Proceedings of the First SIGMAN Workshop on Manufacturing Planning

Detroit, Michigan

August 22, 1989

Editor:

Dana Nau, University of Maryland, College Park, MD

Program Committee:

Keith Hummel, Allied Signal Corporation Kansas City Division
Kate Lines-Browning, National Institute for Standards and Technology
Dana Nau, University of Maryland (*chair*)
Mary Pickett, General Motors Research Labs
Steve Ray, National Institute for Standards and Technology

The International Joint Conference on Artificial Intelligence

August 1989, Detroit, U.S.A.

Workshop 2: Manufacturing Planning

OBJECT-ORIENTED EXPERT SYSTEMS IN MANUFACTURING PLANNING

D.H. Norrie(*+), O.R. Fauvel(*), D.T.Guo (*), B.R. Gaines(+)

Department of Mechanical Engineering (*)
Department of Computer Science (+)
The University of Calgary, Calgary, Alberta, Canada T2N 1N4

EXTENDED ABSTRACT

An object-oriented manufacturing planning system is under development in which the expert-system inferencing capability and the associated facts and rules are distributed across components of the system. Further extension is possible to other domains and incorporating distributed processing. The primary entity of the system is a Rule-Based Object.

There are three broad categories of object in the system: those concerned with function only, those concerned with both function and property, and those concerned with property only. The first category implement procedures ('methods' in SMALLTALK). Objects in this category do not have a knowledge-base attached and are illustrated by specialized inferencing objects. The second category both implement procedures and include a knowledge-base and have as examples process planners and schedulers. In the third category, the objects are essentially knowledge-base storages with little or no functional capability, such as database objects.

All objects in the system, irrespective of their function, have essentially the same internal structure and belong to a particular type of object which the developers have named the Rule-Based Object (RBO). In general, within each object, the specific 'knowledge' for the object is separated into a *Facts* sub-object (containing the factual data about the object) and a *Rules* sub-object (containing the specific 'rules' (procedures) for the object's actions). Together, the facts and rules (data and procedures) comprise the knowledge base in the expert system sense. What corresponds to the inference engine code is elsewhere within the object (for a very specific kind of interpretation) or inherited from a higher level (for more general kinds of interpretation or inferencing).

Since the rule set may simply require consecutive execution, the 'inference engine' may only be a sequential line-by-line interpreter. In contrast, a process planning object may have a backward-chaining inference engine which makes use of the whole knowledge base of the object, i.e. both facts and rules. It will be seen that each object can have, in effect, its own expert system incorporated for 'deciding' how to act or react. The advantage is that not only can each object have its 'expert system' embedded, as simple or sophisticated as needed, but that this capability can be inherited in whole or part from a

higher class. To change the characteristics of a particular object, may require only an appropriate change to a single knowledge base.

The Rule-Based Object described above has three components: Methods, Facts, Rules. The Methods section of the object comprises the methods (procedures) which carry out the object's functions and the Facts and Rules sections together form the knowledge-base of the object. Although it is conceptually convenient to consider the Facts and Rules for a Rule-Based Object as being within the object, for implementation it is desirable to separate these sections out into a database and simply provide references to their location, in the object itself.

The following is the operation sequence for an object which receives a request (message) to accomplish some task, which for example, might be to carry out a process planning function or to search for and return specified information. The incoming message triggers the corresponding method (procedure) which, say, calls a particular inference engine to execute the object's own rule-set, beginning with the first rule. Part way through the rule-set, the inferencer encounters another rule directing a transfer to another rule-set (in object of another class) for a certain sequence and to then return. Further down the rule-set, a similar branching requires the use of rules in an object higher or lower in the hierarchy of the same class. Yet further down the rule-set, a supervisor is appealed to and (say) 'decides' that a different inference engine should now be used with some other rule-set(s) to obtain needed information, following which the original sequence resumes. In principle, any object can be given methods to request action(s) of any other objects, to call inferencing engines or other more general 'reasoning' objects, to reference facts or rules, and to have any appropriate sequence carried out. The value of object hierarchies is evident since the more general methods, facts and rules can be specified at the higher levels and these used in association with the more specific procedures and data be given at lower levels of the hierarchy.

In the early stages of implementation, it is proposed that the [IF---THEN---] production rule syntax be used, for simplicity. For later stages, consideration is already being given to extending the syntax to beyond the simple selection control structure.

As noted earlier, specific inferencing capabilities such as forward chaining and backward chaining would be assigned to objects designed for this purpose. More general 'reasoning' objects could be also created to handle such functions as arithmetic, fuzzy logic, boolean logic, matching, indexing and planning. The rule syntax would need to be appropriately extended and the rules or data-sets or objects on which these operations were to be carried out specified.

A top-down strategy can be implemented most simply if there is a structural hierarchy incorporating successive levels of modularity. For a Smalltalk implementation, the whole consortium would be a complete Smalltalk system. The hardware interface would use Smalltalk primitives to communicate both to the computer hardware and through ports to other processors and factory floor machines.

The initial focus of the project is Manufacturing Planning but it is recognized that the industrial world ultimately requires other functions, within an overall Information System. The next generation of manufacturing information systems will desirably include at least the following general functions: Management and Financial; Design; Manufacturing Planning; Equipment Monitoring and Supervision. For this reason, particular concern is being given in this project to considerations such as structure, modularity, extendability,

inheritance, distributed processing, and both software and hardware interfacing.

Based on the above architectural structure, an Object-Oriented Information Management Planning System (OBJIMPS) is under development in Smalltalk-80 on Sun workstations. The first modules are concerned with process planning, cell grouping and simulation functions. Initially, the system is based on a single CPU; subsequent extension is plannned to distributed processors using ConcurrentSmalltalk. While Smalltalk is well suited for development, it is recognized that other object-oriented languages such as C++ can offer advantages with porting which may be necessary at a later date.

An automatic intelligent interface system is also under development for use with OBJIMPS and other object-oriented environments [1]. Further details on the Object-Oriented Information Management Planning System can be found in [2-4].

ACKNOWLEDGEMENTS.

Appreciation is expressed to the Natural Science and Engineering Research Council of Canada for financial support. (Research Grants #A4192 and #A4948)

REFERENCES.

1. Guo, D.T., Norrie, D.H., and Fauvel, O.R., *ARIS: An Automatic Representation Interface System for Object-Oriented Manufacturing Systems.* IJCAI-89 Workshop: "A New Generation of Intelligent Interfaces", 11th International Joint Conference on Artificial Intelligence, Detroit, 20-25 August 1989.

2. Norrie, D.H., Fauvel, O.R., and Gaines, B.R., *Object-oriented Planning, Scheduling, and Simulation in Advanced Manufacturing Systems.* Report No. 421, Department of Mechanical Engineering, University of Calgary, Alberta, Canada; February 1989

3. Norrie, D.H., Fauvel, O.R., and Gaines, B.R., *Object-oriented Management Planning Systems for Advanced Manufacturing.* Working Notes, Symposium on AI and Manufacturing, American Association of Artificial Intelligence, Stanford University, 28-30 March 1989.

4. Norrie, D.H., Fauvel, O.R., Gaines, B.R., and Guo, D.T., *An Object-oriented Distributed Interfacing System for Advanced Manufacturing.* Report No. 425, Department of Mechanical Engineering, University of Calgary, Alberta, Canada; March 1989. 1989