Process Reasoning

Steven R. Ray
Factory Automation Systems Division
National Bureau of Standards

ABSTRACT

With the increasing levels of automation in factories, production management systems need increasingly sophisticated models of manufacturing processes. A framework is presented for the design and implementation of process reasoning modules which characterize processes within a manufacturing environment. Each reasoning module describes a process in terms of its effects, constraints, and process parameters. The effects include intended objectives, such as material removal, and side effects such as burr production. Constraints include specific limits on process parameters, and preconditions to the invocation of a process. These preconditions can require the execution of several other processes, leading to the chaining of events. The process parameters provide qualitative and quantitative information necessary to execute a process.

To implement a reasoning module, a process is classified in terms of the underlying physics taking place. This immediately generates a number of effects, and possibly some necessary process parameters to be associated with the process. This approach, described in this paper, relies heavily on concepts of object-oriented programming and artificial intelligence techniques. If one of the effects is a change in geometry, an interface to a geometry modeling system will be inherited. Use of a modular, inheritance approach to construct these reasoning modules means that new processes may be quickly defined. The implications of a physical process are inherited, automatically leading to constraints, side effects, etc.

A process reasoning module can play many roles. It can be queried as an expert to determine the feasibility of achieving a given goal, expected side effects, and optimum process parameters. A module can also simulate a process, given specific parameter settings.

Published in **Computers in Industry**, 9, 329-335, Elsevier Science Publishers B.V. (North Holland), 1987.

1. INTRODUCTION

The characterization of processes is becoming increasingly important as manufacturing becomes more automated. A production management system must be able to predict the detailed behavior of a process, including any desirable or undesirable side effects. To do this, process models must be available to describe the different aspects of a process. These models include, for example, characterization of burr formation in metal cutting operations, chatter of cutting tools, heating effects, etc. But process models alone are not enough. They must be linked and organized in a logical way. Process reasoning modules are presented here as an approach to accomplishing this organization, and providing a flexible and powerful means of describing arbitrary manufacturing operations.

In many computer applications the difficulty of solving a problem is dependent upon the representation scheme used. In automated manufacturing the early determination of an appropriate representation is also important. For example, how should some metal cutting process be communicated throughout a factory? In the Automated Manufacturing Research Facility (AMRF) at the National Bureau of Standards [Simpson82, Hocken83, McLean85], this type of question led to the development of the concept of a work element [Ray86]. Work elements represent individual processing steps, which can be combined to create process plans. They 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 (perhaps a part blank), resulting in a goal state (a finished part). The communication of activities throughout the shop floor using work elements is accomplished by means of an ASCII based neutral format specification for process plans [Brown86].

As more intelligence is added to components of an automated manufacturing facility, the work element concept alone becomes insufficient. It must be embedded within some larger, intelligent entity that can predict the consequences of a given process. A process reasoning module can be viewed as the extension of the work element concept to such an intelligent module. Where a work element is a passive entity used to represent a process, a reasoning module is an active unit that can be questioned on the details of a process. Such a module has access to deterministic models, such as geometry engines and models for stress analysis, heat transfer, burr formation, etc. Interaction with a module takes place using message passing under one of the following categories: process selection, process validation, process testing, and process simulation. Each type of interaction is described in Section 4. A module can respond to queries on topics such as the feasibility of a process and possible side effects. It can also perform abstract reasoning, interact with other reasoning modules, and simulate the execution of a process. It is a specialist entity, which ideally can answer any question about the particular process it represents. The process could be a low-level one such as milling or drilling, or it could be a more abstract operation such as part delivery, or even lot production.

2. OVERVIEW

This paper describes a modular framework for process reasoning that can be used for all processes encountered in an automated environment such as the AMRF. Section 3 presents the functional design of a reasoning module and describes its basic features. Section 4 discusses a natural implementation approach, including the classification of modules. Section 5 presents some of the applications which could use the process reasoning concept. Section 6 is a summary of the paper.

3. DESIGN

The purpose of a process reasoning module is to characterize the role and behavior of a process. A module represents a process in terms of its effects, constraints, and process parameters. Each of these three attributes can be further broken down into more detailed categories. For the sake of explanation, this paper focuses on the domain of discrete part production. However, these concepts should be considered as applicable to the entire field of production management.

Effects

The effects of a process can be split into principal, or intended effects, and side effects. This division is primarily to clarify the purpose of a given process. This clarification can facilitate tasks such as process selection. Classification as a principal effect implies controllability, which is not necessarily true for a side effect. Examples of principal effects include material transformation, which can be further classified into material removal, pure geometric change (without material removal), property change (e.g. annealing), and joining or assembly. Each of these operations alters the intrinsic description of a manufactured part. Another class of principal effects is transportation, which does not affect the part itself, but does change its location. Table 1 shows the categorization of process effects in more detail. Examples of side effects include chip production, tool wear, and can also include any of the principal effects mentioned above. What may be a desired effect in one operation may in fact be a side effect in another. For example, a principal effect of geometry change using a torch cutting operation may also have the side effect of material property change near the cut, due to heating. Two particularly important side effects are the cost and execution time for a process. These quantities are critical to the generation of optimized

Category	**Examples**
<u>Principal Effects</u>	
Material Transformation	
Material removal	
Mechanical	Milling, cutting, drilling
Electrical	Electric Discharge Machining
Thermal	Torch cutting
Chemical	Etching
Property change	
Chemical	Traditional chemical reactions
Electric	Doping in semiconductors
Material	Annealing, hardening
Joining / Assembly	
Mechanical	Traditional assembly
Thermal	Welding
Chemical	Bonding
Geometry change	Bending, forming, casting
Transportation	
Material transfer	Parts delivery
Information transfer	Database access
...	
<u>Side Effects</u>	
All of the Principal Effects	
Chip production	
Tool chatter, wear, breakage	
Energy consumption	
Cost	
Execution time	

Table 1.
Categories of Process Effects - Partial Listing

process plans and for scheduling. They are categorized as side effects, since neither the cost nor the passage of time is an intended effect.

It is the portion of the reasoning module dealing with effects which communicates with the library of deterministic process models that supports it. To determine the effects of a particular process instance, this portion of the module would call numerical models to predict tool chatter, perform heat transfer calculations, and update geometric models. This type of information is often called deep knowledge [e.g. Kline82], as compared to a purely heuristic description of a process. The reasoning module serves as the intelligent interface to all of these numerical models, choosing the appropriate routines and parameters as necessary.

Constraints

Another important aspect of a process that is represented in a reasoning module is its constraints. Major categories of constraints include preconditions, postconditions, requirements, parameter limits, and material properties. These are shown in Table 2. Preconditions describe the

Category	**Examples**
Preconditions	Other processes such as setup, fixturing, startup routines.
	Processes such as shutdown, cleanup.
Postconditions	
Requirements	Tools, robots...
Hardware	N/C code, database access.
	Machine limitations on speed, feed, travel.
	Hardness limits, electrical, chemical
Software	properties.

Table 2.
Categories of Process Constraints - Partial Listing

requisite state of the world prior to the execution of a given process. In general, this would include the execution of other processes. Similarly, postconditions include any necessary cleanup operations, reporting, and shutdown procedures. A requirement is any resource that is necessary to perform a process, including software and hardware. An example of a material property constraint is that of electrical conductivity of a part blank for an electrical discharge machining operation.

Process Parameters

The parameters portion of the module contains all the information necessary for the unambiguous execution of a particular process. This aspect bears the most resemblance to the work element concept mentioned earlier. It is a template for a process which, when used to specify an operation, is filled in with run-time parameters. These parameters would include speeds and feeds for milling operations, for example.

4. IMPLEMENTATION

The previous section described a process reasoning module in terms of its external characteristics. This section demonstrates a means of supporting this structure in a way that allows new modules to be quickly designed and implemented. The approach relies heavily on the concept of inheritance to provide most of the functionality of a reasoning module. Object-oriented programming techniques lend themselves naturally to the building of a network of reasoning modules that can share capabilities [Winston84]. Knowledge of object-oriented programming terminology is of benefit in reading the following sections of this paper.

Process Classification

The major step in the design of a new process reasoning module lies in the classification of the process in terms of its underlying physics. This can be accomplished by extending the philosophy of the DCLASS system for manufacturing processes [Allen80]. The DCLASS system categorizes processes in a tree taxonomy. The approach here is to extend this taxonomy to a graph covering a broader range of processes structured in terms of physical effects, such as chemical, transport, cutting, etc. Figure 1 shows a portion of this range of processes and also depicts the inheritance hierarchy of the modules. Each node in the graph represents a reasoning module, with the exception of the top node. Note that all of the modules inherit a common set of capabilities that includes a communications interface, data storage routines and other utilities. The designer would choose the appropriate class of process from the taxonomy by traversing the graph. As each node in the graph is traversed, the reasoning module under design inherits a number of behavioral characteristics. These can include interfaces to deterministic models, necessary parameters, constraints and effects. Once the traversal is complete, the designer is left with a rough version of the reasoning module that already contains most of the necessary functionality. It then remains to add additional constraints, effects and parameters as required for the particular process under consideration to complete the module design.

As an example of the design procedure, consider a twist-drilling process. This falls under the classification of a material removal operation. With this choice, there is an implication of a geometrical change taking place. Thus, an interface to a geometric modeler is added to a list of modules to inherit. Also, a part model is added to the list of requirements in the constraints section. Next, the process operates by mechanical removal. This implies that the material be clamped or fixtured in some fashion, so a precondition of part fixturing is also added as a constraint. As the last step in this example, the process is classified as a rotational cutting operation, as opposed to a

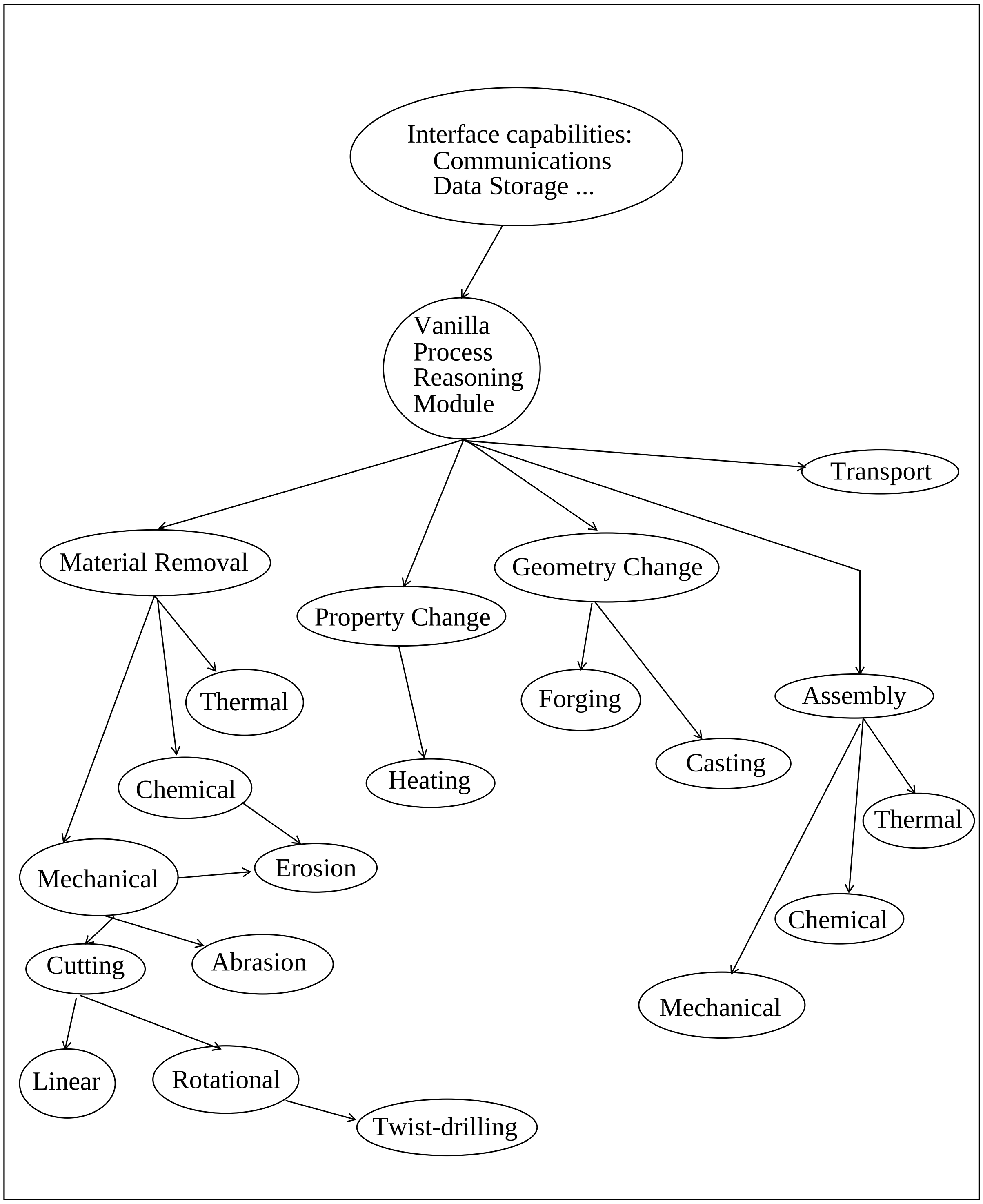

Figure 1.

Process Classification by Effect - Partial Graph linear cutting operation such as sawing. A rotational cutting tool is added as a requirement, and interfaces to tool wear and chatter models are inherited. Any inputs these models may require are added to the parameters section. When the additional attributes for the twist-drilling module have been specified, it is added as a node in the process taxonomy. Thus, its attributes become available for inheritance in future modules.

What has been described above is simply the specification of a frame system to maintain the inheritance structure of the process reasoning modules [Minsky75, Fikes85]. By taking advantage of this inheritance mechanism, the similarities within classes of processes are exploited, minimizing the duplication of software.

Interacting with a Reasoning Module

Communication with a reasoning module is by means of a small set of general requests using message passing to the module. At this point in the design, these requests are of the following types:

A. Process Selection

Given a description of an initial and final state, the module is requested to report on the feasibility of accomplishing the transition from one state to the other. The module reports the effects, constraints and parameters for the transition, or a failure message identifying which constraints were violated. Note that the constraints include identifying any other processes that must have already been completed.

B. Process Validation

Given a description of a process in terms of the parameters to be applied, the module is requested to report on the effects and any constraint violations. In this case a fully specified process, complete with all the necessary parameters, is being provided as input. In transactions of type A, the process specification was the output.

C. Process Testing

For computational efficiency, a module could be requested to report on a particular effect, constraint or parameter when operating on a request of type A or B discussed above. This could be used to answer questions such as "Will I have a problem with tool chatter if I operate under the following conditions?".

D. Process Simulation

A module can be requested to simulate its operation. This could be in the form of a report outlining the timing of all intermediate effects, overall execution time and the invocation of other process modules when appropriate. Alternatively, the module could be requested to run in real time, schedule the simulation of any supporting process modules, and signal the completion of its own simulation run.

A module can also be requested to perform any of the services described in the above paragraphs with additional constraints specified. This is particularly important during process plan optimization. The important point about these communication specifications is that they are identical for all modules.

Supervisory Control of Reasoning Modules

Each reasoning module is charged only with adequately representing the behavior of a specific process or class of processes. To use these modules in a coordinated way, there must be a supervisory program. This supervisor is reponsible for keeping track of all reasoning modules, generating the queries for the modules, and packaging the results for some external agent, such as a planning system. The software architecture of the supervisor has not yet been worked out, although a promising approach has been used in the expert process selection system described in the next section.

5. APPLICATIONS

Process reasoning modules can serve many needs, an important one being process planning. Planning can take place by querying a number of modules as to the feasibility of accomplishing a desired effect, such as the creation of a machineable feature in a part. This corresponds to a number of occurances of query type A. Each module can perform the necessary geometric modeling to establish its feasibility, and can consider its other constraints. These include the preconditions and postconditions, which can result in other modules being requested to consider their own execution. This chaining of modules ultimately results in a number of alternative sequences of processes being presented as candidates, each with a cost and time estimate. This approach is a generalization of an expert process selection system currently in use within the AMRF, called SIPS [Nau87]. The SIPS system operates by evaluating restrictions associated with classes of processes, coupled with a least-cost branch and bound search technique to identify an optimum sequence of processes to produce a given machining feature. The restrictions are analogous to the constraints identified in this paper. The concept of a reasoning module is consistent with the approach used by SIPS, but each node in the search graph is replaced by a module that has interfaces to numerical models and can keep track of the effects of its execution.

A second important application for reasoning modules is that of process simulation. Since the module has access to numerical models describing the underlying physics, a simulation can be carried out in detail, in order to discover unexpected effects, establish timing, etc. For example, conditions producing tool chatter can be identified for milling operations. By coupling the supervisor with a simulation controller, realistic and detailed information can be derived about collections of processes. This information can be particularly valuable for production management, since it allows realistic and detailed world models to be maintained, which can then be compared to the actual state of a system.

In general, the applications for a process reasoning module include any information requests that are normally handled by an engineer expert in a particular process. The modules are intended to be general purpose repositories of all kinds of knowledge about a particular operation, or class of operations. Note that each module must be individually requested to consider its application. There is still a need for a higher level program to choose a module as a candidate for accomplishing a task.

6. SUMMARY

A description has been provided of a general framework for the capture of detailed information about any process in an automated environment. This description has been in terms of the design and implementation of process reasoning modules. A process reasoning module is an intelligent entity with the capability to perform abstract reasoning about a process, and to invoke deterministic models to predict behavior when needed. Each module describes a process in terms of its effects, constraints and parameters. A frame representation provides a convenient implementation of the process reasoning module concept. A frame approach allows the use of inheritance to provide much of the functionality for a module. In this way, the creation and definition of new reasoning modules is made much easier. This paper does not address the question of the supervisory control of process reasoning modules, which will be covered in a future publication.

7. REFERENCES

[Allen80] Allen, D. K. and Smith, P. R., **Fabrication Process Taxonomy**, Monograph No. 5, Computer Aided Manufacturing Laboratory, Brigham Young University, Provo, Utah, January 1980.

[Brown86]	Brown, P.F. and McLean, C., **Interactive Process Planning in the AMRF**, Proceedings of Winter 1986 ASME Conference, Anaheim, California, 1986.
[Fikes85]	Fikes, R. and Kehler, T., **The Role of Frame-Based Representation in Reasoning**, Communications of the ACM, 904, September 1985.
[Hocken83]	Hocken, R. and Nanzetta, P., **Research in Automated Manufacturing at NBS**, *Journal of Manufacturing Engineering*, 91, #4, 68, (1983).
[Kline82]	Kline, W.A., DeVor, R.E. and Lindberg, J.R., **The Prediction of Cutting Forces in End Milling with Application to Cornering Cuts**, *Int. J. Mach. Tool Des. Res.*, 22, #1, 7, (1982).
[McLean85]	McLean, C., **An Architecture for Intelligent Manufacturing Control**, Proceedings of Summer 1985 ASME Conference, Boston, Massachusetts, August, 1985.
[Minsky75]	Minsky, M., **A Framework for Representing Knowledge**, *The Psychology of Computer Vision*, edited by Patrick Winston, McGraw-Hill, New York, New York, 1975.
[Nau87]	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.
[Ray86]	Ray, S.R., **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.
[Simpson82]	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).
[Winston84]	Winston, P.H., **Artificial Intelligence**, Addison-Wesley, Reading, Massachusetts, 1984.