsh [26].

Table 4. Home, loadable and unloadable ports corresponding to each of the MH devices in the manufacturing system illustrated in Fig. 5.

MH	Home Port	Loadable Ports	Unloadable Ports
ABB#1 robot	#11	2,4,5,6	2,4,5,6
ABB#2 robot	#13	7,8,9,10	7,8,9,10
PUMA robot	#12	1,2,3	1,2,3

In this architecture, the resource class object data is stored in a relational database. This takes advantage of the human interfaces provided by commercial relational databases. While commercial object-oriented databases are currently available and provide simple methods for directly storing object data [27], they do not provide good human readable interfaces due to the complexity of object relationships such as ownership and inheritance. The class methods that implement different layers of engineering functions are embedded in object-oriented software. This software creates instances of resource classes by reading the relational database tables and converting the tabular data into object data. This enables humans to monitor directly and adjust the data records for the resources in a manufacturing environment. Naturally, the structure of the database is determined by the standard resource classification so that the object-oriented software can easily be designed to read the object-data from the database records. Similarly, the object-oriented software may also modify data in the database such as the status of resources.

4.2 The Problem Reduction Technique

The problem reduction algorithmic framework, developed by Nilsson [28], proposes to solve problems automatically by reducing them to sub-problems that are easier to solve. This framework is well suited for discrete part manufacturing process planning. Using the state space description, the solution to problems consists of finding the appropriate operators to change the state from the initial to the goal state. In the discrete part manufacturing domain, we identify part features as sub-goals and search for operators from the resource model that can change the part state into the appropriate sub-goal state. In general, a problem description consists of:

{S,F,G}, where S = the current state, F = set of operators, and G = the goal state

The typical state space search solution consists of searching through a tree by applying the operators wherever possible until the correct sequence of operations is found that transforms the state S into state G. However, this search process becomes simpler by decomposing the problem into sub-problems. The sub-problems in turn become decomposed into sub-sub-problems until they are solved by one operation. This problem decomposition is accomplished by providing procedures in the operators that not only determine the operator's effect on the goal state but also to determine the precondition for using the operator to change the goal state in this way.

To illustrate this backward chaining problem reduction, Fig. 6 shows the decomposition of a problem given by {S,F,G}. From the list of operators given by F, *f1* and *f2* are known to eliminate some portion of the difference between the current state S and the goal state G. Thus, the search graph expands into two independent branches separated by an OR node (symbol "!"). One possible solution presented by *f1* is the left

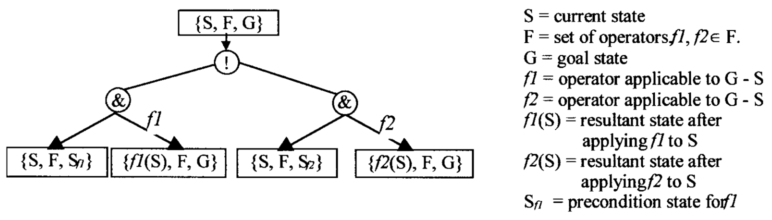

{S, F, G}
!
&
&
f1
f2
{S, F, Sf1}
{f1(S), F, G}
{S, F, Sf2}
{f2(S), F, G}
S = current state
F = set of operators f1, f2 ∈ F.
G = goal state
f1 = operator applicable to G - S
f2 = operator applicable to G - S
f1(S) = resultant state after applying f1 to S
f2(S) = resultant state after applying f2 to S
Sf1 = precondition state for f1

Fig. 6. Illustration of search graph generation.

branch. This branch is further decomposed into two sub-problems. The left sub-problem is represented by $\{S,F,S_{f1}\}$, which corresponds to eliminating the difference between the current state, S, and the state that is a precondition to using operator *f1*, S_{f1}. The sub-problem on the right-hand side is represented by {*f1*(S), F, G}, which consists of eliminating the difference between the state that results as a consequence of applying *f1* to S, and the final goal state, G. These sub-problems are considered solved when the current state of the sub-problem is equal to the goal state.

Applying this problem reduction technique to the discrete part manufacturing domain, we define the part state to be composed of the geometric model of the raw material, a non-ordered set of features added to the raw material, the current port where the part is located, and the orientation of the part at that port. The starting state of a part is the geometric model of its raw material, an empty set of features, and the port corresponding to the raw materials buffer storage entity. The goal state of the part is obtained from the part model and consists of its raw material geometric model, a set of goal features represented in the AND/OR graph, and the port corresponding to the finished materials buffer storage entity. Thus, the part state S consists of the following set:

S = {M,FEAT,P,O}
where M = geometric model, FEAT = {Feat1,Feat2, Feat3,. . .}, P = port, O = orientation

The process planner uses operators drawn only from specific resources in the resource model of the shop. Currently, the process planner uses operators defined in behaviours in MP resource objects and their tooling objects to add missing goal features to the part state. Operators are also defined in behaviours in MH and MT resource objects that move the part closer to the goal port or change the orientation of the part in a buffer. While fixturing operators fit in well within this process planning framework, they are currently not implemented and are reserved for future research.

The basic steps towards generating the process plan AND/OR graph consist of the following:

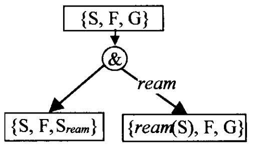
- The planning algorithm expands each unexplored problem node in the graph {S,F,G} by checking for features in G that are not found in S.
- For each feature not present in S, the algorithm determines how many different ways each MP resource in the facility may create that feature.
- Each MP resource queries methods in the T entities (tools) that are defined to be usable by the MP resource through the ownership relationship.

As an example, consider a goal feature being a hole specified by its position on a surface of the part, its depth, positional tolerance, and diametric tolerance. The raw and finished material buffer for the part is associated with port #1. A reamer T entity that is owned by a vertical milling MP entity with port #9 has a method that returns a solution that matches the hole's specifications. This solution creates an OR branch to the problem, and this is illustrated in Fig. 7. In general, if a solution to a portion of the goal is found, and precondition and resultant states are computed. Sub-problems that are generated due to this solution include the left sub-problem where the precondition feature must be created (i.e. a 0.94″ diameter hole), and the part must be located at port #9 with orientation O_1 (i.e. orientation of the part to machine feature 1). A MP operator representing a drilling operation can solve this precondition feature problem if a twist drill with a 0.94″ diameter is accessible to a MP entity in the resource model. The right sub-problem specifies that the part's port must be changed from the milling machine's port #9 to the finished part buffer port #1. Tracing through this sub-graph depth-first, going from the left AND node to right AND node, the resulting process sub-plan becomes the following sequence:

- move part from port #1 to machine with a twist drill;
- twist drill part to 0.94″ diameter;
- move part to machine with reamer at port #9;
- ream part to 1.0″ diameter;
- move part from port #9 to port #1.

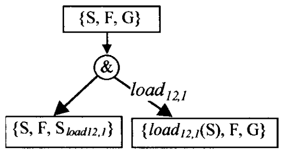
Given a sub-problem in the search graph, if the difference between the current and goal states does not include any features, the planner then attempts to solve the difference between the current and goal locations using MH or MT operators from the resource model. This is accomplished using backward planning from the goal location to the current location. MH operators fall into two categories: load and unload. The MH load operator for a given MH resource may only satisfy the goal of loading the part from the MH resource's home port to a given port if the port is in the MH resource's loadable ports list. The MH unload operator may satisfy the goal of moving the part to the MH resource's home port only if there is a port in the MH resource's unloadable ports list.

As an example, consider the sub-problem created by the *ream* operation situation shown in the right-hand side of the graph in Fig. 7. Since no features need to be solved for this problem, the part movement from port #9 to port #1 needs to be solved. From the resource model, a MH load operator can be found that corresponds to a MH resource entity with home port #12, and having port #1 is in its loadable port list.



S = {M, {}, 1, O}
F = set of operators, *ream* ∈ F.
G = {M, {1.0 DIA hole}, 1, O_I}
ream = operator applicable to G – S = 1.0 DIA hole
ream(S) = resultant state (after applying *ream* to S) = {M, {1.0 DIA hole}, 9, O_I}
S_{ream} = precondition state for *ream* = {M, {0.94 DIA hole}, 9, O_I}

Fig. 7. Search graph generation for a reaming operation.



S = {M, {1.0 DIA hole}, 9, O_I}
F = set of operators, *load* ∈ F.
G = {M, {1.0 DIA hole}, 1, O}
$load_{12,1}$ = operator applicable to G – S = port #1
$load_{12,1}$ (S) = resultant state (after applying $load_{12,1}$ to S) = {M, {1.0 DIA hole}, 1, O}
$S_{load12,1}$ = precondition state for $load_{12,1}$ = {M, {1.0 DIA hole}, 12, O}

Fig. 8. Search graph generation for MH load from home port operator.

Figure 8 illustrates the first step towards solving this problem, which consists of decomposing the problem into the left sub-problem of moving the part from port #9 with orientation O_I to port #12 and the solved right sub-problem. The remaining steps consist of decomposing the left sub-problem into sub-sub-problems using other MH unload and load operations in addition to MT operations.

A MT move operator corresponding to a MT resource entity to solve a sub-problem is applied only if the following conditions are satisfied: (a) the features are solved, and (b) the goal port is in the MT resource's reachable port list. For example, assume that the problem in figure 8 has been further decomposed into the following sub-problem

{S = {M,{1.0 DIA hole}, 9, O_I}, F,G = {M,{1.0 DIA hole}, 3,O}}

which results after applying a MH $load_{3,12}$ operator to the problem in Fig. 8. Figure 9 illustrates how a *move* operator decomposes the sub-problem corresponding to a MT resource with a reachability list composed of ports #3 and #7.

In addition to moving parts from a machine to a buffer or vice-versa, MH resources can also be used to change the orientation of a part within a machine. For instance, a part may have two features on different surfaces that can be machined on the same MP resource. After machining one feature, the part must be unloaded and then re-loaded into the machine in a different orientation. The precondition of this operator is that the part is located at the MH resource entity's home port. The generated sequence is a MH unload from the machine to the MH entity's home position and then a MH load to the machine with the correct orientation.

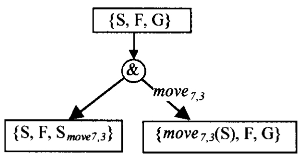
Figure 10 illustrates a sub-graph of such a scenario starting with a part with two features located on different surfaces. The two features are A and B, and the correct orientations for creating these features on the vertical milling machine with port #9 are O_A and O_B, respectively. Tracing through this graph depth first from the left AND node (providing the solution to the precondition of the right node) to the right AND node, the resulting process plan is as follows:

- move part from port #1 (AS/RS) to port #9 (Vertical Milling Machine) with orientation O_A;
- twist drill feature A;
- unload part from port #9;
- load part to port #9 with orientation O_B;
- twist drill feature B;
- move part from port #9 to port #1.

This process plan illustrates how the assumptions of causality in the operators' preconditions are correctly followed in the generation of the process plan. For instance, the part is brought to the milling machine at port #9 before the drilling operation is executed at that machine. Furthermore, the choice of precedence of solving the features first, then the difference between current and goal ports is solved, and then solving the difference in part orientation last is verified with this example.

4.3 Process Plan Graph Generation

The process plan graph is generated from the AND/OR search graph as outlined in Fig. 4. This procedure consists of tracing through the search graph depth first and replacing AND children nodes with a simple sequence of nodes starting with the left node and ending with the right node. Only nodes representing operations are copied into the process plan graph. OR nodes in the search graph translate directly into OR nodes in the process plan graph. On the other hand, OR nodes with branches that represent every single possible sequence of a set of operations are converted into AND nodes with this set



S = {M, {1.0 DIA hole}, 9, O_I}
F = set of operators, $move_{7,3}$ ∈ F.
G = {M, {1.0 DIA hole}, 3, O}
$move_{7,3}$ = operator applicable to G – S = port #3
$move_{7,3}$ (S) = resultant state (after applying $move_{7,3}$ to S) = {M, {1.0 DIA hole}, 3, O}
$S_{move7,3}$ = precondition state for $move_{7,3}$ = {M, {1.0 DIA hole}, 7, O}

Fig. 9. Search graph generation for a MT move operator.

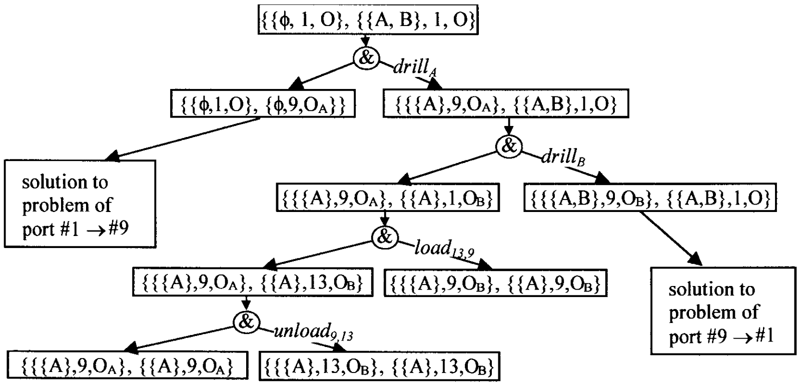

{{φ, 1, O}, {{A, B}, 1, O}}
drill_A
{{φ,1,O}, {φ,9,O_A}}
{{{A},9,O_A}, {{A,B},1,O}
solution to problem of port #1 →#9
drill_B
{{{A},9,O_A}, {{A},1,O_B}
{{{A,B},9,O_B}, {{A,B},1,O}
load_13,9
{{{A},9,O_A}, {{A},13,O_B}
{{{A},9,O_B}, {{A},9,O_B}
solution to problem of port #9 →#1
unload_9,13
{{{A},9,O_A}, {{A},9,O_A}
{{{A},13,O_B}, {{A},13,O_B}

Fig. 10. Search graph generation for changing orientation of the part in a machine.

of operations using the "AND" Graph Constructor algorithm developed by Nettala [29].

The AND/OR plan remains stored until there is an order for that part (or batch), that leads to this AND/OR plan to be retrieved. Then the planning module, which is described in the next section, linearises that plan, in order to enable its execution at the chosen machine(s).

5. The Planning Module

A planning procedure was implemented aiming at linearising the process plan AND/OR graph. This procedure verifies each OR node in the process plan graph, and chooses the best path stemming from each OR_BEGIN (i.e. "! b") node according to a certain objective function.

The objective function consists of choosing a path with the shortest total manufacturing time. The manufacturing time function for each node in the graph ("T^n_m") is given by:

$$T^n_m = t_{tsu} + t_m + t_{tc} \qquad (1)$$

where:

T^n_m = manufacturing time for node "n"

t_{tsu} = set-up time for each cutting tool specified in the node, which may include for example its installation into a machine carrousel

t_m = total machining time with the selected cutting tool

t_{tc} = automatic tool change time

The manufacturing time is calculated for each node in the graph, and in order to calculate it the planning module needs information on the tool that was selected by the process planning module. The resource model provides that information, which includes the tool diameter, its maximum cutting depth, the recommended speeds and feeds, etc. The tool set-up time depends on the location of the tool, which may be at the machine tool carrousel, tool crib or tool room. The tool location is also included in the resource model, and it may affect significantly the manufacturing time.

The algorithm presented below is used to select among alternative tasks in the process plan. In this algorithm, alternative paths in the graph are evaluated and the path with the shortest total manufacturing time is chosen. The partial manufacturing time corresponds to the sum of the manufacturing times of all nodes in a path between an OR_BEGIN and an OR_END nodes. The total manufacturing time for a part is calculated by summing up all the selected partial manufacturing times for each OR node.

```
For each OR node in the graph {
  Minimum Partial Manufacturing Time = large number
  For each alternative M stemming from the OR node {
    Determine the Partial Manufacturing Time for alternative M
    If Partial Manufacturing Time < Minimum Partial Manufacturing Time then {
      Minimum Partial Manufacturing Time = Partial Manufacturing Time
      Alternative = M
    }
  }
Mark selected alternative
}
Total Manufacturing Time = Σ all Partial Manufacturing Times for each OR node
```

6. The NC Program Generation Module

The NC program generation module is responsible for creating a NC file which can be executed by the equipment. The process plan reaches this module only after it is linearised, i.e. the operations along with their sequence have been selected. The NC file generation module reads the process plan one node at a time, and generates the tool path and the corresponding NC code.

For each node, a portion of NC code is created and written to a file where it is assembled to other NC blocks from the previous nodes in the graph. In order to generate the NC file, specific geometric information must be contained within each node, and the tasks that will be performed by the equipment must be defined in detail.

6.1 Structure

In general, the NC program generation module creates a portion of NC code for each feature, which consists of the following tasks:

Task 1: Rapid to feature
Task 2: Tool change (if necessary)
Task 3: Machine feature
Task 4: Rapid to clearance plane
Task 5: Go to task 1 (if necessary)

Obstruction avoidance must be considered when creating the rapid travel motion from one location to another between two tasks. The tool must not collide with the fixturing device or the workpiece. To avoid collision, a clearance plane is used for all rapid travel movements between tasks. A clearance plane is parallel to the table of the machine and is slightly above the highest point of the workpiece. This plane is defined by the start point element in each node. Therefore, any movement on this plane will avoid collision with the workpiece and fixture. In this implementation the clearance plane is equal to 0.10 inches (2.5 mm).

If a tool change is necessary between tasks, it takes place while the tool resides in the clearance plane. Once the tool has been changed, a rapid travel is initiated to the feature plane, which is parallel to the clearance plane and is located where cutting feed is about to be applied to machine the feature. By using this technique, obstruction avoidance can be achieved.

6.2 NC Program Generation Algorithms

The implemented NC program generation procedures are described in detail in [15]. A 3-axis CNC machining centre was considered to machine the features considered in this implementation (see Table 2). For instance, the general routine for drilling a hole consists of the following steps:

1. The spindle rapid travels along the clearance plane until it is directly above the hole.
2. Once the motion is completed, a tool change occurs, if necessary.
3. The pre-defined tool offsets are prescribed.
4. The tool rapid travels to the feature that is set as 0.10 inches (2.5 mm) above the hole.
5. The hole is drilled at a specific feed, speed, and depth with the coolant on.
6. The tool retracts to the feature plane. After the motion is completed, the spindle and coolant are turned off.
7. The tool then moves to the clearance plane where it is in position to rapid travel to the next feature.

The NC code created by the NC program generation module for this feature would be the following:

```
G00 X <x> Y <y> H0 M6 <tool>
H <tool offset> Z <z + 0.10>
Z <z + 0.10>
G01 Z <z – depth> S <rpm> F <feed> M3 M8
G00 Z <z + 0.10>
Z <z + 0.10> M5
```

7 An Example

The proposed schema has been applied to the parts shown in Fig. 11. Each part has two features, and they have one identical hole. A human worker was also considered as a possible way of moving the part from/to the AS/RS to/from the buffer storage. The machine tools are assumed initially without tools in the carrousel. The generated process plan graph for the part in Fig. 11(a) is shown in Fig. 12, and the node numbers shown in the figure, which correspond to the operations, are described in Table 5.

After applying the planning module to the process plan AND/OR graph in Fig. 12, the resulting process plan is as follows:

- Unload part from AS/RS and load buffer storage with PUMA robot (the time to perform these operations is shorter compared to a human)
- Unload part from buffer storage and load vertical machining centre with ABB #1 robot
- Drill and ream 0.5″ diameter hole
- Drill 1.0″ diameter hole
- Unload part from vertical machining centre and load buffer storage with ABB #1 robot
- Unload part from buffer storage and load AS/RS with PUMA robot

The process plan graph for the part in Fig. 11(b) is similar to the one corresponding to the other part, where the hole with diameter 1.0″ is replaced by a pocket. The chosen process plan for this part is also similar, to the other part's process plan, in which the drilling and reaming operations applied to the 0.5″ diameter hole are replaced by a pocket-milling operation, using an end-mill of 1.0″ diameter. The same tools used to machine the 0.5″ diameter hole in the first part are used to machine the identical hole in the second part, and they were already set-up at the machine carrousel before the manufacture of the second part.

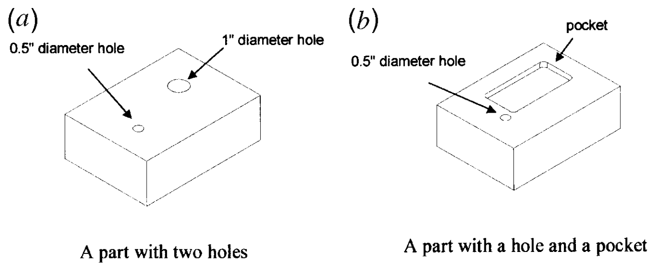


Fig. 11. The parts that were considered for manufacture.

Table 5. Description of the identification numbers of the nodes in Fig. 12.

Node IC	Resource	Operation	From Port → To Port /Coordinates (x,y) and diameter (∅)	Node ID	Resource	Operation	From Port → To Port /Coordinates (x,y) and diameter (∅)
1	PUMA robot	unload	1 → 12	14	Vertical M.C.	drilling	(1.5, 2.0); 0.4375″
2	PUMA robot	load	12 → 2	15	Vertical M.C.	boring	(1.5, 2.0); 0.48″
3	Human	move	1 → 2	16	Vertical M.C.	boring	(1.5, 2.0); 0.5″
4	ABB#1 robot	unload	2 → 11	17	Vertical M.C.	reaming	(1.5, 2.0); 0.5″
5	ABB#1 robot	load	11 → 6	18	Vertical M.C.	drilling	(5.0, 3.0); 0.9375″
6	Vertical M.C.	drilling	(5.0, 3.0); 0.9375″	19	Vertical M.C.	boring	(5.0, 3.0); 1.0″
7	Vertical M.C.	boring	(5.0, 3.0); 1.0″	20	Vertical M.C.	reaming	(5.0, 3.0); 1.0″
8	Vertical M.C.	reaming	(5.0, 3.0); 1.0″	21	Vertical M.C.	drilling	(5.0, 3.0); 1.0″
9	Vertical M.C.	drilling	(5.0, 3.0); 1.0″	22	ABB#1 robot	unload	6 → 11
10	Vertical M.C.	drilling	(1.5, 2.0); 0.4375″	23	ABB#1 robot	load	11 → 2
11	Vertical M.C.	boring	(1.5, 2.0); 0.48″	24	PUMA robot	unload	2 → 12
12	Vertical M.C.	boring	(1.5, 2.0); 0.5″	25	PUMA robot	load	12 → 1
13	Vertical M.C.	reaming	(1.5, 2.0); 0.5″	26	Human	move	2 → 1

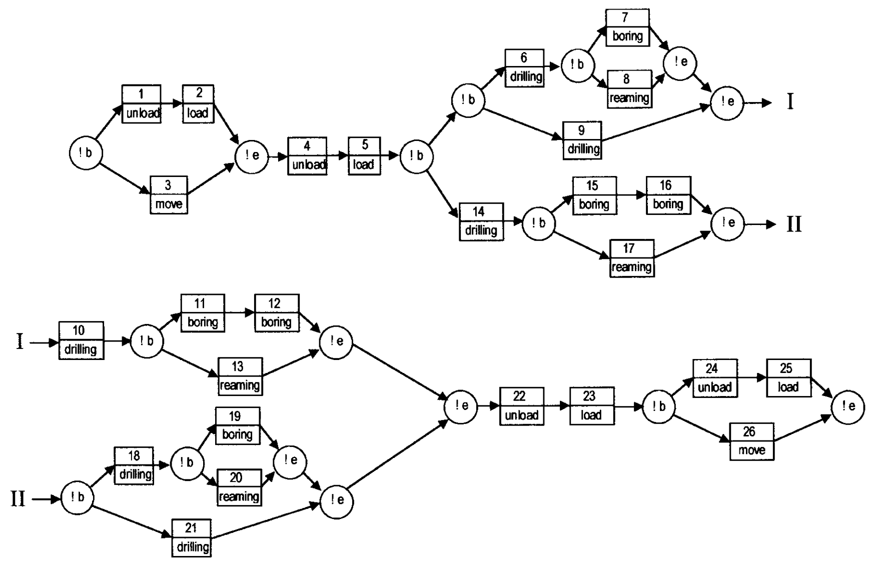
Table 6. The generated NC programs for the parts in Fig. 11.

Part with the two holes

```
%
N001 M64
N002 G80 G40 G17
N003 E2 G90 X0 Y0
N004 G00 X1.5000 Y2.0000 H0 M6 T1
N005 H1 Z1.1000
N006 Z1.1000
N007 G01 Z0.5000 S829 F9.9531 M3 M8
N008 G00 Z1.1000
N009 Z1.1000 M5
N010 G00 X1.5000 Y2.0000 H0 M6 T2
N011 H2 Z1.1000
N012 Z1.1000
N013 G01 Z0.5000 S458 F6.8755 M3 M8
N014 G00 Z1.1000
N015 Z1.1000 M5
N016 G00 X5.0000 Y3.0000 H0 M6 T3
N017 H3 Z1.1000
N018 Z1.1000
N019 G01 Z0.5000 S363 F7.9832 M3 M8
N020 G00 Z1.1000
N021 Z1.1000 M5
N022 H0
N023 G00 X0.0000
N024 G92 X0
N025 E0 X0 Y0
N026 M65
N027 M2
%
```

Part with the pocket and the hole

```
%
N001 M64
N002 G80 G40 G17
N003 E2 G90 X0 Y0
N004 G00 X1.5000 Y2.000 H0 M6 T1
N005 H1 Z2.1000
N006 Z2.1000
N007 G01 Z1.5000 S829 F9.9531 M3 M8
N008 G00 Z2.1000
N009 Z2.1000 M5
N010 G00 X1.5000 Y2.0000 H0 M6 T2
N011 H2 Z2.1000
N012 Z2.1000
N013 G01 Z1.5000 S458 F6.8755 M3 M8
N014 G00 Z2.1000
N015 Z2.1000 M5
N016 G00 X4.0000 Y3.1000 H0 M6 T4
N017 H4 Z2.1000
N018 Z2.1000
N019 G01 Z1.7500 S2723 F16.3408 M3 M8
N020 X5.5000 Y3.7000
N021 X2.5000 Y3.7000
N022 X2.5000 Y2.5000
N023 X5.5000 Y2.5000
N024 X5.5000 Y3.7000
N025 G00 X4.0000 Y3.1000 Z1.7500
N026 G01 Z1.5000 S2723 F16.3408 M3 M8
N027 X5.5000 Y3.7000
N028 X2.5000 Y3.7000
N029 X2.5000 Y2.5000
N030 X5.5000 Y2.5000
N031 X5.5000 Y3.7000
N032 G00 X4.0000 Y3.1000 Z1.5000
N033 G00 X4.0000 Y3.1000 Z2.1000 M5
N034 H0
N035 G00 X0.0000
N036 G92 X0
N037 E0 X0 Y0
N038 M65
N039 M2
%
```



Process plan string:

(((1 2 *) 3 !) 4 5 ((((6 (7 8 !) *) 9 !) 10 ((11 12 *) 13 !) *) (14 ((15 16 *) 17 !) ((18 (19 20 !) *) 21 !) *) !) 22 23 ((24 25 *) 26 !) *)

Fig. 12. The generated process plan graph for a prismatic part with two holes in non-parallel surfaces.

The generated NC codes for both parts in Fig. 11 are shown in Table 6.

8. Conclusions

In this paper a framework for increasing the flexibility of FMS control software was presented. Three software modules were developed as a fundamental portion of this framework, which consisted of a process planning module, a planning module and a NC program generation module. The representation of parts is based on features with alternatives (i.e. it is non-linear). The process plans are also non-linear, and they are represented as a group of operations that may take place in sequence, or as alternatives. Material handling and material processing operations were included in the process plans. For a certain process plan AND/OR graph, the graph is linearised by the planning module, and the NC program is generated for the linearised graph. A description is given of the resource model, which was implemented in order to help provide a greater flexibility of the manufacturing system control software.

The process plan AND/OR graph confers flexibility to shop floor control, since the controller will have many alternatives from which to decide what is the most adequate, according to the conditions of the shop floor, and also what is the best under a certain criterion.

Another advantage of the proposed methodology is that material handling and material processing operations are grouped in the same graph, and it enables the identification of a "good" process plan not only for machining, but also for material handling. Furthermore, in situations where material handling operations are necessary (e.g. to change the set-up of a part in order to machine a feature originating from a surface with a different orientation to a previous feature), they are easily included in the process plan graph.

The resource model plays a very important role in the proposed framework since: (a) it contains the necessary information about the resources available in the shop floor; (b) the specific information about a certain resource may influence significantly the process plan graph. For example, in order to machine a part with two holes in opposing surfaces in a 3-axis CNC machining centre, at least two set-ups will be necessary, and thus a material handling operation would have to be performed. In the case of a 4-axis CNC machining centre, the part could possibly be machined in only one set-up.

References

1. S. Lee, R. A. Wysk and J. S. Smith, "Process Planning Interface for a Shop Floor Control Architecture for Computer-Integrated Manufacturing", Int. Journal of Production Research, 33(9), pp. 2415–2435, 1994.
2. J. I. Llorente, M. I. Sarachaga, A. Burgos, J. Viñolas and R. Bueno,"Reuse of Control Software in Manufacturing Systems", Annals of the CIRP, 46(1), pp. 403–407, 1997.
3. ISO 10303–1, "STEP Part 1: Overview and Fundamental Principles", ISO TC184/SC4/WGPMAG Document N43, NIST, Gaithersburg, USA, 1992.
4. T. C. Chang, "Process Plan Representation Data Format – Version 1.0", QTC Document, Engineering Research Center for Intelligent Manufacturing Systems, Purdue University, USA, 1992.
5. B. A. Catron and S. Ray, "ALPS: A Language for Process Specification", International Journal of Computer Integrated Manufacturing, 4(2), pp. 105–113, 1991.
6. C. I. Schlenoff, A. Knutilla, and S. Ray, "Unified Process Specification Language: Requirements for Modeling Process", NISTIR 5910, National Institute of Standards and Technology, Gaithersburg, MD, 1996.
7. W. E. Wilhelm and H.-M. Shin, "Effectiveness of Alternate Operations in a Flexible Manufacturing System", International Journal of Production Research, 23(1), pp. 65–79, 1985.
8. J. P. Kruth and J. Detand, "A CAPP System for Nonlinear Process Plans", Annals of the CIRP, 41(1), pp. 489–492, 1992.

9. R. A. Wysk and J. S. Smith, "A Formal Functional Characterization of Shop Floor Control", Computers in Industrial Engineering, 28(3), pp. 631–643, 1995.
10. S. Shapiro, "Encyclopedia of AI", John Wiley & Sons, 1992.
11. G. M. Schneider and S. C. Bruell, "Concepts in Data Structures and Software Development", West Publishing Co., 1992.
12. A. Knutilla, C. Schlenoff, S. Ray, S. T. Polyak, A. Tate, S. C. Cheah and R. C. Anderson, "Process Specification Language: an Analysis of Existing Representations", National Institute of Standards and Technology, NISTIR 6160, May 1998.
13. L. S. H. De Mello and A. C. Sanderson, "AND/OR Graph Representation of Assembly Plans", IEEE Transactions on Robotics and Automation, 6(2), 1990.
14. E. G. Mettala and S. B. Joshi, "A Compact Representation of Process Plans for FMS Control Activities", Pennsylvania State University, IMSE Working Paper Series, 1990.
15. D. A. Pasi, "A Formal Structure and Implementation of an Equipment Level Controller in a Shop Floor Control System", M.Sc. Thesis, Pennsylvania State University, USA, May 1996.
16. J. Y. Lee and K. Kim, "Generating Alternative Interpretations of Machining Features", International Journal of Advanced Manufacturing Technology, 15, pp. 38–48, 1999.
17. J. F. Elder, "Compiler Construction: A Recursive Descent Model", Prentice-Hall, 1994.
18. S. B. Joshi and T. C. Chang, "Graph-based Heuristics for Recognition of Machined Features from a 3D Solid Model", Computer-Aided Design, 20(2), pp. 58–66, 1988.
19. D. Kiritsis, "A Review of Knowledge-Based Expert Systems for Process Planning, Methods, and Problems", International Journal of Advanced Manufacturing Systems, 1995.
20. J. S. Smith, B. Peters and A. Srinivasan, "Job Shop Scheduling Considering Material Handling", International Journal of Production Research, 1999.
21. K. Jurrens, J. Fowler and M. Algeo, "Modeling of Manufacturing Resource Information", NISTIR 5707, National Institute of Standards and Technology, 1995.
22. J. Steele, Y. Son and R. A. Wysk, "Resource Modeling for the Integration of the Manufacturing Enterprise", Proceedings of the Joint Japan-USA Symposium on Flexible Automation, Ann Arbor, Michigan, USA, July 23–26, 2000.
23. J. S. Smith, "A Formal Design and Development Methodology for Shop Floor Control in Computer Integrated Manufacturing", Ph.D. Thesis, Pennsylvania State University, 1992.
24. R. A. Wysk, B. A. Peters and J. S. Smith, "A Formal Process Planning Schema for Shop Floor Control", Engineering Design and Automation, 1(1), pp. 3–10, 1995.
25. United States Cutting Tool Institute, "Metal Cutting Tool Handbook", 7th Ed., Ind. Press, 1989.
26. R. Walsh, "Machining and Metalworking Handbook", McGraw-Hill, Inc., 1994.
27. M. Stonebraker, L. Rowe, B. Lindsay, J. Gray, M. Carey, P. Bernstein and D. Beech, "Third-Generation Database System Manifesto", Computer Standards and Interfaces 13, New Holland, pp. 41–54, 1991.
28. N. J. Nilsson, "Problem-solving Methods in Artificial Intelligence", McGraw-Hill, Inc, 1991.
29. E. Mettala, "Automatic Generation of Control Software in Computer-Integrated Manufacturing", Ph.D. Thesis, Penn State University, USA, 1989.