The representation of feature-based component data model in a knowledge-based process planning system

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Master's Thesis. Submitted in partial fulfilment of the requirements for the award of Master of Philosophy at Loughborough University.

Metadata Record: [https://dspace.lboro.ac.uk/2134/27969](https://dspace.lboro.ac.uk/2134/27969)

Publisher: © Xuan-Xuan Huang

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 2.5 Generic (CC BY-NC-ND 2.5) licence. Full details of this licence are available at: [http://creativecommons.org/licenses/by-nc-nd/2.5/](http://creativecommons.org/licenses/by-nc-nd/2.5/)

Please cite the published version.
This item was submitted to Loughborough University as an MPhil thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

![Creative Commons License](https://creativecommons.org/licenses/by-nc-nd/2.5/)

**Attribution-NonCommercial-NoDerivs 2.5**

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

- **Attribution.** You must attribute the work in the manner specified by the author or licensor.

- **Noncommercial.** You may not use this work for commercial purposes.

- **No Derivative Works.** You may not alter, transform, or build upon this work.

- For any reuse or redistribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the [Legal Code](http://creativecommons.org/licenses/by-nc-nd/2.5/).

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
<table>
<thead>
<tr>
<th>VOL. NO.</th>
<th>CLASS MARK</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 JUN 1995</td>
<td>6 OCT 1995</td>
</tr>
<tr>
<td>30 JUN 1995</td>
<td>28 JUN 1996</td>
</tr>
<tr>
<td>25 MAY 1995</td>
<td>2 JUN 1996</td>
</tr>
<tr>
<td>21 JUN 1995</td>
<td>15 JAN 1998</td>
</tr>
</tbody>
</table>

Loan copy date due for return: 10 NOV 1998
No renewal
THE REPRESENTATION OF FEATURE–BASED COMPONENT DATA MODEL IN A KNOWLEDGE–BASED PROCESS PLANNING SYSTEM

by

Xuan–Xuan Huang

A Master’s Thesis
Submitted in partial fulfilment of the requirement
for the award of

Master of Philosophy of the Loughborough University of Technology

April 1993

© by Xuan–Xuan Huang 1993
DECLARATION

No part of the work described in this thesis has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning.
Dedicated to my parents and my husband
ACKNOWLEDGEMENTS

The author wishes to thank:

Dr. N.N.Z.Gindy for his kind supervision, encouragement and critical reading of the draft manuscript;
Dr. G F Modenlen for his helpful advice as my research director;
Dr. J.X.Gao for his understanding, supports and close cooperation;
Mr. D. Walters and R. Doyle for their excellent service to keep the computer system working;
Mrs. J Stevensen for her kind reading of the draft manuscript and correction of spelling mistakes;
Dr. T. Ratchev for his friendly cooperation;
ACME for its financial support for the research project.
PREFACE

This thesis contains six chapters. The first chapter generally introduces the basic concepts of computer aided process planning and the objectives of this project. The work of other researchers is also reviewed. Chapter 2 introduces the functional structure of the prototype generative process planning system and the Knowledge-Based System (KBS) "Generis", used as the main software tool for system development. Chapter 3 describes in detail the part data model and the process capability model implemented in the KBS system. The feature representation scheme, data structure of the part data model and the process capability model are introduced. In Chapter 4, the planning logic and the structure and the rulesets of the prototype planning system are described in more detail. Chapter 5 introduces the feature-based design system which has been integrated with the prototype planning system. An example is given to show the integration mechanism and the input and output of the integrated systems. The conclusions drawn from the research and suggestion for the future work are included in Chapter 6. Finally, the computer programs developed and technical papers published during the project are included as appendixes.
ABSTRACT

Computer-Aided Process Planning (CAPP) is a vital bridge between Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM). The planning domain involves vast amounts of knowledge, experience and data. The structuring and representing of such information, which includes part information and machining processes information, are some of the key issues that have to be addressed during development of a system for process planning.

This research has concentrated on representing the component and machine tool information in a knowledge based process planning system and integrating the planning system with a feature-based computer-aided design system. The system implemented consists of a user interface to the planning system, a pre-processor which handles the output information from the feature-based design system, and a knowledge-based planning system for prismatic machined components.

In this work the knowledge-based software system "GENERIS" has been used as the main software tool for developing a prototype planning system. The system is based on matching the geometric requirements, together with part of the technological requirements, of components to the capabilities of the machining operations. This information is stored in the form of relational tables in GENERIS. Each table contains several slots specifying the attributes of its subject. An inference engine controls the automatic execution of the planning procedures, the system developed is capable of analysing the attributes of component features based upon a component data model, attaching appropriate machining methods and selecting feasible machining directions. Machine tool selection and minimising component set-ups on each machine tool are also performed by the system.

The feature-based component data model has played an important role in integrating the feature-based design and process planning systems used in this work. Adopting a knowledge based approach has proved to be a useful and powerful methodology for use in the development of process planning systems.
# Declaration

# Acknowledgements

# Preface

# Abstract

**Chapter 1: Introduction and Literature Survey**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 The concept of Computer-Aided Process Planning (CAPP)</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Approaches to Computer-Aided Process Planning</td>
<td>3</td>
</tr>
<tr>
<td>1.2.1 The variant approach</td>
<td>3</td>
</tr>
<tr>
<td>1.2.2 The generative approach</td>
<td>4</td>
</tr>
<tr>
<td>1.2.3 The automatic process planning approach</td>
<td>5</td>
</tr>
<tr>
<td>1.3 The Goals of CAPP Research</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Feature definition</td>
<td>6</td>
</tr>
<tr>
<td>1.5 Feature taxonomy</td>
<td>7</td>
</tr>
<tr>
<td>1.6 The methods for part data representation</td>
<td>7</td>
</tr>
<tr>
<td>1.7 Approaches to extracting feature data from CAD systems</td>
<td>8</td>
</tr>
<tr>
<td>1.8 AI techniques in process planning</td>
<td>9</td>
</tr>
<tr>
<td>1.8.1 The structure of process planning knowledge</td>
<td>10</td>
</tr>
<tr>
<td>1.8.2 Some examples of expert process planning systems</td>
<td>10</td>
</tr>
<tr>
<td>1.9 CAD/CAM data exchange standards</td>
<td>12</td>
</tr>
<tr>
<td>1.10 The objectives and scope of this research project</td>
<td>13</td>
</tr>
</tbody>
</table>

**Chapter 2: The Knowledge-Based Process Planning System - GENPLAN**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Introduction to Generis</td>
<td>15</td>
</tr>
<tr>
<td>2.1.1 Knowledge in Generis</td>
<td>15</td>
</tr>
<tr>
<td>2.1.2 The main features of Generis</td>
<td>17</td>
</tr>
<tr>
<td>2.2 The conceptual model of the prototype system</td>
<td>18</td>
</tr>
<tr>
<td>2.3 The structure of the process planning system</td>
<td>18</td>
</tr>
</tbody>
</table>

**Chapter 3: The Part and Process Capability Models**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Introduction</td>
<td>23</td>
</tr>
<tr>
<td>3.2 The Feature Representation Scheme for the Part Data Model</td>
<td>23</td>
</tr>
<tr>
<td>3.2.1 Feature Taxonomy</td>
<td>24</td>
</tr>
<tr>
<td>3.2.2 Feature Relationships</td>
<td>26</td>
</tr>
<tr>
<td>3.2.3 The Content of Feature Information</td>
<td>29</td>
</tr>
<tr>
<td>3.2.4 The content of the data about parts</td>
<td>30</td>
</tr>
<tr>
<td>3.2.5 Representing part data in the Knowledge-Based System</td>
<td>31</td>
</tr>
<tr>
<td>3.3 The Process Capability Model</td>
<td>32</td>
</tr>
<tr>
<td>3.3.1 Representing the capability of machine tools</td>
<td>33</td>
</tr>
<tr>
<td>3.3.2 Representing processes and operations with features</td>
<td>38</td>
</tr>
<tr>
<td>3.3.3 Representing process capability model in the Knowledge-Base System</td>
<td>40</td>
</tr>
</tbody>
</table>

**Chapter 4: The Planning Logic and the Rulesets**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 The general principle</td>
<td>46</td>
</tr>
<tr>
<td>4.2 The selection of machining processes and cutting tools</td>
<td>46</td>
</tr>
<tr>
<td>4.3 The Optimisation Criteria for Set-up Determination</td>
<td>47</td>
</tr>
<tr>
<td>4.4 The manufacturing rules in the rulebase</td>
<td>48</td>
</tr>
<tr>
<td>4.4.1 Some Examples of the Geometric Reasoning Rules</td>
<td>50</td>
</tr>
<tr>
<td>4.5 The Planning Flowchart</td>
<td>52</td>
</tr>
<tr>
<td>4.6 An example</td>
<td>55</td>
</tr>
</tbody>
</table>
Chapter 5: The Integration of the Planning System with a Feature-Based Design System

5.1 Introduction to the feature-based design system
5.2 The basic tasks of the system integration
5.3 An Example

Chapter 6: Discussions, Conclusions and Further Work

6.1 Discussions
6.1.1 Structuring of knowledge in the planning domain
6.1.2 Feature-based component representation
6.1.3 Representation of processing system information
6.1.4 Planning data reference model
6.1.5 System implementation
6.2 Conclusions
6.3 Further Work

References

Appendix I: The Table Structure of the Part Data Model
Appendix II: The Machining Process and Machine Tool Capability Data
Appendix III: The Table Structure of the Process Capability Model
Appendix IV: The Inference Rules in the Rulebase
Appendix V: The Procedure Programs in the Inference Engine to Execute the Planning Logic
Appendix VI: The Feature Processing Program
Appendix VII: The Part Data Output from the Feature-Based Design System
Appendix VIII: The Reformatted Part Data Readable by Generis
Appendix IX: The Published Papers Associated With This Research Project
CHAPTER ONE

Introduction and Literature Survey
Chapter 1: Introduction and Literature Survey

In manufacturing industry, the process planning tasks are still predominantly performed by human planners. Planners reason about the geometry and technological requirements of parts based on engineering drawings produced by design departments. Planners use their knowledge about the capabilities of available manufacturing systems and their planning experiences to determine the feasible processes to machine the required parts.

One of the main disadvantages of the above method is that it depends heavily on the experiences and knowledge of different process planners, and therefore it is hard to assess the different planned processes made by different planners even for the same parts and in the same manufacturing environment. Another disadvantage is that it is time consuming, since the planners may have to search for various technical information about the manufacturing system such as specifications and capabilities of machine tools, tooling, fixtures, etc. from all sorts of documentations. Sometimes human mistakes are inevitable.

With the introduction of computers in engineering, it is possible to solve the above problems to a great extent by storing the information about manufacturing systems in the computer memory and by programming the planning logic into computer codes so that the planning logic can access the data about the manufacturing systems and produces consistent process plans automatically.

1.1 The concept of Computer-Aided Process Planning (CAPP)

Process planning is defined by the society of manufacturing engineering as "the systematic determination of the methods by which a product is to be manufactured economically and competitively" (Alting and Zhang 1989).

In any production environment, specific planning functions take place at different stages, and the global sum of these activities constitutes a complete manufacturing planning system. There are many technical and non-technical (e.g. economic and social) factors that must be taken into consideration in achieving an effective manufacturing plan. On the technical side, manufacturing planning functions can be characterized as production planning, process planning, and operation planning (Ham and Lu 1988).

The tasks of process planning include the following two basic stages: process design and operation design. Process design is a macro-scope decision procedure to make an overall process routine for converting the raw material into a product; and operation design is a microscope design procedure to select and sequence individual operations contained in the process routine. The operation design is concerned with the detailed decisions of production implementation, that is,
the types of operations to be performed in the production process (the content of each operation and the method of performing it). In summary, the major activities involved in process planning are as follows:

- selection of machining operations
- sequencing of machining operations
- selection of cutting tools
- selection of machine tools
- determining set-up requirements
- calculation of cutting parameters
- toolpath planning and generation of NC part programs
- design of jigs and fixtures

Since process planning provides the link between component design and production, Computer-Aided Process Planning (CAPP) is the key activity in engineering production that links Computer-Aided Design and Manufacture (CAD/CAM) and thus plays an important role in achieving Computer Integrated Manufacture (CIM).

1.2 Approaches to Computer-Aided Process Planning

There are three basic approaches to computer-aided process planning, i.e. the variant approach, the generative approach and the automatic process planning approach (Chang 1990). Variant process planning systems are most widely used in industry today. Generative process planning systems are regarded as more advanced than variant process planning systems. In theory, generative process planning systems are turn-key systems with all the decision logic built in. Since this is still far from being realized, current generative systems provide a wide range of capabilities and can at best be described as semi-generative. The automatic process planning approach is still under development in research centres.

1.2.1 The variant approach

The variant approach is based upon the concept of Group Technology (GT). Parts are grouped into families and standard plans are produced manually by planners and stored for each part fam-
ily. Planning for new parts involves identifying similar plans, retrieving plans, and editing of the plans to suit the requirements of new parts.

Group Technology is a manufacturing concept in which similar parts are grouped together in part groups or families (Arm 1975). Parts may be similar in two ways: the design characteristics and the manufacturing processes required to produce them. By grouping similar parts into families, planners can improve their efficiency. Such improvements are the result of advantages gained in such areas as set-up time, standardization of processes, and scheduling. Group technology is an important CAD/CAM concept in that it is a bridge builder between the design and manufacturing components.

Variant systems are the first generation of computer-assistant process planning systems. Examples of such systems are CAPP of CAM-1 (Link 1976) and MIPLAN (Houtzeel 1980). The advantage of such systems is that it is fairly straightforward to generate process plans for parts when classified into pre-defined families. The main disadvantage is that some parts which do not fall into the pre-defined families can not be planned. And also individual parts in the same family may require different processes and therefore the standard plans must be modified manually. Another disadvantage is that the geometry of features in the parts is not considered. This makes automatic NC (Numerical Control) code generation impossible.

1.2.2. The generative approach

A generative process planning system synthesizes a new plan for each part by using decision logic, algorithms, and geometry-based data. Typically each generative system has a manufacturing database and a decision logic (as shown in figure 1-1). The manufacturing database contains part descriptions (geometric and technological information about parts) and process capability data. The planning logic acts as the human process planners in the conventional process planning systems.

A clear advantage of the generative process planning system over the variant system is that every individual part can be planned and since the planning logic of most such systems use feature-based information about parts, detailed plans can be made at feature level (compared with the plans at part family level made by the variant systems) and NC codes may be generated. Other advantages of the generative process planning approach are that it generates consistent process plans rapidly and also has potential for integrating with an automated manufacturing facility to provide detailed control information. Examples of generative process planning systems are APPAS (Wysk 1977), CMPP (Waldman 1983), EXCAPP (Daveis and Darbyshire 1984) and XPLAN (Lenau and Alting 1986).
The problem with this approach is that it is difficult to develop such a system that is general enough to handle a large range of complex components. The decision logic is rigidly built into the program code. For new applications, the code may have to be re-written. This can be improved by using Knowledge-based systems. Moreover, the planning systems still need manual input of part data. This certainly needs improvement since powerful CAD tools are already available and widely used in design offices.

1.2.3 The automatic process planning approach

The third approach is called automatic process planning. It denotes process planning that can generate a complete process plan directly from an engineering design model (CAD data). There is no human assistance required in decision making. This approach has been taken by Choi, et al (1984), Kanumury, et al (1988) etc.

The automatic process planning approach poses two major issues, i.e. an automatic integration mechanism between the planning systems and CAD systems has to be developed and the planning logic should also be intelligent enough to handle problems without human assistance in decision making.

The main difficulty with the integration with CAD systems is that most current CAD systems are geometry-based. Features cannot be represented. Although some CAD systems can be
tailored to be feature-based (Case and Gao 1993), they are still not so satisfactory in representing features.

On the other hand, to make the planning system more intelligent, Artificial Intelligence (AI) tools and Knowledge-Based Systems (KBS) have been introduced.

1.3 The Goals of CAPP Research

As discussed above, automatic process planning is the approach to be achieved. In other words, the ultimate goals of the current research in computer aided process planning are that computer-based systems must be developed to automate individual planning functions and to integrate those functions into a unified environment, i.e. functional automation and system integration.

One of the first challenges facing system integration is the definition of features in a way that they can serve both the design and the manufacturing functions. A problem is that in many cases both functions cannot be covered by one definition. The design of a global related data structure as a framework for identification of geometric shapes in relation to design and manufacturing functions is therefore of extreme importance (Ham & Lu 1988).

Another challenge in system integration is that part features should be automatically extracted from the product model without human interaction. But existing interfaces to CAD systems do not sufficiently consider this requirement of automated process planning. The designer generating the drawings is often not aware of the constraints and limited resources that manufacturing engineering has to deal with when carrying out these intentions. This may be due either to a lack of communication with planners, or to a lack of experience with production, or a lack of information about the factory facility.

For the functional automation of the process planning systems, optimized planning logic should be developed, probably with the aid of Artificial Intelligence tools and Knowledge-Based Systems (KBS).

1.4 Feature definition

As discussed above, feature representation is the vital task in the integration of process planning systems and computer-aided design systems. Features retain a high level of abstraction of a part's description such as the part shapes and technological data such as tolerance and surface finish. This coupling between shape and technological data implies that there is a strong link between features and manufacturing processes.
A broad definition of features is given by Pratt and Wilson (1985) as: 'A feature is a region of interest on the surface of a part'. In the manufacturing process, features are treated as 'manufacturing features'. In the design process, a feature is treated as a 'design feature', with its geometry, topology and specifications to satisfy certain functional requirements. When the geometry of a feature is being considered, it is usually called a 'form feature' or a 'geometric feature' (Jared 1986). A feature is defined by Gindy as 'a volume enveloped by a set of real faces and imaginary faces' (Gindy 1989). Real faces are the enveloping faces that exist on the component. Imaginary faces are the enveloping faces that do not exist on the component.

1.5 Feature taxonomy

Features are usually classified into different classes so that the members in the same class can share the same characteristics and attributes. This is useful in both design and process planning. In design, this helps to select feature primitives and to retrieve the feature data. In process planning, it helps to select manufacturing strategies by using the feature classification characteristics such as number of access directions, geometry, depression or protrusion, etc. A feature class may have a number of sub-classes, and so on which form a hierarchy.

The way of classifying features depends on the way the features are used in different applications. Butterfield et al (1985) classified form features into three main categories: sheet features, prismatic features and rotational features. Each of the classes are further classified into sub-classes, etc. Gandhi (1989) classified features using a parametric approach, i.e. form features which have the same topology are grouped together so that they can be represented using the same number of parameters. Gindy's scheme (Gindy 1989) is mainly based on the number of External Access Directions (EADs). Form features are divided into three categories i.e. protrusion, depression and surface. Each of the super-classes are further classified into nine basic classes, i.e. boss, pocket, hole, non-through slot, through slot, notch, step, real face and imaginary face, etc.

1.6 The methods for part data representation

In process planning, parts are represented as a stock material (block) plus a set of features. In design and manufacturing engineering, there are a number of ways to represent part data. Examples are two-dimensional (2D) engineering drawings, Group Technology (GT) codes, symbolic representations and geometric representations (wireframe, surface and solid models), etc. (Chang 1990).

2D drawings can not represent 3D real objects completely, human assistance is often needed to reason about the geometry of features. Sometimes ambiguity is inevitable. GT coding technique is used to group parts with similarity (Henderson 1986). It does not represent details about
individual features. Symbolic representations define parts using descriptive languages. This is widely used in Knowledge-Based process planning systems (Chung et al 1988, Shah et al 1988). The disadvantage of this method is that the procedure for defining features is time-consuming and not interactive. Graphical verification is not available. Solid models describe 3D objects completely in terms of geometry and topology. Graphical tools are fully used in this method which allow interactive design and editing of complex parts. The problem with this method in supporting part data for process planning is that parts are not represented in terms of features. Functional and technological information is also missing in the design data models.

One of the tasks of this research project was to combine the advantages of the Knowledge-Based Systems and geometric modelling systems in the process planning methodology.

1.7 Approaches to extracting feature data from CAD systems

An integrated CAD/CAPP system requires automatic extraction of feature data from CAD systems. There are typically two basic approaches to obtaining feature data automatically from CAD systems, i.e. feature recognition and design by features.

Process planning systems using a feature recognition approach extract feature data from geometric models already created by CAD systems through a feature recognizer (Gavankar et al 1990, Choi et al 1984). Designers are allowed to use conventional geometric modellers (such as solid modellers, surface modellers and wireframe modellers) to construct part geometry without worrying about feature representations. When the design is finished, the feature recognizer matches the design data with pre-defined patterns or rule templates so as to determine part features (Joshi and Chang 1990, Brooks et al 1987). The advantage of this method is that the designers are free to design complex parts using the fairly mature geometric modelling tools. The disadvantage of this approach is that when the geometry of parts is complex, the feature recognizer may become very complicated and in some circumstances there may be some features which can not be identified. The problems here are that it may not be possible to recognise all the features because of feature interactions and that the model includes the design intent.

Design by feature systems allows designers to define components by creating instances of pre-defined feature primitives. Therefore, feature data is generated once the design is finished (Case and Acar 1989, Dixons 1988, Gao et al 1992, Shah and Rogers 1990). Feature recognition is not required. The benefit of using this approach is that feature data can be obtained immediately after design from the CAD systems without writing a complicated feature recognizer. And design with features is also the natural way for designers (designers also think that a part is raw material plus a set of functional features).
1.8 AI techniques in process planning

As process planning becomes integrated with product design and manufacturing into one intelligent system, tasks will become more and more complex. Present research and development efforts in process planning are directed mainly at generative systems, using knowledge-based techniques from AI research to handle the complex problems associated with knowledge representation for decision making (Gupta 1990).

A process planning system developed using AI techniques is called an expert process planning system. An expert system is "A computer program using expert knowledge to attain high levels of performance in a narrow problem area." (Waterman 1986). Generally an expert system consists of three major parts: declarative knowledge about the problem, procedural knowledge about the problem solving method, and a control system or the inference engine.

The advantages in the use of an expert system approach for CAPP are:

(a) the generation of a process plan requires consideration of a number of factors which influence selection and sequencing decisions for processes, and their important parameters. This problem is difficult for the traditional programming languages to handle.

(b) Decision trees and decision tables, often used in traditional generative CAPP systems, work effectively only for simple decision making processes. These are primary methods to describe knowledge and are coded line by line in the program. Any modifications to the existing knowledge would require rewriting of at least some portion of the original program. To automate the process planning functions, the system must be able to perform certain level of intelligent reasoning. An expert system developed using AI techniques organizes the domain knowledge and employs inference mechanism to reason intelligently.

(c) An expert system stores knowledge in a well-organized manner so that it is easier to add, delete and modify its knowledge base without re-coding the program.

The current use of Artificial Intelligence (AI) techniques in automated process planning can be divided into two parts:

(1) the use of AI for automated interpretation of the part description to perform geometric reasoning about the shape, features and relationships between features

(2) Expert system for the development of the process planning system itself

Most of the past research has been focused on the above areas (Joshi and Chang, 1990). However, in order to develop a successful system, a common framework incorporating the two parts is
needed. Some automatic planners using Artificial Intelligence have been proposed and implemented, e.g. GARI (Descotte and Latombe 1981), PART (Houten et al. 1989).

Expert process planning systems have been developed both at the macro level and at the micro level. At the micro level, expert systems have been designed to plan the sequence of operations and the resources required to manufacture individual products. At the macro level, planning systems look at the job shop scheduling needs and the constraints of an entire factory.

1.8.1 The structure of process planning knowledge

The knowledge of process planning may be divided into three categories, i.e. the knowledge of problem awareness, the knowledge of the domain of application and the knowledge of problem solving [Zust and Taiber 1990]. The knowledge of problem awareness expresses the actual problem of machining tasks, covering the description of the workpieces, geometry, tolerances, materials and additional machining data. The knowledge of the application domain describes tasks such as cutting processes, tooling, tool paths, cutting data, process capabilities including necessary geometrical and technological conditions relative to the elements to be manufactured and other unrelated practical issues. The knowledge of problem solving includes tasks related to planning and processing strategies.

1.8.2 Some examples of expert process planning systems

GARI was developed at the university of Grenoble in France (Descotte Y, Latombe J C, 1981, 1984). GARI is probably the first AI-based CAPP system reported in the literature (Alting & Zhang 1989). GARI presents feature types, dimensional information, inter-feature relationships, tolerances and surface finish data. Some machining related information is also contained in the feature representation. The process planning is represented in a three-level hierarchy, i.e. single cut, operation phase and machine tool. The machining knowledge is expressed as rules of the form "if ... then ....". The rules govern precedence and grouping of cuts, machine selection, set-up, and so forth. GARI generates process plans by, constraining the initial plans, and selecting a rule of maximal weight among those rules whose conclusions are satisfied. The output seems very sensitive to weights associated with the rules. This is undesirable because the numerical weights are assigned somewhat arbitrarily, to reflect qualitative preferences. Part features in GARI are assumed to be orthogonally positioned. Another important shortcoming of GARI is the lack of a complete solid model. Thus GARI may produce plans that are unfeasible because, for example, there are collisions or certain features are inaccessible in a particular set-up. In short, GARI is represented as a significant advance in automatic process planning, but left several issues unanswered.
EXCAPP, standing for EXpert Computer–Aided Process Planning, was designed to generate process plans for rotational parts (Zhang et al 1988, 1989, Joseph et al 1990). EXCAP can not generate process plans for complex parts because the part geometry is limited to simple rotational parts. A feature recognition knowledge base, rather than an algorithmic approach is used for identifying features.

The process plans generated by EXCAPP consist of two phases. In the first phase, the sequence in which the volume–removal operations are to be performed is determined. In the second phase, parameters such as feed rate, speed, and depth of cut for the operation in the first phase are determined. A backward chaining inference strategy is used in EXCAPP.

QTC, standing for Quick Turnaround Cell, is an automatic process planning system (Chang and Wysk 1984, Chang 1990, Turner and Anderson 1988). The QTC system is capable of not only performing process planning, but also design, cell control, and vision inspection. The design system has a featured–based interface. After a design is completed, the design data—a feature file—is sent to the process planning system. The process planning module reasons about the manufacturing features, their relationships, and feasible approach directions. This information is then sent to an expert process planner for processes, tools, fixture selection, and process sequencing. The process planning module of the QTC system is called AMPS (Automatic Machining Planning System) (Kanumury et al 1988). AMPS planner is implemented in the expert system shell KEE (Intelicop). The major functional modules including tool database management system and machinability database management system are written in C language. The C functions are interfaced with KEE through a KEE/C interface package.

APPAS is an acronym for 'Automated Process Planning and Selection' developed by Wysk at Purdue (Wysk 1977). It is probably the first well–known generative CAPP system written in standard FORTRAN with description of the detailed technological information of machined surface by means of a special code.

CADCAM is an extension of APPAS developed by Chang and Wysk (1985) which links APPAS with an interactive computer graphics terminal to demonstrate the concept of an integrated CAD and process planning system.

TIPPS (Totally Integrated Process Planning System) is a generative process planning system developed by Chang and Wysk at Virginia Polytechnic Institute and state University in 1983 (Chang and Wysk 1983). It is perhaps the first system that integrates CAD with generative process planning into a unified system utilizing AI techniques and decision tree approaches. TIPPS uses a boundary representation from a CAD database for a part. A user applies the crossbar cursor on a graphics terminal and a menu display to specify surfaces to be machined. The system then utilizes the information stored in a process knowledge base to determine manufacturing pro-
cesses, sequence, cutting parameters and time estimation. However, features are recognized with human assistance. Full automation has not been achieved.

HAPPI (Edinburgh process planning system) was developed at Edinburgh University to investigate methods for the automatic generation of optimal process plans in a multi-machine environment (Edinburgh 1989). HAPPI plans prismatic two and half degree (2½D) components where the features lie in orthogonal directions. It has the ability to handle geometric and in particular relational tolerances. Generic algorithms are applied to the sequencing of the optimal plans from a network of all feasible plans.

PART (Planning of Activities, Resources and Technology) is a generative CAPP system developed at Laboratory of Production Engineering of University of Twente in the Netherlands (Houten et al 1989). A Boundary Representation (Brep) modeller was used together with a CAD interface which translates the file from the CAD system into STEP/PDES (see section 1.9 for STEP/PDES) and vice versa. A syntactic pattern recognition approach is used for manufacturing feature extraction. Tolerances can not be defined before the feature recognition process. The system integrates both process and some production planning functions. A specification language called SSL (Scenario Specification Language) was used to construct its feature recognition module.

In PROPLAN (Philips et al 1984), a CAD system was developed to facilitate part design, data storage and retrieval processes, and subsequent data manipulation for the requirements of the CAPP module. Thus, a CAD database can be viewed as an integrated module within the overall system. In fact, the major emphasis in PROPLAN is on the problem of interfacing a CAD system with an expert system based process planner.

1.9 CAD/CAM data exchange standards

It is recognized world-wide that an international standard for CAD/CAM data exchange is necessary for the integration of CAD/CAM systems. The first attempt was made by the Initial Graphics Exchange Specification (IGES) Organization in 1979 (Smith 1983, Smith et al 1988). The aim of the organization was to develop a data standard format for the exchange of graphical information, including 2D drawings and 3D wireframe models. The later versions include exchange of information in solid models. Some of the criticisms of IGES are: the inability to ensure accurate and complete data translation because of fundamental conceptual differences in CAD systems, the loss of surface data in translation, the inability to convert attributed information, etc. (Shah and Mathew 1991).

To overcome the problems with IGES and to broaden the scope of CAD data exchange beyond geometry and topology, IGES organization initiated a second project in 1984: Product Data Ex-
change Specification (PDES) (Kelly, 1986). The aim of this project was to develop a single unified standard called the STandard for Exchange of Product data (STEP) (Danner 1990, Mason 1991). Form feature information model is one of the product data models in PDES/STEP, which, when fully developed, will provide a mechanism for exchanging definition data for a wide variety of products.

1.10 The objectives and scope of this research project

From the above discussions, it can be seen that large research effort is devoted to the development of automated CAPP systems. One of the main difficulties in CAD/CAM integration is the representation of feature information. This research project, which was sponsored by SERC (the Science and Engineering Research Council), aimed to contribute to solving some of the problems encountered in the development of automatic process planning systems. The emphases of the work are as follows:

* to investigate the representation of a feature-based component model and process capability data in process planning systems and to explore the use of a knowledge-based environment in structuring and integrating the diverse information requirements of the planning domain;

* to develop and represent some of the basic geometric reasoning logic needed in process planning and to examine the extent to which KBS facilities can be utilised to simulate human process planning logic;

* to develop a prototype generative process planning system and to link it with a feature-based design system to demonstrate some aspects of automated process planning and CAD/CAM integration.

* to identify the problems and difficulties in completing the above tasks and to propose approaches to solving them.

This project is currently aimed at one-of-a-kind prismatic components. Most parts modelled by the developed prototype system can be machined on machining centres. Rotational parts are considered outside the scope of this work.
CHAPTER TWO

The Knowledge–Based Process Planning System – GENPLAN
Chapter 2: The Knowledge–Based Process Planning System – GENPLAN

In this chapter, the structure of the developed prototype knowledge–based process planning system – GENPLAN, standing for GENerative PLANing, will be introduced. The software tool used in the project, a knowledge–based system "Generis" will also be introduced. The details of main modules of the planning system will be discussed in the following chapters.

2.1 Introduction to Generis

Generis is a software system for the development of knowledge–based applications in engineering, finance, health services, etc (DSL 1988, 1990). It combines the capabilities of a relational database and a rule based inference engine in a single, integrated product.

Standard relational database systems have been successfully used in representing tabular data (precise data) for many years. The tabular data is commonly encountered in many business applications. However, in our everyday life, there are all kinds of irregular, imprecise and poorly structured data which are difficult to handle using conventional database systems. Generis retains the inherent simplicity of the relational model while extending its capabilities in a number of important ways to handle real–world knowledge.

2.1.1 Knowledge in Generis

The knowledge in Generis is represented in the form of facts and rules. Facts are of the form:

<subject> <relationship> <property>

where "subject" can be the name of any entity. An entity is any object, concrete or abstract, or class of objects. "Property" can be the name of any other entity, or an attribute. "Relationship" is a named relationship between the subject and property. An example of a fact would be:

Part consists of features.

where "part" is the subject, "consists of" is the relationship, and the "features" is the property.
A property in one fact may be the subject of another. For example:

**Feature contains faces.**

A particular property may either be an entity or an attribute. In the above examples, features and faces are both entities. The following example shows a property as an attribute.

**Feature has area 100.00**

Where area is the attribute’s name and 100.00 is its value. Five types of attribute are provided in Generis, i.e. integer, decimal, date, time and text.

The central concept behind the representation of knowledge in Generis is the entity. Entities represent objects in the outside world. Every entity will, in general, be a member of one or more classes. For example:

**feature1 is a pocket**

**h1 is a hole**

Using this relationship, one entity is made a member of another, i.e. the latter becomes a class or set containing the former. No distinction is made by the system between entities and classes, and the class membership links can therefore be built up into a network of arbitrary complexity.

Groups of facts relating to a single class of subject are held together in a **table**. Examples of tables can be found in the next chapter.

**Inference rules** are of the form:

**<fact> if <condition>**

where "fact" can be any fact of the form above. "If" is a keyword. "Condition" is any combination of facts which, if true, results in the initial fact in the rule being accepted as true. An example of a rule would be:

feature.1 has feasible operation.1 if feature.1 requires surface_finish.1 and operation.1 produces surface_finish.2 and surface_finish.1 > surface_finish.2.

Rules are grouped together in **rulesets**. Each ruleset is given a name and one or more of the rulesets in an application can be used in conjunction with the facts held in tables in answering queries and in producing reports.
In Generis the collection of tables used in an application is referred to as the database. The collection of rulesets in an application is called the rule base. The tables and rulesets (the database and the rule base) together with the class membership data constitute a knowledge base.

2.1.2 The main features of Generis

The selection of Generis as the software tool for the development of the prototype process planning system is based on its facilities which meet the requirement of the development. The main features of Generis are listed below (DSL 1990):

* Generis inherits relational database technology which is widely used in industry to represent tabular engineering data.

* It can also represent inference rules and imprecise, poorly structured data which is difficult for the standard relational database systems to handle.

* An inference mechanism can be developed using the command language facilitated in Generis. This mechanism is also called an inference engine. The inference engine retrieves the data and switches on/off the rules stored in the knowledge base in a pre-determined sequence and makes decisions.

* Data and rules stored in the knowledge base can relate to individual objects or to entire classes of objects.

* A program interface to Generis query language is available, allowing other application programs to make queries to Generis applications. A further low level interface to the Generis data structure is also available.

* Comprehensive arithmetic and data processing facilities are provided, including data arithmetic and automatic summarisation.

* External application softwares can be executed from within Generis. Information can be transferred between them through parameters.

* Generis is written in C and available on a range of UNIX and UNIX–like environments.

The above facilities are extremely useful for this application in a number of aspects such as: (1) the representation of manufacturing rules and descriptive information and knowledge in the process planning domain which would be difficult
to represent using conventional relational database systems; (2) the inference mechanism is used for process planning decision making and also for geometric reasoning functions; (3) the interfacing facilities are useful for the integration of the process planning system with a feature-based design system; and (4) the C and UNIX working environment is the same as the design system.

2.2 The conceptual model of the prototype system

The functional modules of the integrated design and planning system are illustrated in Figure 2.1. The system consists of three parts: a feature-based design system, a process planning system, a feature processor which is written in C.

A part is designed using a feature-based design approach. The design system was developed by adding a design by features front end to a commercially available solid modeler Imaginer (Pafec Limited 1991). The output of the design system is in terms of features, rather than conventional geometric representations. This has avoided the need for feature recognition. The feature processor reads the output data file from the design system and reformats it into Generis readable structure. It also calculates additional data required for process planning automatically. The planning system accepts the formatted data file and builds up a part data model and stores it in the knowledge-base system. It also reasons about the geometry, deduces the relationships between features and feasible approach directions to the features on the components. The planning system outputs a list of machines selected, the sequenced machining processes, and tools required for each machining process through the printer.

2.3 The structure of the process planning system

The main work carried out in this project is the development of the process planning system and the integration with the feature-based design system. The prototype process planning system GENPLAN consists of five main components as shown in Figure 2.2: a Knowledge Base (KB), a working memory, an inference engine, a user interface and a feature processor.

The knowledge base contains a database and a rule base. The database stores the Feature–Based Component Data Model (FBCDM) and the Process Capability Data Model (PCDM). The rulebase contains manufacturing rules for planning decision
making. The inference engine is the heart of the system. It updates the knowledge base, performs logical reasoning and deduces new knowledge by applying rules to facts until the posed problems are solved.

The inquiry-specific information and temporary results are stored in the working memory. The KBS uses this information at run-time to store transient and dynamic information, such as data values, when facts have been instantiated, or when rules have been fired. This is the part of the host machine memory allocated to the KBS, which is controlled by the inference engine.
The user interface provides the link between the planning system and the outside world. For example, a knowledge acquisition sub-system may be used by the KBS application builder to add facts and process details to the knowledge-base. The interface with other systems is one of the most significant components in the diagram. In this project, the interface to the feature-based design system is implemented. The feature processor reads output from the design system and reformats the data into the form that Generis accepts. This will be discussed in more detail in later chapters.

Generis also has explanation facilities which allow knowledge engineers to build explanations into applications about why questions are asked, how conclusions have
been reached and general help information. The inference engine causes explanatory messages to be displayed in response to user requests.
CHAPTER THREE

The Part and Process Capability Models
Chapter 3: The Part and Process Capability Models

3.1 Introduction

The implemented prototype planning system includes two data models: part data model and process capability data model. The information contained in the part data model includes geometric, technological and manufacturing data, etc. Whilst the process model consists of information about the production resources and capabilities such as operations, tooling, machine tools, etc.

As discussed in the chapter 2, the information content of the two models and the rules in the rulebase is called knowledge (rules will be discussed in the next chapter). The knowledge in both part and process capability models can be represented as absolute knowledge plus constrained knowledge. The absolute knowledge about components is its shape and topology information, the constrained knowledge relates to technological constraints such as materials, dimensions and tolerances, surface finish etc. The absolute knowledge in the process capability model is the machining operations and tool types, and the constrained knowledge is the limits of the absolute knowledge and other technological constraints. Figure 3.1 shows the absolute and constraint knowledge in the part and the process capability data models.

From Figure 3.1 it can be seen that the absolute knowledge in both data models is used to generate initial (preliminary) process plans for machining components, and the constrained knowledge in both models is used for optimizing the preliminary plans.

3.2 The Feature Representation Scheme for the Part Data Model

The part data model is feature-based. Form features are defined as volumes enveloped by sets of real faces and imaginary faces (Gindy 1989). Real faces are the enveloping faces which exist on the components. Imaginary faces are the enveloping faces which do not exist on the components. Form features plus manufacturing data (attributes) are called manufacturing features. The representation of features is crucial for process planning and also for the integration of planning systems with Computer-Aided Design (CAD) systems, as features are information carriers throughout the product life cycle from conceptual design to production.

23
3.2.1 Feature Taxonomy

In the part data model, Gindy's feature taxonomy scheme (Gindy 1989) has been adopted. In this scheme, nine basic feature classes are identified, which are bosses, pockets, holes, non-through slots, through slots, notches, steps, real faces and imaginary faces (as shown in figure 3.2). The feature classes are determined by the number of External Access Directions (EADs), ranging from 0 to 6 respectively. External Access Directions (EADs) are normal vectors to the imaginary surfaces of features, defined with respect to feature local co-ordinate frames. An EAD can be considered as a possible machining direction to a feature volume, and a feature has as many EAD's as it has imaginary surfaces. Each feature class has a unique number of EAD's. Other characteristics of features such as open, closed, through, non-through, etc are also of importance in manufacturing planning.
For each feature class, a number of profile shapes can be attached. For example, a boss may have a round profile, a rectangular profile, a triangular profile, or a hexagonal profile, etc.

Figure 3.3 is a slot feature with its classification characteristics. The slot is classified into category depression, with profile type square. It has three external access directions and its boundary is open. The entry and exit status is through.
3.2.2 Feature Relationships

The relationships between adjacent features on a component, is referred to as feature connectivity, can be represented in a binary matrix (see Figure 3.4). It represents the parent/child relationships between features on the component by using inheritance rules between the component real and imaginary surfaces. In figure 3.4, all the real surfaces of the component (s1, s2, s3, s4, s5, s6, s7, s8, s9, s10) are tagged and represented as the columns of the connectivity matrix. The features on the component (slot f1, hole f2 and the 6 faces of the block s1, s2, s3, s4, s5 and s6) represent the rows of the connectivity matrix. If the face of a feature is real and coincides with a face of the component, then it is coded as '1'. Whilst if the face of a feature is imaginary and coincides with a face of the component, it is coded as '0'. And in this case, the parent face of the feature's imaginary face is the real face of the component.

The parent/child relationships can also be illustrated by a feature connectivity graph as shown in figure 3.5. The sense of arrows is from parent features to child features. For example, from surface s5 in Figure 3.5, it may be possible to machine features f1 and f2, since f2 is a child feature of f1 and f1 is the child of s5. However, although s1 and s3 are also parent features of f1, f2 still cannot be machined from s1 or s3. This is because the External Access Directions from both s1 and s3 to f1 are
different from the EAD from f1 to f2. Therefore, the parent/child relationships alone can not be used to determine the machining directions to child features. EADs must be taken into account in determining the feasible machining directions for features.

The approach directions of a feature are unobstructed paths such that a tool can access the features on the workpiece. In order to determine the feasible approach directions, one must check whether there are any obstacles blocking the tool approach to the feature. This can be achieved by examining the geometry of the
Figure 3.5: The feature connectivity graph

feature and the relationship between the feature and its parent feature. If the check is successful, the EADs are feasible tool access directions to the feature.

Figure 3.6 illustrates how the feasible approach directions to a child feature are determined. Both Figure 3.6 (a) and (b) show a parent feature non-through slot and a child feature round hole. The round hole in (a) has two access directions and one of them is through its parent feature. The situation in part (b) is different. One of the access directions to the round hole is restricted by its parent feature. So the hole has only one feasible access direction.
3.2.3 The Content of Feature Information

From the above discussions, it can be summarized that the feature information should include feature class, face type (real or imaginary), EADs, Parent–child relationships, positions and orientations and technical attributes such as tolerances and surface finish, etc. Figure 3.7 shows the information about a slot. A square through slot has a square profile, three real faces, three imaginary faces and three external access directions. A local coordinate frame is attached to the feature for the purpose of representing the position and orientation of the slot in the component coordinate frame. The dimensions, tolerances and surface finish information is also included.
Feature Name: Square Through Slot

Geometric Information: Faces s1 s2 s3 s4 s5 s6;
  Real Faces: s1 s2 s3;
  Imaginary faces: s4 s5 s6;
  EAD: Ead1, Ead2, Ead3;
Relationships of Features: Parent–child;

Local Coordinate Frame: O – ijk ;
Position: (X0, Y0, Z0) ;
Orientation: (ai, Bi, ri), (aj, Bj, rj), (ak, Bk, rk);
Technological Information:
  Dimension & Tolerance & Surface finish
  Length       Tol_L
  width        Tol_w
  Depth        Tol_d

Figure 3.7: The data content about a slot feature

3.2.4 The content of the data about parts

The feature–based part data is represented as two levels: the global level (part level) and feature level. The feature level information has been discussed in the above section (3.2.3). The global level information includes stock geometry and technological specifications, e.g. materials, hardness, quantity, etc.
The stock/block is represented as a special feature. The difference between stock features and other features on the component is that the stock is a boss but with external access directions to each of its faces which are the Potential Access Directions (PADs) of the component (as shown in figure 3.8). If all of the features on the component are orthogonal, then the number of EADs of the block would be equal to the number of component PADs. If there are non-orthogonal features on the component, the number of PADs is equal to the number of block EADs plus the number of feasible non-orthogonal EADs of the non-orthogonal features. The basic block shapes are cylindrical for rotatory parts and cubic for prismatic parts. The faces of the block could be used for the determination of set-ups and fixture strategy.

The features shown in Figure 3.8 (i.e. round hole1, round hole2, square step1, square step2 and square pocket1) are all orthogonal. Therefore, they can be machined from a maximum of six component PADs.

### 3.2.5 Representing part data in the Knowledge-Based System

The data structure of the part information model in the prototype Knowledge-Based process planning system can be illustrated by Entity Relationship Diagrams. In the KB the actual component data is stored as relational tables. Figure 3.9 is the entity relationship diagram of a component model. It consists of features and attributes, such as specifications, materials, hardness, quantity, stock and Potential Access Directions, etc.

Figure 3.10 shows the table structure for part level information. It has two separate tables. The first table contains the names of features on the part. The second table contains global specifications about the part.

Figure 3.11 shows the entity relationship diagram of single features. A feature has a name, a group of EADs, a group of faces (real faces or imaginary) and a local coordinate frame.

The feature data is stored as tables in the Knowledge Based system as shown in Figure 3.12. The EADs and face names are stored in the same table called "features". The details of the faces are stored in another table called "faces". The names of the
Specifications include: materials, hardness, block size, roughness, etc.

Figure 3.8: Global level information about a component

local frame are stored in a table called "positions" and the parameters of the local frame are stored in a separate table called "origins".

The complete structure of the set of tables for part and feature level information can be found in Appendix I.

3.3 The Process Capability Model

The process capability model in GENPLAN system includes information about machine tools, machining processes, operations and cutting tools. Those issues are discussed next.
3.3.1 Representing the capability of machine tools

A machine tool is represented as chains of elementary kinematic motions (translations and rotations). Rules for grouping these motions constitute the machine’s form generating capability. The machine’s form generating capabilities can be described as sets of elementary form generating schemas. A form generating schema includes cutting tool geometry and the motions of the tools with respect to workpieces (Portman 1981, 1982). The motions are divided into four types, i.e: T — translatory motion, the direction of which is perpendicular to both the axis of the nearest proceeding rotation as well as to the direction of the proceeding translatory motion; H — translatory motion, the direction of which is parallel to the axis of the proceeding rotation; R — rotation, the axis of which is perpendicular to the axis of the preceding rotation; C — rotation, but for parallel axis of rotation.
Figure 3.10: The table structure for part level data

Figure 3.13 shows some of the form generating schemas that were used to represent operations performed on machine tools. Figure 3.13a is a plan-ing operation. It can be represented as a schema with a plan-ing tool and two motions, i.e. two translations. Figure 3.13b is an end milling operation with an end mill tool and three motions: i.e. one rotation and two translations. Figure 3.13c is a side milling operation with a side mill tool and three motions, i.e. one rotation and two translations. Figure 3.13d is a surface grinding operation with a grinding wheel and three motions i.e. one rotation and two translations. Figure 3.13e is a grinding
operation with a grinding wheel and three motions, i.e. one rotation, one translation and another opposite rotation.

A machine tool's capability can be represented by a set of resource elements (REs) and each RE is a set of form generating schemas (operations) which can be obtained from one or more of its available machine tools (Gindy 1991). The REs, therefore, provide a basis for comparing the capabilities of different machine tools and give the planning system an ability to generate alternative process plans on less than a whole machine tool basis. Figure 3.14 shows a number of machine tools which are represented as resource elements in the prototype system.
<table>
<thead>
<tr>
<th>TABLE</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN SUBJECT</td>
<td>feature</td>
</tr>
<tr>
<td>VALUEDNESS</td>
<td>multi_valued GENERIC on</td>
</tr>
<tr>
<td>RELATIONSHIP</td>
<td>PROPERTY</td>
</tr>
<tr>
<td>has face</td>
<td>face</td>
</tr>
<tr>
<td>has ead</td>
<td>vector</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE</th>
<th>origins</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN SUBJECT</td>
<td>origin</td>
</tr>
<tr>
<td>VALUEDNESS</td>
<td>multi_valued GENERIC on</td>
</tr>
<tr>
<td>RELATIONSHIP</td>
<td>PROPERTY</td>
</tr>
<tr>
<td>along axis</td>
<td>axis_g</td>
</tr>
<tr>
<td>has coordinate</td>
<td>coordinate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE</th>
<th>positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN SUBJECT</td>
<td>feature</td>
</tr>
<tr>
<td>VALUEDNESS</td>
<td>single_valued GENERIC on</td>
</tr>
<tr>
<td>RELATIONSHIP</td>
<td>PROPERTY</td>
</tr>
<tr>
<td>has local origin</td>
<td>origin</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE</th>
<th>faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN SUBJECT</td>
<td>face</td>
</tr>
<tr>
<td>VALUEDNESS</td>
<td>single_valued GENERIC on</td>
</tr>
<tr>
<td>RELATIONSHIP</td>
<td>PROPERTY</td>
</tr>
<tr>
<td>has normal</td>
<td>vector</td>
</tr>
<tr>
<td>has face type</td>
<td>type</td>
</tr>
<tr>
<td>has surf finish</td>
<td>surf_finish</td>
</tr>
<tr>
<td>angle to</td>
<td>face</td>
</tr>
<tr>
<td>has parent face</td>
<td>face</td>
</tr>
</tbody>
</table>

Figure 3.12: The table structure for feature level data
OPERATION NAME: planning
MOTION SET: TT
TOOL NAME: shaping tool

OPERATION NAME: end milling
MOTION SET: RTT
TOOL NAME: end mill

OPERATION NAME: peripheral milling
MOTION SET: RTT
TOOL NAME: side mill

OPERATION NAME: surface grinding
MOTION SET: RTT
TOOL NAME: grinding wheel

OPERATION NAME: cylindrical grinding
MOTION SET: RTC
TOOL NAME: grinding wheel

Figure 3.13: The defined form generating schemas
Machine Resource element operation

<table>
<thead>
<tr>
<th>Machine</th>
<th>Resource element</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makino</td>
<td>Re1_dr</td>
<td>Drilling</td>
</tr>
<tr>
<td></td>
<td>Re2_efm</td>
<td>Reaming</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>End_milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Face_milling</td>
</tr>
<tr>
<td>Surface_grinder</td>
<td>Re4_fg</td>
<td>Surface_grinding</td>
</tr>
<tr>
<td>Cylinder_grinder</td>
<td>Re5_hg</td>
<td>Hole_grinding</td>
</tr>
<tr>
<td>Planner</td>
<td>Re3_p</td>
<td>Shaping</td>
</tr>
</tbody>
</table>

Figure 3.14: The capability of machine tools

3.3.2 Representing processes and operations with features

Machining process data includes operation types (such as milling, turning, grinding, etc.), cutting tools, shape geometry that can be produced, machine tools which can perform the process etc. The relationships between processes should also be taken into consideration. For example, a drilling process may have a relationship with other processes, such as reaming and boring (e.g. a drilling process may be followed by a reaming and then a boring process). In fact, the boring operation can not be performed unless a hole already exists.

A Feature Transition Diagram (FTD) is a representation of all the possible Feature Technological Solutions (TSFs) that can be used to produce a feature geometry and topology at various levels of technological requirements (Gindy and Huang 1992b). A fully designed feature has a set of equally feasible and equally weighted TSFs and each TSF is an ordered set of operations capable of producing the feature form (geometry and topology) and satisfy its technological requirements (mini-plan for producing a feature).
Figure 3.15 shows the feature transition diagrams for machining different types of holes, i.e. through round holes, counterbore holes, shallow round holes and threaded holes.

The information which is used to determine operations to machine features is shown in Figure 3.16 and Figure 3.17. The information is based on the capabilities of conventional machining processes.

The feasible processes for machining features and satisfying the technological constraints are called Feature Technological Solutions (TSFs). TSFs are also called mini–plans in our system. In fact, feature technological solutions are ordered sets of operations, and stored in the system according to the feature transition diagrams.

Feature technological solutions are defined for each feature at subclass level which includes the information of feature profiles. For example, to make a TSF for a hole, the information such as through or not through, round or square, flat bottom or cone etc is required. Appendix II lists some of the TSFs for feature classes defined in the prototype system.
3.3.3 Representing process capability model in the Knowledge-Base System

As discussed above, the process capability model contains information about machines, tooling, processes and operations. Machine and tooling capabilities are represented as form generating schemas and resource elements (sets of form generating schemas plus technological capability). Processes and operations are represented as feature technological solutions associated with feature transition diagrams. Figure 3.18 is the entity relationship diagram of the process model. The table structure for the process capability model is shown in Appendix III.
As shown in Figure 3.18, features are associated with the miniplans which are actually the Technological Solutions for producing the feature shapes and satisfying technological requirements. The determination of the feature technological solutions are based on the feature geometry and technological requirements, such as tolerances and surface finish. Feature technological solutions are sequenced sets of operations. Once the operations are selected for producing the features, the resource elements can be found and therefore the machines can be selected, since machines are represented as resource elements and form generating schemas. The cutting tools can also be

<table>
<thead>
<tr>
<th>Feature</th>
<th>Process</th>
<th>Operation</th>
<th>Tool</th>
<th>Surface Finish (um)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square_slot</td>
<td>Milling</td>
<td>peripheral milling</td>
<td>Slitting Saw</td>
<td>0.76 ~ 1.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>End Milling</td>
<td>Plain</td>
<td>1.27 ~ 1.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Shell End</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Broaching</td>
<td>Form Tool</td>
<td>0.81 ~ 3.17</td>
</tr>
<tr>
<td>square pocket</td>
<td>Milling</td>
<td>End Milling</td>
<td>Plain</td>
<td>1.27 ~ 1.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Shell End</td>
<td></td>
</tr>
<tr>
<td>Blind Round Hole</td>
<td>Drilling</td>
<td>Drilling</td>
<td>Twist Drilling</td>
<td>2.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deep drilling</td>
<td>Spade Drill</td>
<td></td>
</tr>
<tr>
<td>deep hole</td>
<td>Drilling</td>
<td>Drilling</td>
<td>Deep-hole drill</td>
<td>&gt; 2.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gun drill</td>
<td></td>
</tr>
<tr>
<td>Counterbore Hole</td>
<td>Drilling</td>
<td>Drilling</td>
<td>Counterbore</td>
<td>2.54</td>
</tr>
<tr>
<td>Shallow Round Hole</td>
<td>Drilling</td>
<td>centre drilling</td>
<td>Centre Drill</td>
<td></td>
</tr>
<tr>
<td>Round Holes</td>
<td>Honing</td>
<td>Honing</td>
<td>Honing Stone</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td>Lapping</td>
<td>Lapping</td>
<td>lap</td>
<td>0.035 ~ 0.12</td>
</tr>
<tr>
<td></td>
<td>Tapping</td>
<td>Tapping</td>
<td>Tap</td>
<td>1.90</td>
</tr>
</tbody>
</table>

Figure 3.17: The capability of conventional machining processes
selected for each operation based on the shape and technological requirements of the features. The tooling data stored in table form in the system is shown in Figure 3.19.

The integrated product and process model implemented in the prototype process planning system is illustrated in Figure 3.20.
<table>
<thead>
<tr>
<th>TABLE</th>
<th>toolbank</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN SUBJECT</td>
<td>tool</td>
</tr>
<tr>
<td>VALUEDNESS</td>
<td>single_valued</td>
</tr>
<tr>
<td>RELATIONSHIP</td>
<td>PROPERTY</td>
</tr>
<tr>
<td>has tool code</td>
<td>tool no</td>
</tr>
<tr>
<td>has description</td>
<td>descrip tool</td>
</tr>
<tr>
<td>is made of</td>
<td>material</td>
</tr>
<tr>
<td>has tool life</td>
<td>tool life</td>
</tr>
<tr>
<td>needs power</td>
<td>power</td>
</tr>
<tr>
<td>has cutting edge</td>
<td>cutting edge</td>
</tr>
<tr>
<td>has recom speed</td>
<td>surf speed</td>
</tr>
<tr>
<td>has recom feed</td>
<td>cutting feed</td>
</tr>
<tr>
<td>has max depth</td>
<td>max depth</td>
</tr>
<tr>
<td>has diameter</td>
<td>dia tool</td>
</tr>
<tr>
<td>has coolant</td>
<td>coolant</td>
</tr>
<tr>
<td>has station no</td>
<td>station no</td>
</tr>
<tr>
<td>used in</td>
<td>machine</td>
</tr>
<tr>
<td>has remarks</td>
<td>remarks</td>
</tr>
<tr>
<td>has thread pitch</td>
<td>pitch</td>
</tr>
<tr>
<td>has thread spec</td>
<td>spec</td>
</tr>
</tbody>
</table>

Figure 3.19: The table structure for cutting tools
Figure 3.20: The partial planning data reference model
CHAPTER FOUR

The Planning Logic and Rule sets
Chapter 4: The Planning Logic and the Rulesets

In this chapter, the underlying planning logic of the prototype system is explained. The manufacturing rules developed and stored in the rulebase (called rulesets) are also discussed. The rules are invoked by the inference engine to make process planning decisions. In the next chapter, examples will be given to show the procedure for automatically generating process plans and the integration with a feature–based design system.

4.1 The general principle

The general idea of process planning is to match the requirements for producing component features with the available processing capabilities of the manufacturing systems. This match is done in two steps. The first step is to check, on a feature by feature basis, whether the shape of the features on the component can be produced by the available manufacturing system, regardless of the sizes, accuracy of features and the material, cost of the component, etc. In other words, only absolute knowledge is taken into account in this step.

The second step is to check whether the technological requirements of the component features can be satisfied by the capabilities of the manufacturing system. In other words, this step of the match is based on the constrained knowledge (materials, surface finish, sizes, tolerances, etc).

4.2 The selection of machining processes and cutting tools

There are many factors which affect the selection of suitable machining processes for components. Some of these factors are commercial, such as the cost of production. Other factors are of a more technical nature, which are discussed below.

The shapes of components dictate the selection of manufacturing processes. The quantity, or batch size, dictates the economics of the processes.

In the knowledge–base system, process capabilities are represented as process knowledge and organized as groups of sequenced operations. The rules are written in a backward planning approach. To select a feasible machining method, the user inputs the required machined features together with technological requirements. The planning system then generates the planned processes for each feature on the component automatically. The criteria for process selection are:
* operation type
* tool geometry

When the processes are selected, the appropriate cutting tools are then selected. Some of the factors which affect the selection of cutting tools are listed below:

(a) Machining processes
(b) Workpiece material
(c) Cutting tool material
(d) Cutting tool geometry (single point tools or multi-point tools)
(e) Process variables – cutting speed, feed, depth of cut, power available
(f) Application of a cutting fluid

For simplicity, at the current stage we only consider machining processes and cutting tool geometry. Machining processes require tool types (e.g. a drilling operation requires a drilling tool). Tools are classified into single-point and multi-point based on the geometry of the tools.

4.3 The Optimisation Criteria for Set-up Determination

When the processes and cutting tools are selected, the set-ups of the components on the machines need to be determined and optimised. The criteria used by the prototype planning system are as follows:

\[ \begin{align*}
\text{a) Minimise the resource elements required for producing the features on the components} \\
\text{b) Minimise the number of set-ups for each resource element used. This also means maximising the number of operations performed in parallel (if feasible)} \\
\text{c) Sequencing the operations for each set-up so as to minimise the number of tool changes required to produce component features}
\end{align*} \]

The determination of the final component set-ups involves a two stage optimisation procedure. In the first stage, optimization is performed at the feature level, and in the second, the component as a whole is considered.

At the feature level, the objective is to select a single Feature Technological Solution (TSF) from among the equally feasible equally weighted alternatives for each component feature. All the resources needed for the operation required to produce all component features are considered.
Based on the required surface finish of each feature, a TSF is selected which contains the minimum number of operations and generates the lowest value of surface roughness. The TSFs selected for all the features belong to different sets of resources.

To optimize these resources, feature clusters are established at component level. Feature clusters take two facts into account: one is the component potential access direction "PAD" and another is the resource element. A component PAD is a common direction that coincides with the EAD of at least one of its features. The features to be machined are listed under each cluster (as shown in figure 4.1).

<table>
<thead>
<tr>
<th>CLUSTER</th>
<th>PAD</th>
<th>RE</th>
<th>FEATURES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S1</td>
<td>RE.1</td>
<td>F1+F2+F5</td>
</tr>
<tr>
<td>2</td>
<td>S4</td>
<td>RE.3</td>
<td>F8+F3</td>
</tr>
<tr>
<td>3</td>
<td>S3</td>
<td>RE.1</td>
<td>F1+F5+F7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Figure 4.1 Feature Clusters

The final selected (planned) processes should contain the minimum number of feature clusters required for machining all the features in the component. The feature clusters which are listed but not included in the planned processes, are omitted.

The final step is to find a list of machines, the number of set-ups on each machine, and a list of features to be machined using the information in the process capability model.

4.4 The manufacturing rules in the rulebase

The manufacturing rules are stored in the rule base of Generis as rulesets. The rules in the same ruleset together perform a single function, although they may require different pieces of information from the part data model or the process data model. Usually one rule set is invoked by the inference engine for each step of the inferencing process.
The rules in the prototype system are grouped into different modules, and the rule modules are divided into two levels (see Figure 4.2). Applicable rules for a specific function are identified and invoked by evaluating the conditions in all rules in different rule modules according to the current database state. The data-driven approach adopted in Generis allows rules to generate new facts from the database and add them to the database.

In Figure 4.2, the rules at level.1 are also called Global rules. The four modules at level.1 are as follows:

(a) Global rules for process selection/sequencing
(b) Global rules for Feature Technological Solution (TSF) selection
(c) Global rules for geometric reasoning
(d) Global rules for set-up optimization

---

Figure 4.2: The Rule Modules in the Prototype System
The process selection module contains rules for selecting and sequencing roughing, semi-finish and finishing operations required for producing features on a component. It is assumed that only one cut is needed for each operation (roughing, semi-finish and finish) and the sequence of the three types of operations is always as roughing -> semi-finishing -> finishing. The TSF selection module contains rules for selecting technological solutions of features based on technological constraints such as surface finish. The geometric reasoning module contains rules for determining machining directions. It also has two lower level modules, i.e. reasoning for parent feature and reasoning for child feature. This is discussed in more detail in section 4.4.1. The set-up optimisation rules contain criteria discussed in section 4.3, such as minimum resource elements required for producing the features on a component, minimum number of set-ups on the machine tool, minimum number of tool changes for each set-up, etc. The complete set of rules are documented in Appendix IV.

4.4.1 Some Examples of the Geometric Reasoning Rules

The rules in the geometric reasoning module were designed to reason about the geometric relationships between features such as parent/child relationships so as to determine the feasible machining directions for each feature. A typical geometric reasoning rule is given below.

IF
feature.1 has external access direction EAD.1 and
feature has parent feature feature.2 and
feature.2 has external access direction EAD.2 and
EAD.1 has unit vector V.1 and
EAD.2 has unit vector V.2 and
V1 = V2

THEN
feature.1 can be machined from feature.2 direction on V1.

The rules at level 2 in Figure 4.2 are used to find out the parent–child relationships between features. Since a feature is composed of a group of real faces and imaginary faces, when a face is the real face of a feature and also the imaginary face of another feature (or part of it), there is a parent–child relationship between these two features (see Figure 4.3). Assume feature.1 and feature.2 were linked by
face.1. Based on the entities, attributes and the relationships stored in the component database, the above knowledge can be represented in the form of a production rule as:

IF

- feature.1 'has face' face.1 and
- feature.2 'has face' face.2 and
- face.1 'has parent_face' face.2

THEN

- feature.1 'is machined after' feature.2.

Another ruleset is used to reason about the feasible machining directions for child features:

IF

- part.1 'has raw_material' block_part.1 and
- part.1 consists_of feature.1 and
- feature.1 'has face' face.1 'has ead' vector.1 and
- block_part.1 'has face' face.2 and
- face.1 'has parent_face' face.2 and
face.2 'has normal_vector' vector.2 and
vector.1 'has unit_vector_x' angle_unit_x.1 'has unit_vector_y'
angle_unit_y.1 'has unit_vector_z' angle_unit_z.1 and
vector.2 'has unit_vector_x' angle_unit_x.2 'has unit_vector_y'
angle_unit_y.2 'has unit_vector_z' angle_unit_z.2 and
angle_unit_x.1 = angle_unit_x.2 and
angle_unit_y.1 = angle_unit_y.2 and
angle_unit_z.1 = angle_unit_z.2.

THEN

feature.1 'has access_direction' face.2.

To find out the feasible machining methods for each feature, the following rule in
the set-up optimisation module compares the required technological constraints of a
feature with the machining capability of the mini-plan:

IF

part.1 consists_of feature.1 and
feature.1 'has miniplan' miniplan.1 and
miniplan.1 'can produce' surface_finish.1 and
feature.1 'has highest_accuracy' surface_finish.2 and
surface_finish.2 >= surface_finish.1

THEN

feature.1 'has feasible_plan' miniplan.1.

4.5 The Planning Flowchart

The planning logic is written as procedure programs using Generis command
language. It is based on the rule modules discussed in the above section (4.4) and uses
the part and process data models discussed in chapter 3. The planning logic or
procedure consists of a number of sequenced functional modules, each of which has
an input and output specification. Figure 4.4 illustrates the sequenced planning
functional modules and the input and output specifications of each module.

The procedure programs perform the main planning tasks including making
queries to the database, enabling rules when necessary, manipulating data and other
functions such as writing user defined interfaces, using report forms etc. Figure 4.5
To find feasible Technological Solution of Feature

To reason about the parent-child relationships

To find feasible tool access directions

To establish feature clusters

To optimise feature clusters

To generate and optimise set-ups

| Input: Feature list with constraints | Output: Feasible TSF |

| Input: Feature geometry | Output: Parent-child relationships |

| Input: Parent-child relationships | Output: Feasible feature access direction |

| Input: Feasible access directions + RE list for each feature | Output: Feature clusters |

| Input: Feature clusters | Output: Optimized feature clusters |

| Input: Optimized feature clusters + feasible TSFs | Output: Optimized set-up solutions |

Figure 4.4: The sequenced planning functional modules

shows the data flow between the planning functional modules, the process model, part model and the rulesets in the rulebase. Appendix V lists the main procedure programs used to implement the planning logic.

As seen in Figure 4.5, the planning modules matches the feature data model (FDM) at the left hand side with the process capability model (PCM) at the right hand side and find out the feasible technological solutions and set-ups for machining the
component. The steps for generating feasible and optimised process plans using the prototype system are as follows:

Figure 4.5: The data flow in the prototype planning system
Since the technological solutions at the feature level (TSF) are stored in the manufacturing database as feature state diagrams, the first module at the top of figure 4.5 (module 1) determines all the feasible solutions based on the individual features' technological requirements.

A TSF is divided into states, each of which represents a group of operations that can be obtained from one or more resource elements based on the form generating capabilities of the available machine tools. The set of resources needed to produce each component feature, based on its technological solutions, is determined. The result is stored and transferred to module 4.

The next step is to reason about the geometry of the features in the part data model and the relationships between them and find out the tool access directions to each feature on the component (module 2 and 3).

Taking into account the clustering constraints that may exist due to feature relationships, the objective of module 6 is to group the operations to be used for producing all component features into feasible component PADs, such that the number of component set-ups is minimised.

The final component set-ups are selected by minimising the number of clusters (module 8). This is based on a step by step selection of the cluster containing the maximum number of feature states and then removing those states from the remaining clusters in the other component PADs.

The optimisation criterion is as follows:

(a) minimum number of set-ups selected for machining a component
(b) maximum number of features to be machined at each set-up
(c) minimum number of resource elements used
(d) maximum number of operations at each set-up

4.6 An example

Figure 4.6 is an example component which shows the planning logic and the optimisation criterion of the prototype planning system. The example is a part currently being manufactured by GEC Alsthom Large Machines Ltd. It consists of nine threaded holes, two square steps and four square slots. The square slots are the child features of the steps and the threaded holes are the child features of the square slots. The starting raw material form is a six sided block.
The optimized set-up solutions for the example component is given in Figure 4.7. Two machine tools are needed to machine the component. The machine tool used to square the six external surfaces is a planner. The machine tool used to finish up all the other features is a milling centre (Makino). Two set-ups are required. At the first set-up, two steps are end milled, then the two through slots are end milled. Then drilling tools are used to drill all the holes. Finally, the nine holes are tapped. A similar procedure is used for the second set-up.
<table>
<thead>
<tr>
<th>MACHINE</th>
<th>COMPONENT PAD</th>
<th>FEATURES</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planner</td>
<td>s1_square_boss</td>
<td>flat_surface_1</td>
<td>shaping</td>
</tr>
<tr>
<td></td>
<td>s2_square_boss</td>
<td>flat_surface_2</td>
<td>shaping</td>
</tr>
<tr>
<td></td>
<td>s3_square_boss</td>
<td>flat_surface_3</td>
<td>shaping</td>
</tr>
<tr>
<td></td>
<td>s4_square_boss</td>
<td>flat_surface_4</td>
<td>shaping</td>
</tr>
<tr>
<td></td>
<td>s5_square_boss</td>
<td>flat_surface_5</td>
<td>shaping</td>
</tr>
<tr>
<td></td>
<td>s6_square_boss</td>
<td>flat_surface_6</td>
<td>shaping</td>
</tr>
<tr>
<td>Makino</td>
<td>s1_square_boss</td>
<td>square_step_1</td>
<td>end milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>square_step_2</td>
<td>end milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>through_square_slot_1</td>
<td>end milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>through_square_slot_2</td>
<td>end milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_1</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_2</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_3</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_4</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_5</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_6</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_7</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_8</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_9</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td>s5_square_boss</td>
<td>through_square_slot_3</td>
<td>end milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>through_square_slot_4</td>
<td>end milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_5</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_6</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_7</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_8</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_9</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_10</td>
<td>tapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_11</td>
<td>tapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_12</td>
<td>tapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blind_thread_hole_13</td>
<td>tapping</td>
</tr>
</tbody>
</table>

Figure 4.7: The process planning report about the example component
CHAPTER FIVE

The Integration of the Planning System With a Feature-Based Design System
Chapter 5: The Integration of the Planning System with a Feature-Based Design System

In this chapter, the integration between the prototype process planning system and a prototype feature-based design system will be discussed. An example will also be given to show the data flow from design to process planning and the final results of the planning system.

5.1 Introduction to the feature-based design system

Most contemporary Computer Aided Design (CAD) tools, including two dimensional drafting systems, three dimensional wireframe, surface and solid modellers, have been developed for representing the geometry of components. However, information about component features (e.g. holes, pockets, slots and steps) and their relationships, tolerances, surface finish and so on is missing in the geometric data models. This information is essential for manufacturing planning processes and therefore, the representation of features at the design stage is significant for the integration of CAD and CAM (Computer Aided Manufacture) systems (Dixon, 1988, Shah and Rogers, 1990).

In order to represent the feature information and also utilise the geometric modelling technologies so far developed, a prototype feature based design system, LUT–FBDS (Loughborough University of Technology Feature Based Design System) is currently under development. The output from the FBD is used as an input to the planning system. The structure of the feature-based design system is shown in Figure 5.1.

The system consists of a design by features user interface, a solid modeller (Pafec Imaginer), a feature processor and a post-processor. The design by features interface allows designers to create components from a set of feature primitives; to perform feature edit operations, such as move, rotate and delete; and to define feature relationships, such as compound features, tolerances and parent-child relationships.

The feature taxonomy described in section 3.2.1 (see figure 3.2) has been implemented in the prototype design system. The feature primitives are classified into bosses, pockets, holes, through slots, non-through slots, notches and steps. Each feature class has a number of profile shapes associated with it. Once a feature is created through the design by features interface, a boundary representation (Brep) model is generated by the solid modeller and stored in its database. At the same time, information about the feature such as its dimensional and positional parameters, tolerances and surface finish is output into a preliminary output file. This temporary file is further processed by the feature processor into a feature data model which contains some relevant data.
relating to component manufacturing. The post-processor reads the processed data and reformats it into a file which can be read by the knowledge-based planning system and stored in the planning database. The feature-based design system has been done in a different research project and the post-processor is done in this project. The feature processor is done by the joint effort of the two projects.

The integration tasks are mainly undertaken by the feature processor and the post-processor and this will be discussed in the next section.

5.2 The basic tasks of the system integration

The principal objective of the integration between the design system and the process planning system is to transfer the data output from the design system to the planning system (Generis database). Both the format and the content of the output data from the design system are different from the data stored in the knowledge-based system. For example, before processing, the feature data in the design system includes feature geometry (parameters), tolerances, surface finish,
relationships and material specifications. Whilst, the feature data after processing includes extra information such as External Access Directions (EADs), imaginary faces, parent–child relationships and normal vectors of the faces of the block. Therefore the tasks of achieving the integration between the design and planning systems are as follows:

(a) to derive implicit data which is required explicitly by the planning system, e.g. EADs stored implicitly in the design system as normal vectors of imaginary surfaces are explicitly required by the planning system.

(b) to add information missing in the design system but essential for process planning, e.g. parent faces of every imaginary face of a feature.

(c) to re–format the output data from the design system so that it can read by Generis and stored in its database (in the form of tables)

The first two tasks are undertaken by the feature processor and the data re–formatting is done by the post–processor shown in Figure 5.1.

Some feature processing programs written in C are documented in Appendix VI.

5.3 An Example

Figure 5.2 shows an example component that includes a non–through slot feature (named nslot1), a pocket at the right–hand side (pock1) and a through hole (hole1) at the bottom of the slot. The overall dimensions of the component are (300, 200, 100).

It should be noted that hole1 is a child feature of nslot1 and the bottom face of nslot1 (named nslot1.f5.re) is the parent face of the top face of hole1 (named hole1.f1.im). The block is a boss with six PADs. It is made of six real surfaces (free surface features, each with five EADs). A PAD of real surface represents the normal vector to the real surface ignoring the other four EADs.

After the component is defined using the prototype feature–based design system, the component output from the design system is a list of parameters and attributes. It is documented in Appendix VII. The feature processor then processes the initial data and divides the information into component level and feature level. The component level information contains general specifications of the component and relational data about different features, such as tolerances, compound features and parent–child relationships (tolerances and compound features are not considered in this case study). The processed component level information for the example component is as follows:

COMPONENT NAME: part8
MATERIAL: cast
Figure 5.2: The example component (part8)

STOCK ROUGHNESS: 3.00
STOCK HARDNESS: 120.00
STOCK X_DIM: 300.00
STOCK Y_DIM: 200.00
STOCK Z_DIM: 100.00
NO OF FEATURES: 4
NO OF TOLERANCES: 0
NO OF RELATIONSHIPS: 1

PARENT FEATURE ID: nslot1
PARENT FACE ID: nslot1.f5.re
CHILD FEATURE ID: hole1
CHILD FACE ID: hole1.f1.im
Feature level information contains data about individual features, which includes items such as feature classes, parameters, attributes, locations, orientations and information associated with faces. The processed feature data about the example is as follows:

FEATURE NAME: blk1
FEATURE CLASS: boss
SUB-TYPE: blrec
PARAMETERS: DIM1: 300.000000
         DIM2: 200.000000
         DEPTH: 100.000000
         RADIUS: 0.000000
         FILLET: 0.000000
         ANGLE: 0.000000
         SURFACE FINISH: 3.000000
LOCATION: (0.000000, 0.000000, 0.000000)
LCS X-AXIS: (1.000000, 0.000000, 0.000000)
LCS Y-AXIS: (0.000000, 1.000000, 0.000000)
LCS Z-AXIS: (0.000000, 0.000000, 1.000000)

LOCATION: (0.000000, 0.000000, 0.000000)
LCS X-AXIS: (1.000000, 0.000000, 0.000000)
LCS Y-AXIS: (0.000000, 1.000000, 0.000000)
LCS Z-AXIS: (0.000000, 0.000000, 1.000000)

FACE NAME: blk1.f1.bl
    TYPE: r
    PARENT:
    NORMAL: (0.000000, 0.000000, 1.000000)

FACE NAME: blk1.f2.bl
    TYPE: r
    PARENT:
    NORMAL: (0.000000, 0.000000, -1.000000)

FACE NAME: blk1.f3.bl
    TYPE: r
    PARENT:
NORMAL: (1.000000, 0.000000, 0.000000)

FACE NAME: blk1.f4.bl
TYPE: r
PARENT:
NORMAL: (0.000000, -1.000000, 0.000000)

FACE NAME: blk1.f5.bl
TYPE: r
PARENT:
NORMAL: (-1.000000, 0.000000, 0.000000)

FACE NAME: blk1.f6.bl
TYPE: r
PARENT:
NORMAL: (0.000000, 1.000000, 0.000000)

FEATURE NAME: nslot1
FEATURE CLASS: nslot
SUB-TYPE: nsrec
PARAMETERS:
DIM1: 40.000000
DIM2: 25.000000
DEPTH: 80.000000
RADIUS: 0.000000
FILLET: 0.000000
ANGLE: 0.000000
SURFACE FINISH: 1.500000

LOCATION: (150.000000, 0.000000, 100.000000)
LCS X-AXIS: (1.000000, 0.000000, 0.000000)
LCS Y-AXIS: (0.000000, 1.000000, 0.000000)
LCS Z-AXIS: (0.000000, 0.000000, 1.000000)

FACE NAME: nslot1.f1.im
TYPE: i
PARENT: blk1.f3.bl
NORMAL: (1.000000, 0.000000, 0.000000)
FACE NAME: nsbot1.f2.re
  TYPE: r
  PARENT: 
  NORMAL: (-1.000000, 0.000000, 0.000000)

FACE NAME: nsbot1.f3.im
  TYPE: i
  PARENT: blk1.s1.bl
  NORMAL: (0.000000, 0.000000, 1.000000)

FACE NAME: nsbot1.f4.re
  TYPE: r
  PARENT: 
  NORMAL: (0.000000, 1.000000, 0.000000)

FACE NAME: nsbot1.f5.re
  TYPE: r
  PARENT: 
  NORMAL: (0.000000, 0.000000, -1.000000)

FACE NAME: nsbot1.f6.re
  TYPE: r
  PARENT: 
  NORMAL: (0.000000, -1.000000, 0.000000)

FEATURE NAME: hole1
FEATURE CLASS: hole
SUB-TYPE: bocir
PARAMETERS: DIM1: 0.000000
  DIM2: 0.000000
  DEPTH: 75.000000
  RADIUS: 8.000000
  FILLET: 0.000000
  ANGLE: 0.000000
  SURFACE FINISH: 0.950000

LOCATION: (110.000000, 0.000000, 75.000000)
LCS X-AXIS: (1.000000, 0.000000, 0.000000)
LCS Y-AXIS: (0.000000, 1.000000, 0.000000)
LCS Z-AXIS: (0.000000, 0.000000, 1.000000)

FACE NAME: hole1.f1.lm
   TYPE: i
   PARENT: uslot1.f5.re
   NORMAL: (0.000000, 0.000000, 1.000000)

FACE NAME: hole1.f2.lm
   TYPE: i
   PARENT: blk1.f2.bl
   NORMAL: (0.000000, 0.000000, -1.000000)

FACE NAME: hole1.f3.re
   TYPE: r
   PARENT:
   NORMAL: (0.000000, 0.000000, 0.000000)

FACE NAME: hole1.f4.re
   TYPE: r
   PARENT:
   NORMAL: (0.000000, 0.000000, 0.000000)

FEATURE NAME: pock1
FEATURE CLASS: pocket
SUB-TYPE: pocir
PARAMETERS:
   DIM1: 0.000000
   DIM2: 0.000000
   DEPTH: 50.000000
   RADIUS: 10.000000
   FILLET: 0.000000
   ANGLE: 0.000000
   SURFACE FINISH: 3.800000

   LOCATION: (0.000000, 100.000000, 50.000000)
   LCS X-AXIS: (1.000000, 0.000000, 0.000000)
Once the information is processed by the feature processor, it is re-formatted by the post-processor into the form that can be read by the knowledge-based system Generis. The re-formatted data is documented in Appendix VIII. Generis reads the data and stores it in the form of tables in its database as the part data model (refer to Chapter 3 for part data model). Then the inference engine applies rules already stored in the rulebase to the part data model and the process data model and generates the planned processes for machining the example component (refer to Chapter 4 for the planning logic). The report output from the planning system is shown in Figure 5.3.

The first table at the top of Figure 5.3 gives general specifications about the component. The second table gives the machines selected and the operations for machining features on each
<table>
<thead>
<tr>
<th>Material</th>
<th>cast iron</th>
<th>Blank</th>
<th>blk1</th>
<th>Heat treatment</th>
<th>anneal</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Machine</th>
<th>Component PAD</th>
<th>Features</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>cylinder_grinder</td>
<td>blk1.f1.bl</td>
<td>hole1</td>
<td>hole_grinding</td>
</tr>
<tr>
<td>makino</td>
<td>blk1.f1.bl</td>
<td>nslot1</td>
<td>end_milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hole1</td>
<td>drilling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hole1</td>
<td>reaming</td>
</tr>
<tr>
<td></td>
<td>blk1.f6.bl</td>
<td>pock1</td>
<td>end_milling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>Code_plan</th>
<th>Surface quality</th>
<th>Operation</th>
<th>Tool</th>
<th>Motion set</th>
</tr>
</thead>
<tbody>
<tr>
<td>hole1</td>
<td>hole_plan_5</td>
<td>0.10</td>
<td>drilling</td>
<td>twist_drill</td>
<td>RH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>reaming</td>
<td>reaming_tool</td>
<td>RH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>hole_grinding</td>
<td>grind_wheel</td>
<td>RHC</td>
</tr>
<tr>
<td>hole1</td>
<td>step_plan_2</td>
<td>1.27</td>
<td>end_milling</td>
<td>end_mill</td>
<td>RTT</td>
</tr>
<tr>
<td>pock1</td>
<td>step_plan_2</td>
<td>1.27</td>
<td>end_milling</td>
<td>end_mill</td>
<td>RTT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Planner</th>
<th>cad2</th>
<th>Approved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>17–8–1992</td>
<td>Date</td>
</tr>
</tbody>
</table>

Figure 5.3: The process planning report about the example component machine in the optimised tool access directions. The Third table shows the sequenced set of operations for each feature, cutting tools to be used and motion sets required by each operation.
CHAPTER SIX

Discussions, Conclusions and Future Work
Chapter 6: Discussions, Conclusions and Further Work

6.1 Discussions

Computer Aided Process Planning (CAPP) is a crucial link between CAD and CAM systems and therefore is one of the key components of any successful Computer Integrated Manufacture (CIM) implementation. A CAPP system can be viewed as a decision support system for component manufacture. Three fundamentally inter-related knowledge domains are involved in process plan generation, the component domain (its geometry and technological requirements), processing methods domain (their capabilities and constraints) and the domain of machine tools (their form generating capabilities and technological output).

Process planning is a domain which involves vast amounts of heterogeneous data, knowledge and complicated inferencing processes. The identification, categorisation and structuring of planning information are some of the key issues that have to be addressed during system development. Knowledge representation and integration to facilitate information transformation between these basic domains utilising the facilities of a knowledge-based software environment for process planning is the main thrust of the work reported in this thesis. The project was sponsored by the Science and Engineering Research Council (SERC) Directorate of Applications of Computers in Manufacturing Engineering (ACME), entitled 'Functional Description of Machines, Components and Tools for Generative Process Planning Systems' (Ref. GR/F68232).

The principle objectives of the research undertaken can be summarised as follows:

* to investigate the representation of a feature-based component model and process capability data in process planning systems and to explore the use of a knowledge-based environment in structuring and integrating the diverse information requirements of the planning domain;

* to develop and represent some of the basic geometric reasoning logic needed in process planning and to examine the extent to which KBS facilities can be utilised to simulate human process planning logic;

* to develop a prototype generative process planning system and to link it with a feature-based design system to demonstrate some aspects of automated process planning and CAD/CAM integration.

The basic objectives of the work have been achieved. Below are some comments relating to the main features of the work undertaken.
6.1.1 Structuring of knowledge in the planning domain

Within this work an integrated information structure for planning systems implementation was undertaken. The information flow in the prototype planning system has the following characteristics:

* Separation between component and processing system information.
* The representation of component requirements and processing system capabilities using absolute and constrained types of knowledge.
* The integration of all the system information through planning logic and plan optimisation strategies.

For components, the absolute component knowledge relates to its geometry, topology and connectivity of its constituent features, while the constrained knowledge relates to the technological characteristics of the component form (dimensions, tolerances, surface finish, etc.). The absolute knowledge for the processing system is represented by its form generating functions while its constrained knowledge relates to the system’s technological capabilities.

The advantages of this knowledge structure include: simplifying the planning logic, allowing the decomposition of the planning domain into manageable sub-tasks which can be tackled using appropriate methods for information representation and processing, and producing modular planning systems that are easier to update and maintain.

6.1.2 Feature-based component representation

In the last decade, the concept of using component features for design and manufacturing applications has received much attention and research effort. Features give a higher conceptual meaning to component characteristics by dissecting its geometry into recognisable and meaningful forms. The management, control and use of these groups of basic geometric entities is seen as a practical means of converting designs into manufacturable products. Thus, features are considered to be a suitable communication medium between design and manufacture.

Within the research community however, the term “feature” does not have a clear or exact definition. Features are generally described as “information sets” that refer to aspects of form or other attributes of a part. While this approach gives flexibility in system development, it has serious disadvantages.

For features to succeed in being a true communication medium in CAD/CAM applications, I believe that “features” need to be more strictly defined and that the “feature-based component model” created during the design stage should be capable of conveying sufficient useful information for use in the down-stream manufacturing activities.
This was achieved by developing and representing a component model which has the following new attributes and functionality:

* Capable of unambiguously describing the individual component features using a form features hierarchy
* Capable of describing the structural aspects of component geometry, i.e. how the features are connected together to form the component.
* Many of the component description parameters convey useful information that can be used directly in generating component process plans.

These aspects of the component model, and especially the new concept of feature connectivity, are crucial in an application such as process planning, which places heavy demands on the ability to reason about component geometry in developing the component process plans. Feature connectivity information has been used as the main mechanism for simulating the logic of component set-up determination during planning.

6.1.3 Representation of processing system information

Process planning is an application in which the logic of planning decisions is primarily based on matching component requirements to the capabilities of manufacturing processes and equipment used for their manufacture. Therefore, the description and representation of the capabilities of machining operations and processing machine tools are important issues that need to be addressed during system development.

The work in this area has resulted in the development of quantitative models for describing machining operations by defining them as form generating schemas of relative motions between machined parts and cutting tools. The form generating capabilities of machine tools and cutting tools are also represented using the same set of motion parameters. This has simplified the representation of technological solutions at the feature level and the matching process between component requirements and machining operations and machine tools to be used for its processing.

6.1.4 Planning data reference model

Integrating and structuring all product information into one logical context using models is becoming a powerful methodology that has an important role to play in many manufacturing applications. In such models the information concerning geometry, topology, functionality, equipment and processes can be logically linked to serve as a basis for several manufacturing applications, such as process planning, costing and scheduling, etc.

A "planning data reference model" has been developed (Gindy and Huang 1992b). The model is a logical structure that has evolved during the work to describe my thinking regarding the strat-
egy of process plan generation for prismatic components. The entities and relationships used in the model relate to the main data requirements of the planning domain, the optimisation procedures needed in process plan generation and the integration concepts with other manufacturing systems. In evolving a "reference model", several integration issues that I believe to be critical in developing successful planning systems were given attention. These include:

* Integration of component and processing system information.
* Integration of process planning knowledge with component information produced by CAD systems.
* Flexibility in representing planning logic to deal with company specific issues.
* Inclusion of machining strategies in the development of component process plans.
* Allowing for multi-level optimisation strategies during process plan generation.

The planning data reference model serves as an essential first step in defining the entities and relationships to be used in building the planning system knowledge base during software implementations.

6.1.5 System implementation

The knowledge–based software system "GENERIS" was used as the main software tool for developing the prototype planning system for prismatic components "GENPLAN". The implemented system consists of a user interface to "Generis", an interface to the Imaginer solid modelling CAD system, a knowledge acquisition function, an inference engine, a manufacturing knowledge base and a system output function.

The knowledge acquisition function describes the planning domain in a form that can be interpreted by Generis. The manufacturing database contains the feature–based component description and machining capability data. The planning logic is represented using production rules stored in a rule base. The inference engine executes the manufacturing rule base, constructs queries for extracting information from the database and provides the link to the system’s front end.

The system accepts manual or CAD input of component features and their connectivity information. Starting from the set of faces associated with each component feature and the component connectivity information, the system "reasons" about component geometry to minimise component set–ups. The system is also capable of generating a list of the machining operations for producing component features in each component set–up automatically (an outline process plan).
The current version of "GENPLAN" represents a partial implementation of the full "planning data reference model" developed in this work and deals mainly with the absolute knowledge aspects of components and processing system information. However, implemented software includes sufficient entities, relationships and technological information to demonstrate the main aspects of this research.

6.2 Conclusions

The main conclusions that can be drawn from this research are:

(1) A small scale prototype generative process planning system for prismatic machined components has been developed and tested through a variety of case studies. The output from a prototype feature-based design system has been successfully processed and used as input to the knowledge-based planning system to demonstrate some aspects of automated planning and integration between the design and process planning functions.

(2) The feature-based model for describing component geometry and connectivity has proved to be very useful in automating many of the process planning tasks. Feature connectivity has been successfully used as the main "reasoning" mechanism to simulate human logic in component set-up determination.

(3) A process capability model has been represented and integrated with the part data model through features. The model includes information about machine tools, machining processes, operations and cutting tools. This representation scheme provides an appropriate level of detail which is helping to solve many of the problems faced in developing generative process planning systems.

(4) A rule base has been developed which stores process planning rules as rulesets. The rules are used for selecting and sequencing processes, for selecting feature technological solutions and for optimizing set-ups. The rules in the same ruleset perform a single function. The rulesets are further grouped into different functional modules. The functional modules are identified and invoked by the inference engine.

(5) KBS has proved successful in storing process planning knowledge in the form of facts and rules. Planning knowledge is no longer provided in rigidly programmed or tabular form, but is available in a flexible structured knowledge base. The processing of planning knowledge takes place through the inference engine which deduces the results from the facts and rules available in the knowledge base. This has proved to be an appropriate structure for developing component process plans.

(6) The knowledge base system Generis offers a flexible environment for the development of the prototype generative process planning system GENPLAN. Generis is capable of cap-
turing component and process capability models as entities and relationships as well as the data to instantiate these models during component planning. The ability to repeatedly restructure the models during system development without a heavy programming overhead has proved very useful.

6.3 Further Work

GENPLAN has been developed to act as a testbed (a prototyping tool) for testing appropriate representations for process planning knowledge and not as a fully operational process planning system. As such, the prototype system has proved very successful in pointing the way towards how future systems may be developed. Extensions and enhancements in several areas are needed before the system can be considered as a working generative planning system that can be tested in an industrial application. Chief among these are:

* Increasing the content of the technological data held in the systems knowledge base. This should include: extending the scope of the machine tool and cutting tools database and increasing the variety of technological solutions associated with component features.

* Extending the variety of the initial raw material forms that can be handled by the system. Currently the system can only deal with six sided block type initial raw material forms. The planning logic should be extended to deal with other initial component forms.

* Increasing the capability of the geometric reasoning module to deal with non-orthogonal component features.

* Extensions to the types of component features that can be planned by the system. Currently the planning logic is directed towards producing plans for prismatic components. This should be extended to deal with cylindrical type components.
References
References


DSL: *Introduction to Generis*, 1990

Edinburgh Research report 6: *Application to Real Components*, September 1989, Manufacturing Planning Group, Department of Mechanical Engineering, Edinburgh University


Gindy N. N. Z, Huang S X, Feature-Based Planning Data Model For Generative Planning Systems, 29th International MATADOR Conference, Manchester, UK. April 1992 (a), pp37–44

Gindy N. N. Z and Huang S X, Functional description of machines, components and tools for a generative process planning system, ACME research conference, Brunell University, 1–4 September 1992 (b), pp36 –41


Jared G. E. M: Feature Recognition and Expert Systems for Operating Planning in NC Machining, proceedings of the international Conference on AI Europa. 1986


Shah J J, Bhatnagar A and Hsiao D, Feature mapping and application shell, Computer Aided Engineering, 1988, pp 489 –496


Appendix I

The Table Structure of the Part Data Model
**Appendix I: The Table Structure of the Part Data Model**

Table `parts`

<table>
<thead>
<tr>
<th>Valuedness</th>
<th>Property</th>
<th>Datatype</th>
<th>Status</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi Valued</td>
<td>feature</td>
<td>Generic</td>
<td>On</td>
<td></td>
</tr>
<tr>
<td>Relationship</td>
<td>consists_of</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table `global_part`

<table>
<thead>
<tr>
<th>Valuedness</th>
<th>Property</th>
<th>Datatype</th>
<th>Status</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Valued</td>
<td>description</td>
<td>Text</td>
<td>Mandatory None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>part_id</td>
<td>Text</td>
<td>Mandatory None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>block_part</td>
<td>Entity</td>
<td>Mandatory None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>material_id</td>
<td>Text</td>
<td>Mandatory None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>using heat_treatment</td>
<td>Text</td>
<td>Optional None</td>
<td></td>
</tr>
</tbody>
</table>

Table `features`

<table>
<thead>
<tr>
<th>Valuedness</th>
<th>Property</th>
<th>Datatype</th>
<th>Status</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi Valued</td>
<td></td>
<td>Generic</td>
<td>On</td>
<td></td>
</tr>
<tr>
<td>Relationship</td>
<td>Property</td>
<td>Datatype Status</td>
<td>Index</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>-----------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>has face</td>
<td>face</td>
<td>Entity</td>
<td>Mandatory None</td>
<td></td>
</tr>
<tr>
<td>has ead</td>
<td>vector</td>
<td>Entity</td>
<td>Optional None</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table origins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Subject</td>
</tr>
<tr>
<td>Valuedness</td>
</tr>
<tr>
<td>Generic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Property</th>
<th>Datatype Status</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>along_axis</td>
<td>axis_g</td>
<td>Text</td>
<td>Mandatory None</td>
</tr>
<tr>
<td>has coordinate</td>
<td>coordinate</td>
<td>Decimal</td>
<td>Mandatory None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Subject</td>
</tr>
<tr>
<td>Valuedness</td>
</tr>
<tr>
<td>Generic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Property</th>
<th>Datatype Status</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>has local_origin</td>
<td>origin</td>
<td>Entity</td>
<td>Mandatory None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Subject</td>
</tr>
<tr>
<td>Valuedness</td>
</tr>
<tr>
<td>Generic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Property</th>
<th>Datatype Status</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>has normal_vector</td>
<td>vector</td>
<td>Entity</td>
<td>Mandatory None</td>
</tr>
<tr>
<td>has face_type</td>
<td>type</td>
<td>Entity</td>
<td>Mandatory None</td>
</tr>
<tr>
<td>Relationship</td>
<td>Property</td>
<td>Datatype</td>
<td>Status</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>has unit_vector_x</td>
<td>angle_unit_x</td>
<td>Integer</td>
<td>Mandatory</td>
</tr>
<tr>
<td>has unit_vector_y</td>
<td>angle_unit_y</td>
<td>Integer</td>
<td>Mandatory</td>
</tr>
<tr>
<td>has unit_vector_z</td>
<td>angle_unit_z</td>
<td>Integer</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>

**Table vectors**

**Main Subject vector**

**Valuedness** Multi Valued  **Generic On**
Appendix II

The Machining Process and Machine Tool Capability Data
Appendix II: The Machining Process and Machine Tool Capability Data

**FEATURE CLASS: STEP, SLOT**

<table>
<thead>
<tr>
<th>Code of manufacturing Method</th>
<th>Operation description</th>
<th>Tool</th>
<th>Accuracy of Surface (um)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step_plan_1</td>
<td>End_milling</td>
<td>End_mill</td>
<td>0.81 ~ 0.05</td>
</tr>
<tr>
<td></td>
<td>Grinding</td>
<td>Grind_wheel</td>
<td>(0.13)</td>
</tr>
<tr>
<td>Step_plan_2</td>
<td>End_milling</td>
<td>End_mill</td>
<td>1.27 ~ 1.52</td>
</tr>
</tbody>
</table>

**FEATURE CLASS: FLAT SURFACE**

<table>
<thead>
<tr>
<th>Code of Manufacturing Method</th>
<th>Operation Description</th>
<th>Tool</th>
<th>Accuracy of Surface (um)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surface_plan_1</td>
<td>Shapping</td>
<td>Shapping Tool</td>
<td>2.54</td>
</tr>
<tr>
<td>Surface_plan_2</td>
<td>Face_milling</td>
<td>Face Mill</td>
<td>1.27 ~ 0.76</td>
</tr>
</tbody>
</table>
### FEATURE CLASS: THREAD_HOLE

<table>
<thead>
<tr>
<th>Code of manufacturing Method</th>
<th>Operation Description:</th>
<th>Tool</th>
<th>Accuracy of Surface (um)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hole_plan_1</td>
<td>Drilling, Tapping</td>
<td>Twist drill, Tapping_tool</td>
<td>2.54</td>
</tr>
<tr>
<td>Hole_plan_2</td>
<td>Drilling, Reaming, Tapping</td>
<td>Twist drill, Reaming_tool, Tapping_tool</td>
<td>1.27 - 3.17</td>
</tr>
</tbody>
</table>

### FEATURE CLASS: ROUND HOLE

<table>
<thead>
<tr>
<th>Code of manufacturing Method</th>
<th>Operation Description:</th>
<th>Tool</th>
<th>Accuracy of Surface (um)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hole_plan_3</td>
<td>Drilling</td>
<td>Twist drill</td>
<td>2.54</td>
</tr>
<tr>
<td>Hole_plan_4</td>
<td>Drilling, Reaming</td>
<td>Twist drill, Reaming_tool</td>
<td>1.27</td>
</tr>
<tr>
<td>Hole_plan_5</td>
<td>Drilling, Reaming, Cylinder grinding</td>
<td>Twist drill, Reaming_tool, Grinding wheel</td>
<td>0.10</td>
</tr>
<tr>
<td>Hole_plan_6</td>
<td>Drilling, Counterbore drilling</td>
<td>Twist drill, Counterbore</td>
<td>2.54</td>
</tr>
<tr>
<td>Hole_plan_7</td>
<td>centre drilling</td>
<td>Centre drill</td>
<td>2.54</td>
</tr>
<tr>
<td>Feature</td>
<td>FTS Code</td>
<td>Operations</td>
<td>Tools</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------</td>
<td>------------</td>
<td>------------------</td>
</tr>
<tr>
<td><strong>Blind Round Hole &amp; Through Round Hole</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole_plan_1</td>
<td>Drilling</td>
<td>Twist Drill</td>
<td>Spade Drill</td>
</tr>
<tr>
<td>Hole_plan_2</td>
<td>Drilling</td>
<td>Twist Drill</td>
<td>Spade Drill</td>
</tr>
<tr>
<td></td>
<td>Boring</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole_plan_3</td>
<td>Drilling</td>
<td>Twist Drill</td>
<td>Spade Drill</td>
</tr>
<tr>
<td></td>
<td>Reaming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole_plan_4</td>
<td>Broaching</td>
<td>Form Tool</td>
<td></td>
</tr>
<tr>
<td>Hole_plan_5</td>
<td>Drilling</td>
<td>Twist Drill</td>
<td>Spade Drill</td>
</tr>
<tr>
<td></td>
<td>Reaming</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Honing</td>
<td>Honing Stone</td>
<td></td>
</tr>
<tr>
<td><strong>Counterbore Hole</strong></td>
<td>Hole_plan_6</td>
<td>Drilling</td>
<td>Twist Drill</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spade Drill</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Counterbore</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Counterbore</td>
</tr>
<tr>
<td><strong>Shallow Round Hole</strong></td>
<td>Hole_plan_7</td>
<td>Center Drilling</td>
<td>Center Drill</td>
</tr>
<tr>
<td><strong>Thread Hole</strong></td>
<td>Hole_plan_8</td>
<td>Drilling</td>
<td>Twist Drill</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spade Drill</td>
</tr>
<tr>
<td></td>
<td>Tap</td>
<td>Tapping</td>
<td></td>
</tr>
</tbody>
</table>
### MACHINE TOOL CAPABILITIES:

<table>
<thead>
<tr>
<th>Machine</th>
<th>Resource element</th>
<th>Operation description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makino</td>
<td>Re1_dr</td>
<td>Drilling</td>
</tr>
<tr>
<td></td>
<td>Re2_efm</td>
<td>Reaming</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>End_milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Face_milling</td>
</tr>
<tr>
<td>Surface_grinder</td>
<td>Re4_fg</td>
<td>Surface_grinding</td>
</tr>
<tr>
<td>Cylinder_grinder</td>
<td>Re5_hg</td>
<td>Hole_grinding</td>
</tr>
<tr>
<td>Planner</td>
<td>Re3_p</td>
<td>Shapping</td>
</tr>
</tbody>
</table>
Appendix III

The Table Structure of the Process Capability Model
Appendix III: The Table Structure of the Process Capability Model

<table>
<thead>
<tr>
<th>Table</th>
<th>Miniplans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Subject</td>
<td>Feature</td>
</tr>
<tr>
<td>Valuedness</td>
<td>Multi Valued</td>
</tr>
<tr>
<td>Relationship</td>
<td>Property</td>
</tr>
<tr>
<td>has miniplan</td>
<td>miniplan</td>
</tr>
<tr>
<td>Generic On</td>
<td>Datatype Status Index</td>
</tr>
<tr>
<td>Entity</td>
<td>Mandatory None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Subject</td>
<td>Miniplan</td>
</tr>
<tr>
<td>Valuedness</td>
<td>Multi Valued</td>
</tr>
<tr>
<td>Relationship</td>
<td>Property</td>
</tr>
<tr>
<td>contains</td>
<td>operation</td>
</tr>
<tr>
<td>requires_motionset</td>
<td>motionset</td>
</tr>
<tr>
<td>with_tool</td>
<td>tool</td>
</tr>
<tr>
<td>Generic On</td>
<td>Datatype Status Index</td>
</tr>
<tr>
<td>Entity</td>
<td>Mandatory None</td>
</tr>
<tr>
<td>Text</td>
<td>Optional None</td>
</tr>
<tr>
<td>Entity</td>
<td>Mandatory None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Workshop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Subject</td>
<td>Operation</td>
</tr>
<tr>
<td>Valuedness</td>
<td>Multi Valued</td>
</tr>
<tr>
<td>Relationship</td>
<td>Property</td>
</tr>
<tr>
<td>performs_on</td>
<td>release</td>
</tr>
<tr>
<td>Generic On</td>
<td>Datatype Status Index</td>
</tr>
<tr>
<td>Entity</td>
<td>Mandatory None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Planadd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Subject</td>
<td>Miniplan</td>
</tr>
<tr>
<td>Valuedness</td>
<td>Single Valued</td>
</tr>
<tr>
<td>Generic On</td>
<td></td>
</tr>
</tbody>
</table>
### Table cells

<table>
<thead>
<tr>
<th>Main Subject</th>
<th>Valuedness</th>
<th>Relationship</th>
<th>Property</th>
<th>Datatype</th>
<th>Status</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>machine</td>
<td>Multi Valued</td>
<td>can produce</td>
<td>surface_finish</td>
<td>Decimal</td>
<td>Mandatory</td>
<td>None</td>
</tr>
</tbody>
</table>

### Table toolbank

<table>
<thead>
<tr>
<th>Main Subject</th>
<th>Valuedness</th>
<th>Relationship</th>
<th>Property</th>
<th>Datatype</th>
<th>Status</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>tool</td>
<td>Single Valued</td>
<td>is_constructed_by</td>
<td>resele</td>
<td>Entity</td>
<td>Optional</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Property</th>
<th>Datatype</th>
<th>Status</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>has code of tool</td>
<td>tool_no</td>
<td>Text</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>has tool_description</td>
<td>descrip_tool</td>
<td>Text</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>is_made_of</td>
<td>material</td>
<td>Text</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has tool_life</td>
<td>tool_life</td>
<td>Dec</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>needs_power</td>
<td>power</td>
<td>Dec</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has cutting_edge</td>
<td>cutting_edge</td>
<td>Int</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has recommended_speed</td>
<td>surface_speed</td>
<td>Dec</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has recommended_feed</td>
<td>cutting_feed</td>
<td>Dec</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has max_depth</td>
<td>max_depth</td>
<td>Dec</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has tool_dia</td>
<td>dia_tool</td>
<td>Dec</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has tool_coolant</td>
<td>coolant</td>
<td>Text</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has station_no</td>
<td>station_no</td>
<td>Text</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>used_in</td>
<td>machine</td>
<td>Entity</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has tool_remarks</td>
<td>remarks</td>
<td>Text</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has thread_pitch</td>
<td>thread_pitch</td>
<td>Dec</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>has thread_spec</td>
<td>thread_spec</td>
<td>Text</td>
<td>Optional</td>
<td>None</td>
</tr>
</tbody>
</table>
Appendix IV

The Inference Rules in the Rulebase
Appendix IV: The Inference Rules in the Rulebase

Rules in Inference Ruleset machine_2

1 feature.1 'can machine_from' face.2 if
   feature.1 'has face' face.1 and
   feature.2 'has face' face.2 and
   face.1 'has parent_face' face.2.

Rules in Inference Ruleset select_mc

1 miniplan.1 needs resele.1 if
   feature.1 'has practical_plan' miniplan.1 and
   miniplan.1 contains operation.1 and
   operation.1 performs_on resele.1.

Rules in Inference Ruleset accuracy_machining

1 feature.1 'has highest_accuracy' surface_finish.1 if
   feature.1 'has face' face.1 and
   face.1 'has surface_finish' surface_finish.1.

Rules in Inference Ruleset access_dir

1 feature.1 'has access_direction' face.2 if
   feature.1 'can machine_from' face.1 and
   feature.2 'can machine_from' face.2 and
   block_part.1 'has face' face.2 and
   feature.2 'has face' face.1 and
   feature.1 'has ead' vector.1 and
   face.2 'has normal_vector' vector.2 and
   vector.1 'has unit_vector_x' angle_unit_x.1 'has unit_vector_y' angle_unit_y.1 'has unit_vector_z' angle_unit_z.1 and
   vector.2 'has unit_vector_x' angle_unit_x.2 'has unit_vector_y' angle_unit_y.2 'has unit_vector_z' angle_unit_z.2 and
   angle_unit_x.1 = angle_unit_x.2 and
   angle_unit_y.1 = angle_unit_y.2 and
   angle_unit_z.1 = angle_unit_z.2.

2 feature.1 'has access_direction' face.2 if
   part.1 'has raw_material' block_part.1 and
   part.1 consists_of feature.1 and
   feature.1 'has face' face.1 'has ead' vector.1 and
   block_part.1 'has face' face.2 and
   face.1 'has parent_face' face.2 and
   face.2 'has normal_vector' vector.2 and
   vector.1 'has unit_vector_x' angle_unit_x.1 'has unit_vector_y' angle_unit_y.1 'has unit_vector_z' angle_unit_z.1 and
   vector.2 'has unit_vector_x' angle_unit_x.2 'has unit_vector_y' angle_unit_y.2 'has unit_vector_z' angle_unit_z.2 and
   angle_unit_x.1 = angle_unit_x.2 and
   angle_unit_y.1 = angle_unit_y.2 and
   angle_unit_z.1 = angle_unit_z.2.
Rules in Inference Ruleset tsf_of_feature

1 feature.1 'has mass_plan' miniplan.1 if
   part.1 consists_of feature.1 and
   feature.1 'has miniplan' miniplan.1 and
   miniplan.1 'can produce' surface_finish.1 and
   feature.1 'has highest_accuracy' surface_finish.2 and
   surface_finish.2 > surface_finish.1.

Rules in Inference Ruleset out1_freqmc

1 feature.1 requires resele.1 if
   feature.1 'has feasible' miniplan.1 and
   miniplan.1 contains operation.1 and
   operation.1 performs_on resele.1.

Rules in Action Ruleset fsplan_act

1 feature.1 'has feasible' miniplan.1 if part.1 consists_of feature.1 and feature.1 'has miniplan'
   miniplan.1 and miniplan.1 'can produce' surface_finish.1 and feature.1 'has highest_accuracy'
   surface_finish.2 and surface_finish.2 > surface_finish.1.

Rules in Inference Ruleset final_decision

1 machine.1 set_from face.1 to_produce feature.1 if
   machine.1 is_constructed_by resele.1 and
   resele.1 'has cluster' face.1 passes feature.1.

Rules in Inference Ruleset extrc_decision

1 machine.1 set_from face.1 to_produce feature.1 by_using_of_operation operation.1 if
   machine.1 is_constructed_by resele.1 and
   resele.1 'has cluster' face.1 passes feature.1 and
   feature.1 'has practical_plan' miniplan.1 and
   miniplan.1 contains operation.1 and
   operation.1 performs_on resele.1.

Rules in Inference Ruleset newtsf

1 feature.1 'has practical_plan' miniplan.1 if
   feature.1 'has mass_plan' miniplan.1 and
   feature.1 'has mass_plan' miniplan.2 and
   miniplan.1 'can produce' surface_finish.1 and
   miniplan.2 'can produce' surface_finish.2 and
   surface_finish.1 > surface_finish.2.

Rules in Inference Ruleset parent_rel

1 feature.1 should_after feature.2 if
   feature.1 'has face' face.1 and
   feature.2 'has face' face.2 and
   face.1 'has parent_face' face.2 and
   feature.2 is not square_boss.1.

Rules in Inference Ruleset stranger
1 feature.1 'has access_direction' face.1 if
   feature.1 'has face' face.1 'has ead' vector.1 and
   face.1 'has parent_face' face.2 angle_to face.2.
Appendix V

The Procedure Programs in the Inference Engine to Execute the Planning Logic
DELETE WINDOW ALL
WINDOW NEW
CLEAR

WINDOW NEW "CAPP" SIZE 40 65 AT 5 10
    position 3,10
    message "***************************************************************************"
    position 5,10
    message " PROTOTYPE OF GENERATIVE PROCESS PLANNING SYSTEM "
    position 7,10
    message "***************************************************************************"
    position 27,12
    message " Department of Manufacturing Engineering"
    position 29,18
    message " Loughborough University"
    position 15,26
    message " Welcome!"
    hold 3

CLEAR
POSITION 10,8
MESSAGE:"*---------------------*---------------------*---------------------*"
POSITION 11,8
MESSAGE:"1"
POSITION 12,8
MESSAGE:" This demonstration was designed to apply 1"
POSITION 13,8
MESSAGE:" the feature–based component data model 1"
POSITION 14,8
MESSAGE:" and machining capability data model to 4"
POSITION 15,8
MESSAGE:" realize the automation of process plan– 1"
POSITION 16,8
MESSAGE:" ning. The main application is in a Know– 1"
POSITION 17,8
MESSAGE:" ledge Base Software GENERIS. 1"
POSITION 18,8
MESSAGE"I"
POSITION 19,8
MESSAGE"*---------*---------*---------*---------*
HOLD 5
CLEAR
POSITION 20,20
OPEN cad2
MESSAGE" Progressing ..."
DO int_xy
DO code0_ap6
EXIT
RETURN
CREATE LOCAL fbun INTEGER
CREATE LOCAL p1 INTEGER
CREATE LOCAL tol_loc INTEGER 0
CREATE LOCAL ft_local TEXT
CREATE LOCAL plan_local TEXT
CREATE LOCAL tol_tsf INTEGER 0

DISABLE ALL
ENABLE tsf_of_feature
ENABLE accuracy_machining

DELETE RECORDS IN em_feature1
DO bypass
ENABLE newtsf

FETCH feature has practical_plan miniplan
  LET fbun = $FETCH
  LET tol_loc = $COUNT
LABEL START
  IF p1 <= tol_loc
    LET ft_local = FETCH VALUE ( fbun, p1, 1 )
    LET plan_local = FETCH VALUE ( fbun, p1, 2 )
    FACT ft_local has feasible plan_local
    LET p1 = p1 + 1
    GOTO START
  ENDIF
RETURN
GENERIS COMMAND FILE "c4_ap5"

Application: CAD2

Function: To test whether or not the resource element (glob) is essential for all the features?

Global variable: Min_index_machine

CREATE LOCAL p_local INTEGER 1
CREATE LOCAL point_local INTEGER 1
CREATE LOCAL tol_miniplan INTEGER 0
CREATE LOCAL total_mc INTEGER 0
CREATE LOCAL tol INTEGER 0
CREATE LOCAL mc_miniplan TEXT
CREATE LOCAL fbuf INTEGER
CREATE LOCAL fbuf2 INTEGER
CREATE LOCAL total_feature INTEGER 0
CREATE LOCAL p_feature INTEGER 1
CREATE LOCAL fcode_local TEXT
CREATE LOCAL index_miniplan TEXT

LABEL START
    FETCH records in parts
    LET fbuf2 = $FETCH
    LET total_feature = $COUNT
    IF p_feature <= total_feature
        LET fcode_local = FETCH VALUE ( fbuf2, p_feature, 2 )
        GOTO MAIN
    ENDIF
RETURN

LABEL MAIN

* to consider feasible miniplans for the feature

FETCH NEW feature 'has practical_plan' miniplan for feature = fcode_local
    LET fbuf = $FETCH
    LET tol_miniplan = $COUNT
    WHILE p_local <= tol_miniplan
        LET index_miniplan = FETCH VALUE ( fbuf, p_local, 2 )
    ENDWHILE
LABEL ACTION2
FETCH NEW miniplan needs reuse for miniplan =

index_miniplan

LET total_mc = $COUNT
IF point_local <= total_mc

GOTO ACTION1

ELSE

FACT fcode_local 'has temp' index_miniplan 'need not' min_index_machine

LET point_local = 1
LET p_local = p_local + 1
GOTO MAIN

ENDIF

ENDWHILE

LET p_local = 1
LET p_feature = 1 + p_feature
GOTO START

LABEL ACTION1

FETCH RECORD point_local
LET mc_miniplan = FIELD 2

IF mc_miniplan = min_index_machine

LET point_local = 1
LET p_local = p_local + 1
GOTO MAIN

ELSE

LET tol = tol + 1
LET point_local = point_local + 1
GOTO-ACTION2

ENDIF
Generis Command File: code0_ap6
Application: CAD2

*** ENABLE ACTION RULE fsplan_act to create records in 'em_temp' ***

DELETE RECORDS IN em_temp
DO act0_ap8

DELETE RECORDS IN em_resele_inf
DO code1_ap6
DO c4_ap5
DO code12_ap6

DISABLE ALL
ENABLE outl_freqmc
ENABLE machine_2
ENABLE access_dir

ENABLE stranger

DO code4_ap6
POSITION 22,20
MESSAGE " finish !"
CREATE LOCAL MAX_local INTEGER 0
CREATE LOCAL current_local INTEGER 0
CREATE LOCAL MAX_pad TEXT
CREATE LOCAL ft_local TEXT
CREATE LOCAL fbuf1 INTEGER
CREATE LOCAL tol_ft INTEGER 0
CREATE LOCAL p_local INTEGER 1

CREATE LOCAL tol_face_local INTEGER 0
CREATE LOCAL fbuf_local INTEGER
CREATE LOCAL p_face_local INTEGER 1
CREATE LOCAL pad_boss_local TEXT

LABEL PRECIRCLE

FETCH face has face for block_part
    LET tol_face_local = $COUNT
    LET fbuf_local = $FETCH

IF p_face_local <= tol_face_local
    LET pad_boss_local = FETCH VALUE (fbuf_local, p_face_local, 2)
    GOTO CIRCUS
ENDIF

GOTO FINAL

LABEL CIRCUS

FETCH resele passes feature for face = pad_boss_local for resele = resource_glob

    LET current_local = $COUNT
    IF current_local > MAX_local
        LET MAX_pad = pad_boss_local
        LET MAX_local = current_local
    ENDIF

LET p_face_local = p_face_local + 1
GOTO PRECIRCLE

***Output this result***

LABEL FINAL
DUMP RECORDS IN ftcl FOR resele = resource_glob FOR face = MAX_pad TO ftcl.dp

FETCH resele passes feature for face = MAX_pad for resele = resource_glob
    LET fbuf1 = $FETCH
LET tol_ft = $COUNT

LABEL REDEL
IF p_local <= tol_ft
    LET ft_local = FETCH VALUE ( fbuf1, p_local, 3 )
    DELETE RECORDS in ftcl for feature = ft_local for sele = resource_glob
    LET p_local = p_local + 1
* DISPLAY RECORDS in ftcl
GOTO REDEL
ENDIF
RETURN
Generis Command File "code8_ap5"
Application: cad2
To delete unnecessary resource element

CREATE LOCAL total_waste INTEGER 0
CREATE LOCAL total_ft INTEGER 0
CREATE LOCAL total_re INTEGER 0
CREATE LOCAL p1_ft INTEGER 1
CREATE LOCAL p2_re INTEGER 1
CREATE LOCAL ft_local TEXT
CREATE LOCAL res_ele TEXT
CREATE LOCAL rule_local TEXT plan_select_re5_hg

LABEL s1
fetch feature 'consists_of'
    LET total_ft =$COUNT

IF p1_ft <= total_ft
    FETCH RECORD p1_ft
    LET ft_local = field 2

LABEL s2
fetch feature 'need_not' 'resele' for feature = ft_local
LET total_re =$COUNT
IF p2_re <= total_re
    FETCH RECORD p2_re
    LET res_ele = field 2

    IF res_ele = min_index_machine
        LET total_waste = total_waste + 1
        LET p2_re = 1
        LET p1_ft = p1_ft + 1
        GOTO s1
    ELSE
        LET p2_re = p2_re + 1
        GOTO s2
    ENDIF

    LET p2_re = 1
    LET p1_ft = p1_ft + 1
    GOTO s1
ENDIF
ENDIF

IF total_waste >= total_ft
    DO code2_ap6
ENDIF

RETURN
CREATE LOCAL MAX_local INTEGER 0
CREATE LOCAL current_local INTEGER 0
CREATE LOCAL MAX_pad TEXT
CREATE LOCAL ft_local TEXT
CREATE LOCAL fbuf INTEGER
CREATE LOCAL tol_ft INTEGER 0
CREATE LOCAL p_local INTEGER 1

FETCH resele passes feature for face = "blkl.f1.bl" for resele = resource_glob

LET MAX_local = $COUNT
LET MAX_pad = blkl.f1.bl

FETCH resele passes feature for face = "blkl.f2.bl" for resele = resource_glob
LET current_local = $COUNT

IF current_local > MAX_local
    LET MAX_local = current_local
    LET MAX_pad = blkl.f2.bl
ENDIF

FETCH resele passes feature for face = "blkl.f3.bl" for resele = resource_glob
LET current_local = $COUNT

IF current_local > MAX_local
    LET MAX_local = current_local
    LET MAX_pad = blkl.f3.bl
ENDIF

FETCH resele passes feature for face = "blkl.f4.bl" for resele = resource_glob
LET current_local = $COUNT

IF current_local > MAX_local
    LET MAX_local = current_local
    LET MAX_pad = blkl.f4.bl
ENDIF

FETCH resele passes feature for face = "blkl.f5.bl" for resele = resource_glob
LET current_local = $COUNT

IF current_local > MAX_local
    LET MAX_local = current_local
    LET MAX_pad = blkl.f5.bl
ENDIF
FETCH resele passes feature for face = "blk1.f6.bl" for resele = resource_glob
LET current_local = $COUNT

IF current_local > MAX_local
  LET MAX_local = current_local
  LET MAX_pad = blk1.f6.bl
ENDIF

** for angle feature only*

***FETCH resele passes feature for face = "s1_blind_thread_hole_10" for resele = resele
***esource_glob
***
  LET current_local = $COUNT
***
  IF current_local > MAX_local
    LET MAX_local = current_local
    LET MAX_pad = s1_blind_thread_hole_10
  ***
  ENDIF
** Should output the following result to a application window**
*  MESSAGE "Cluster "resource_glob" + "MAX_pad" has "MAX_local " features. "
****Output this result****
DUMP RECORDS IN ftcl FOR resele = resource_glob FOR face = MAX_pad TO ftcl.dp

FETCH resele passes feature for face = MAX_pad for resele = resource_glob
LET fbuf1 = $FETCH
LET tol_ft = $COUNT
LABEL REDEL
  IF p_local <= tol_ft
    LET ft_local = FETCH VALUE ( fbuf1, p_local, 3 )
    DELETE RECORDS in ftcl for feature = ft_local for resele = resource_glob
    LET p_local = p_local + 1
    GOTO REDEL
  ENDIF
RETURN
LABEL START
CREATE LOCAL fbuf INTEGER
CREATE LOCAL tol_re INTEGER 0
CREATE LOCAL p1_local INTEGER 1
CREATE LOCAL re_local TEXT
CREATE LOCAL reno_local INTEGER 0
CREATE LOCAL min_re TEXT
CREATE LOCAL min_reno INTEGER 0

FETCH resele appears aptimes
  LET tol_re = $COUNT
  IF tol_re = 0
    GOTO STOPEND
  ENDIF

  LET fbuf1 = $FETCH
  LET re_local = FETCH VALUE ( fbuf1, p1_local, 1 )
  LET reno_local = FETCH VALUE ( fbuf1, p1_local, 2 )
  LET min_re = re_local
  LET min_reno = reno_local
  LET p1_local = p1_local + 1

LABEL SOURCE1
IF p1_local <= tol_re
  LET re_local = FETCH VALUE ( fbuf1, p1_local, 1 )
  LET reno_local = FETCH VALUE ( fbuf1, p1_local, 2 )
  IF reno_local = 0
    DELETE RECORDS IN em_resele_inf FOR resele = re_local
    LET p1_local = p1_local + 1
    GOTO SOURCE1
  ENDIF
  IF reno_local < min_reno
    LET min_re = re_local
    LET min_reno = reno_local
  ENDIF

  LET p1_local = p1_local + 1
  GOTO SOURCE1

LET p1_local = p1_local + 1
GOTO SOURCE1
ENDIF

LET min_index_machine = min_re
DELETE RECORDS IN em_resele_inf FOR resele = min_re

do code8_ap5
GOTO START
LABEL STOPEND
RETURN
Generis Command File: code1_ap6
Application: CAD2
To find the minimum no of resource element

CREATE LOCAL fbun INTEGER
CREATE LOCAL fbuf2 INTEGER

CREATE LOCAL total_mc INTEGER
CREATE LOCAL total_plan INTEGER

CREATE LOCAL no_re1 INTEGER 0
CREATE LOCAL no_re2 INTEGER 0
CREATE LOCAL no_re3 INTEGER 0
CREATE LOCAL no_re4 INTEGER 0
CREATE LOCAL no_re5 INTEGER 0
CREATE LOCAL no_re6 INTEGER 0

CREATE LOCAL p2_local INTEGER 1
CREATE LOCAL p1_local INTEGER 1

CREATE LOCAL index_miniplan TEXT
CREATE LOCAL index mc TEXT

DISABLE ALL
ENABLE select_mc
ENABLE tsf_of_feature
ENABLE accuracy_machining
ENABLE newtsf

FETCH feature has practical_plan miniplan
   LET fbun = $FETCH
   LET total_plan = $COUNT
LABEL STAR
   IF p1_local <= total_plan
      LET index_miniplan = FETCH VALUE ( fbun, p1_local, 2 )
      GOTO ACT1
   ELSE
      GOTO CODE8
ENDIF

LABEL ACT1
FETCH NEW miniplan needs reselc for miniplan = index_miniplan
   LET fbuf2 = $FETCH
   LET total_mc = $COUNT
IF p2_local <= total_mc
    LET index_mc = FETCH VALUE ( fbuf2, p2_local, 2 )
    GOTO COMP
ELSE
    LET p2_local = 1
    LET p1_local= p1_local + 1
    GOTO STAR
ENDIF

LABEL COMP
    IF index_mc = re1_dr
        LET no_re1 = no_re1 + 1
    ENDIF
    IF index_mc = re3_p
        LET no_re6 = no_re6 + 1
    ENDIF
    IF index_mc = re2_efm
        LET no_re2 = no_re2 + 1
    ENDIF
    IF index_mc = re4_fg
        LET no_re3 = no_re3 + 1
    ENDIF
    IF index_mc = re5_hg
        LET no_re4 = no_re4 + 1
    ENDIF
    IF index_mc = re6_b
        LET no_re5 = no_re5 + 1
    ENDIF
    LET p2_local = p2_local + 1
    GOTO Act1

LABEL CODE8

FACT re1_dr appears no_re1
FACT re2_efm appears no_re2
FACT re5_hg appears no_re4
FACT re6_b appears no_re5
FACT re4_fg appears no_re3
FACT re3_p appears no_re6

RETURN
CREATE LOCAL prt1_local INTEGER 1
CREATE LOCAL plan_local TEXT
CREATE LOCAL tol_plan INTEGER 0

FETCH miniplan needs resele for resele = min_index_machine
LET tol_plan = $COUNT
LABEL TAG1
IF prt1_local <= tol_plan
    FETCH RECORDS prt1_local
    LET plan_local = FIELD 1
    DELETE RECORDS IN em_temp FOR miniplan = plan_local
    LET prt1_local = prt1_local + 1
GO TO TAG1
ENDIF
RETURN
Generis command file: int_xy
To read the data output from CAD system

1 To clear memory

DELETE RECORDS IN parts
DELETE RECORDS IN global_part
DELETE RECORDS IN features
DELETE RECORDS IN origins
DELETE RECORDS IN positions
DELETE RECORDS IN faces
DELETE RECORDS IN vectors

DO desg.dump
CREATE LOCAL fbun INTEGER
CREATE LOCAL fbuf2 INTEGER

CREATE LOCAL tolf_local INTEGER 1
CREATE LOCAL tolpad_local INTEGER 1

CREATE LOCAL p1 INTEGER 1
CREATE LOCAL p2 INTEGER 1

CREATE LOCAL ft_local TEXT
CREATE LOCAL re_local TEXT
CREATE LOCAL pad_local TEXT

LABEL start
FETCH feature requires resele for resource requires = resource_glob
    LET fbun = $FETCH
    LET tolf_local = $COUNT
IF p1 <= tolf_local
    LET ft_local = FETCH VALUE ( fbun, pl, 1 )
    LET re_local = FETCH VALUE ( fbun, pl, 2 )
    GOTO work
ENDIF
RETURN

LABEL work
FETCH feature has access_direction face for feature = ft_local
    LET fbuf2 = $FETCH
    LET tolpad_local = $count
LABEL action
    IF p2 <= tolpad_local
    LET pad_local = FETCH VALUE ( fbuf2, p2, 2 )
    FACT re_local has cluster pad_local passes ft_local
    LET p2 = p2 + 1
    GOTO action
ENDIF
    LET p2 = 1
    LET pl = pl + 1
    GOTO start
CREATE LOCAL total_local INTEGER

*****to build up the feature cluster*
***by using global(resource_glob) *****

DELETE RECORDS IN ftcl
unix hold screen no -f ftcl.dp 2>/dev/null

LET resource_glob = re3_p
DO code3_ap6
*DISPLAY RECORDS IN ftcl
LABEL REDO7
DO code5_ap6
*DO code5_cad1

FETCH resele passes feature for resele = resource_glob
LET total_local = $COUNT
IF total_local > 0
    GOTO REDO7
ENDIF

LET resource_glob = re1_dr
DO code3_ap6
LABEL REDO2
DO code5_ap6
*DO code5_cad1

FETCH resele passes feature for resele = resource_glob
LET total_local = $COUNT
IF total_local > 0
    GOTO REDO2
ENDIF

LET resource_glob = re2_efm
DO code3_ap6
LABEL REDO
DO code5_ap6
*DO code5_cad1

FETCH resele passes feature for resele = resource_glob
LET total_local = $COUNT
IF total_local > 0
    GOTO REDO
ENDIF
LET resource_glob = re6_b
DO code3_ap6
*DISPLAY RECORDS IN ftcl
LABEL REDO4
DO code5_ap6
*DO code5_cad1

FETCH resele passes feature for resele = resource_glob
LET total_local = $COUNT
IF total_local > 0
    GOTO REDO4
ENDIF

LET resource_glob = re4_fg
DO code3_ap6
LABEL REDO1
DO code5_ap6
*DO code5_cad1

FETCH resele passes feature for resele = resource_glob
LET total_local = $COUNT
IF total_local > 0
    GOTO REDO1
ENDIF

LET resource_glob = re5_hg
DO code3_ap6
LABEL REDO3
DO code5_ap6
*DO code5_cad1

FETCH resele passes feature for resele = resource_glob
LET total_local = $COUNT
IF total_local > 0
    GOTO REDO3
ENDIF

DO ftcl.dp

************************
*For apple9 only***
************************
DISABLE ALL
ENABLE tsf_of_feature
ENABLE accuracy_machining
ENABLE newtsf
ENABLE extre_dicision
UNIX HOLD SCREEN rm -f output1.form 2>/dev/null
WRITE REPORT USING routine to output1.form
GO
LET mach_glob = planner
write report using look3 to output1.form
go
LET mach_glob = makino
write report using look3 to output1.form
go
LET mach_glob = surface_grinder
write report using look3 to output1.form
go
LET mach_glob = cylinder_grinder
write report using look3 to output1.form
go
**********
*The output designed for review
**********
DISPLAY REPORT USING general_setup
GO
UNIX HOLD SCREEN rm -f output.form 2>/dev/null
FETCH RECORDS IN global_part
FETCH RECORD 1
LET part_g = field 1
WRITE REPORT USING routine to output1.form
GO
*WRITE REPORT USING setup2 to output1.form
*GO
WRITE REPORT USING general_setup TO output1.form
GO
WRITE REPORT USING routine to output1.form
GO
WRITE REPORT USING processnew to output1.form
GO
UNIX HOLD SCREEN lpr output.form
UNIX HOLD SCREEN lpr output1.form
clear
POSITION 21,10
MESSAGE Please Fetch Your Result from the Printer Room.
HOLD 3
RETURN
Appendix VI

The Feature Processing Program
Appendix VI: The Feature Process Program

/* Program Name: dog.c;
/* Author: Sue Huang;
/* Date: 19th June. 1992
/* Function: c processor to genera

***************************************************************************/

#include <stdio.h>
#include "data_struct.h"
#include <math.h>

extern FT_LIST *ptr_flist;
extern TOLL_LIST *ptr_tollist;
extern PCL_LIST *ptr_kslist;
extern COMP_SPEC comp_spec;
extern FACES *ptr_flist1;

void write_global_part() /*To form the dump file for table 'global_part'*/
{
    FILE *fp_w;
    FILE *fopen(); /*fopen return a point to a file*/
    int fclose(); /*fclose return an integer*/
    char file_name[80];
    double para1, para2, para3;
    FT_LIST *ptr_flist1;
    FACES *ptr_flist1;
    COMP_SPEC comp_spec;
    printf("output file name:\n");
    scanf ("%s", file_name);
    printf("output file name is : %s", file_name);
    fp_w = fopen (file_name, "w+");
    if (comp_spec.component_id != NULL)
    {
        fprintf(fp_w, "%n");
        fprintf(fp_w, "create records in 'global_part' with 'l' and ':
");
        fprintf(fp_w, "part' \%s\n", comp_spec.component_id);
        fprintf(fp_w, "has description' 'description' 'product of GEC\n");
        fprintf(fp_w, "has raw_material' 'block_part' \%s", ptr_flist1 ->
        feature_id);
        fprintf(fp_w, "has Material' 'material_id' \%s\n",
        comp_spec.material);
        fprintf(fp_w, "use heat_treatment' 'heat_treatment' \n.anneal\n");
        fprintf(fp_w, "\n\n");
    }
    if (ptr_flist != NULL)
    {
        ptr_flist1 = ptr_flist;
    }
}
while (ptr_ftlist != NULL) {
    fprintf (fp_w, "\n\n");
    fprintf (fp_w, "create records in 'parts' with 'I' and ':'\n");
    fprintf (fp_w, "part %s", comp_spec componente_id);
    fprintf (fp_w, "'consists_of' 'feature' %s", ptr_ftlist -> feature_id);
    fprintf (fp_w, "\n");
    ptr_ftlist = ptr_ftlist -> next;
}

/***** to create dump file for all the features in table 'features'****/

ptr_ftlist = ptr_ftlist1;
while (ptr_faceslist != NULL) {
    fprintf (fp_w, "\n\n");
    fprintf (fp_w, "create records in 'features' with 'I' and ':'\n");

/*1st face of the feature***/

fprintf (fp_w, "feature %s", ptr_ftlist -> feature_id);
fprintf (fp_w, "'has face' %s", ptr_faceslist -> face1_id);
if (strcmp ((ptr_faceslist -> face1_type ),"I") == 0)
    {
        fprintf (fp_w, "'has ead' %s", ptr_ftlist -> feature_id);
    }
    fprintf (fp_w, "\n");

/*2nd face of the feature***/

fprintf (fp_w, "feature %s", ptr_ftlist -> feature_id);
fprintf (fp_w, "'has face' %s", ptr_faceslist -> face2_id);
if (strcmp ((ptr_faceslist -> face2_type ),"I") == 0)
    {
        fprintf (fp_w, "'has ead' %s", ptr_ftlist -> feature_id);
    }
    fprintf (fp_w, "\n");

/*3rd face of the feature***/

fprintf (fp_w, "feature %s", ptr_ftlist -> feature_id);
fprintf (fp_w, "'has face' %s", ptr_faceslist -> face3_id);
if (strcmp ((ptr_faceslist -> face3_type ),"I") == 0)
    {
        fprintf (fp_w, "'has ead' %s", ptr_ftlist -> feature_id);
    }
    fprintf (fp_w, "\n");

/*4th face of the feature***/

fprintf (fp_w, "feature %s", ptr_ftlist -> feature_id);
fprintf (fp_w, "'has face' %s", ptr_faceslist -> face4_id);
if (strcmp ((ptr_faceslist -> face4_type ),"I") == 0)
    {
        fprintf (fp_w, "'has ead' %s", ptr_ftlist -> feature_id);
    }
    fprintf (fp_w, "\n");
/** 5th face of the feature **/
   fprintf(fp_w, "feature 1 %s", ptr_ftlist -> feature_id);
   fprintf(fp_w, "has face 'face' %s", ptr_faceslist -> face5_id);
   if (strcmp ((ptr_faceslist->face5_ty ), "") )
   {
     fprintf (fp_w, "has ead' ead' %s", ptr_ftlist -> feature_id);
   }
   fprintf (fp_w, "\n");
/** 6th face of the feature **/
   fprintf(fp_w, "feature 1 %s", ptr_ftlist -> feature_id);
   fprintf(fp_w, "has face 'face' %s", ptr_faceslist -> face6_id);
   if (strcmp ((ptr_faceslist->face6_ty ), "") )
   {
     fprintf (fp_w, "has ead' ead' %s", ptr_ftlist -> feature_id);
   }
   fprintf(fp_w, "\n");
/** create records for table origins **/
   fprintf(fp_w, "\n");
   fprintf(fp_w, "create records in 'origins' with 'I' and ':\n"");
   fprintf(fp_w, "origin log %s", ptr_ftlist -> feature_id);
   fprintf(fp_w, "along_axis 'axis g' l %z\n"");
   fprintf(fp_w, "has coordinate 'coordinate' %s", ptr_ftlist -> locate_z);
   fprintf(fp_w, ":\n");
   fprintf(fp_w, "origin log %s", ptr_ftlist -> feature_id);
   fprintf(fp_w, "along_axis 'axis g' l x\n"");
   fprintf(fp_w, "has coordinate 'coordinate' %s", ptr_ftlist -> locate_x);
   fprintf(fp_w, ":\n");
   fprintf(fp_w, "origin log %s", ptr_ftlist -> feature_id);
   fprintf(fp_w, "along_axis 'axis g' l y\n"");
   fprintf(fp_w, "has coordinate 'coordinate' %s", ptr_ftlist -> locate_y);
   fprintf(fp_w, ":\n");
/** create records in table "positions" ********/
   fprintf(fp_w, " create records in 'positions' with 'I' and ':\n"");
   fprintf(fp_w, "feature l %s", ptr_ftlist -> feature_id);
   fprintf(fp_w, "has local_origin 'origin' %s", ptr_ftlist -> feature_id);
   fprintf(fp_w, ":\n");
/** create records in table "faces" **/
   if (ptr_faceslist != NULL)
     ptr_faceslist1 = ptr_faceslist;
    
    while ( ptr_faceslist != NULL)
    {
      /** create records for 1st face **/
        fprintf(fp_w, "create records in 'faces' with 'I' and ':\n"");
      fprintf(fp_w, "face l %s", ptr_faceslist -> face1_id);
      fprintf(fp_w, " has normal_vector 'vector' %s", ptr_faceslist -> face1_id);
fprintf (fp_w, "'has face_type' 'type' 'face' 1 %s'\n", ptr_faceslist->face1_ty);
fprintf (fp_w, "'has surface_finish' 'surface_finish' 'face' 1 %s'\n", ptr_faceslist->surf_finish);
if ( (ptr_faceslist->parent_face) != NULL )
    fprintf (fp_w, "'has parent_face' 'face' 1 %s'\n", ptr_faceslist->parent_face);
fprintf(fp_w, "\n\n");
if (strncmp((ptr_faceslist->face1_ty), "i") == 0)
{
    fprintf(fp_w,"create records in 'vectors' with '1' and ':'\n");
    fprintf(fp_w,"vector 1\n");
    para1 = acos ((ptr_faceslist->normal_x1) / sqrt ((ptr_faceslist->normal_x1) * (ptr_faceslist->normal_x1) + (ptr_faceslist->normal_y1) * (ptr_faceslist->normal_z1) + (ptr_faceslist->normal_z1) * (ptr_faceslist->normal_z1)));
    para2 = acos ((ptr_faceslist->normal_y1) / sqrt ((ptr_faceslist->normal_y1) * (ptr_faceslist->normal_y1) + (ptr_faceslist->normal_y1) * (ptr_faceslist->normal_y1) + (ptr_faceslist->normal_z1) * (ptr_faceslist->normal_z1)));
    para3 = acos ((ptr_faceslist->normal_z1) / sqrt ((ptr_faceslist->normal_z1) * (ptr_faceslist->normal_z1) + (ptr_faceslist->normal_y1) * (ptr_faceslist->normal_y1) + (ptr_faceslist->normal_z1) * (ptr_faceslist->normal_z1)));
    fprintf(fp_w, "'has unit_vector_x' 'angle_unit_x' 1 %s'\n", para1);
    fprintf(fp_w, "'has unit_vector_y' 'angle_unit_y' 1 %s'\n", para2);
    fprintf(fp_w, "'has unit_vector_z' 'angle_unit_z' 1 %s'\n", para3);
}

/****create records in faces for 2nd face****/

fprintf (fp_w, "create records in 'faces' with '1' and ':'\n");
fprintf (fp_w, "face 1\n");
fprintf (fp_w, "'has normal_vector' 'vector' 'face' 1 %s'\n", ptr_faceslist->face2_id);
fprintf (fp_w, "'has face_type' 'type' 1 %s'\n", ptr_faceslist->face2_ty);
fprintf (fp_w, "'has surface_finish' 'surface_finish' 1 %s'\n", ptr_faceslist->surf_finish);
if ( (ptr_faceslist->parent_face2) != NULL )
    fprintf (fp_w, "'has parent_face' 'face' 1 %s'\n", ptr_faceslist->parent_face2);
fprintf(fp_w, "\n\n");
if (strncmp((ptr_faceslist->face2_ty), "i") == 0)
{
    fprintf(fp_w,"create records in 'vectors' with '1' and ':'\n");
    fprintf(fp_w,"vector 1\n");
    para1 = acos ((ptr_faceslist->normal_x2) / sqrt ((ptr_faceslist->normal_x2) * (ptr_faceslist->normal_x2) + (ptr_faceslist->normal_y2) * (ptr_faceslist->normal_z2) + (ptr_faceslist->normal_z2) * (ptr_faceslist->normal_z2)));
    para2 = acos ((ptr_faceslist->normal_y2) / sqrt ((ptr_faceslist->normal_y2) * (ptr_faceslist->normal_y2) + (ptr_faceslist->normal_y2) * (ptr_faceslist->normal_y2) + (ptr_faceslist->normal_z2) * (ptr_faceslist->normal_z2)));
    para3 = acos ((ptr_faceslist->normal_z2) / sqrt ((ptr_faceslist->normal_z2) * (ptr_faceslist->normal_z2) + (ptr_faceslist->normal_y2) * (ptr_faceslist->normal_y2) + (ptr_faceslist->normal_z2) * (ptr_faceslist->normal_z2)));
}
fprintf(fp_w, "'has unit_vector_x' 'angle_unit_x' 1 %s
", para1);
fprintf(fp_w, "'has unit_vector_y' 'angle_unit_y' 1 %s
", para2);
fprintf(fp_w, "'has unit_vector_z' 'angle_unit_z' 1 %s
", para3);
}

/*********create records for 3rd face*****/

fprint((fp_w, "create records in 'faces' with 'l' and ":\n");
fprintf(fp_w, "face l %s\n", ptr_faceslist -> face3_id);
fprintf(fp_w, "'has normal_vector' 'vector' l vs1_%s\n", ptr_faceslist -> face3_id);
fprintf(fp_w, "'has face_type' 'type' l %s\n", ptr_faceslist -> face3_ty);
fprintf(fp_w, "'has surface_finish' 'surface_finish' l %s\n", ptr_ftlist -> surf_finish);

if ((ptr_faceslist -> parent_face3) != NULL )
  fprintf(fp_w, "'has parent_face' 'face' l %s\n", ptr_faceslist -> parent_face3);
  fprintf(fp_w, "'\n\n");

if (strcmp((ptr_faceslist -> face3_ty), "i") == 0)
  
  fprintf(fp_w,"create records in 'vectors' with 'l' and ":\n");
  fprintf(fp_w, "vector l e3_%s\n", ptr_ftlist -> feature_id);

para1 = acos ((ptr_faceslist -> normal_x3 ) / sqrt ((ptr_faceslist -> normal_x3)* (ptr_faceslist -> normal_x3) + (ptr_faceslist -> normal_y3) * (ptr_faceslist -> normal_z3) * (ptr_faceslist -> normal_z3)));

para2 = acos ((ptr_faceslist -> normal_y3 ) / sqrt ((ptr_faceslist -> normal_x3)* (ptr_faceslist -> normal_x3) + (ptr_faceslist -> normal_y3) * (ptr_faceslist -> normal_z3) * (ptr_faceslist -> normal_z3)));

para3 = acos ((ptr_faceslist -> normal_z3 ) / sqrt ((ptr_faceslist -> normal_x3)* (ptr_faceslist -> normal_x3) + (ptr_faceslist -> normal_y3) * (ptr_faceslist -> normal_z3) * (ptr_faceslist -> normal_z3)));

fprintf(fp_w, "'has unit_vector_x' 'angle_unit_x' 1 %s\n", para1);
fprintf(fp_w, "'has unit_vector_y' 'angle_unit_y' 1 %s\n", para2);
fprintf(fp_w, "'has unit_vector_z' 'angle_unit_z' 1 %s\n", para3);
}

/*********create face 4***********/

fprint((fp_w, "create records in 'faces' with 'l' and ":\n");
fprintf(fp_w, "face l %s\n", ptr_faceslist -> face4_id);
fprintf(fp_w, "'has normal_vector' 'vector' l vs1_%s\n", ptr_faceslist -> face4_id);
fprintf(fp_w, "'has face_type' 'type' l %s\n", ptr_faceslist -> face4_ty);
fprintf(fp_w, "'has surface_finish' 'surface_finish' l %s\n", ptr_ftlist -> surf_finish);

if ((ptr_faceslist -> parent_face4) != NULL )
  fprintf(fp_w, "'has parent_face' 'face' l %s\n", ptr_faceslist -> parent_face4);
  fprintf(fp_w, "'\n\n");

if (strcmp((ptr_faceslist -> face4_ty), "i") == 0)
  
  fprintf(fp_w,"create records in 'vectors' with 'l' and ":\n");
  fprintf(fp_w, "vector l e4_%s\n", ptr_ftlist -> feature_id);

paral = acos ((ptr_faceslist -> normal_x4) / sqrt ((ptr_faceslist -> normal_x4) * (ptr_faceslist -> normal_y4) * (ptr_faceslist -> normal_z4))); 
para2 = acos ((ptr_faceslist -> normal_y4) / sqrt ((ptr_faceslist -> normal_x4) * (ptr_faceslist -> normal_y4) * (ptr_faceslist -> normal_z4))); 
para3 = acos ((ptr_faceslist -> normal_z4) / sqrt ((ptr_faceslist -> normal_x4) * (ptr_faceslist -> normal_y4) * (ptr_faceslist -> normal_z4))); 

fprintf(fp_w, "'has unit_vector_x' 'angle_unit_x' l %f\n", para1); 
fprintf(fp_w, "'has unit_vector_y' 'angle_unit_y' l %f\n", para2); 
fprintf(fp_w, "'has unit_vector_z' 'angle_unit_z' l %f\n", para3);

 rêve create records for 5th faces

fprintf(fp_w, "create records in 'faces' with 'l' and ':'\n"); 
fprintf(fp_w, "face l %s\n", ptr_faceslist -> face5_id); 
fprintf(fp_w, "'has normal_vector' 'vector' l vs5_%s\n", ptr_faceslist -> face5_id); 
fprintf(fp_w, "'has face_type' 'type' l %s\n", ptr_faceslist -> face5_ty); 
fprintf(fp_w, "'has surface_finish' 'surface_finish' l %s\n", ptr_faceslist -> surf_finish);

if ( (ptr_faceslist -> parent_face5) != NULL ) 
  fprintf(fp_w, "'has parent_face' 'face' l %s\n", ptr_faceslist -> parent_face5);
  fprintf(fp_w, "'has parent_face' 'face' l %s\n", ptr_faceslist -> parent_face5);
endif

parm = acos ((ptr_faceslist -> normal_x5) / sqrt ((ptr_faceslist -> normal_x5) * (ptr_faceslist -> normal_y5) * (ptr_faceslist -> normal_z5))); 
para2 = acos ((ptr_faceslist -> normal_y5) / sqrt ((ptr_faceslist -> normal_x5) * (ptr_faceslist -> normal_y5) * (ptr_faceslist -> normal_z5))); 
para3 = acos ((ptr_faceslist -> normal_z5) / sqrt ((ptr_faceslist -> normal_x5) * (ptr_faceslist -> normal_y5) * (ptr_faceslist -> normal_z5))); 

fprintf(fp_w, "'has unit_vector_x' 'angle_unit_x' l %f\n", para1); 
fprintf(fp_w, "'has unit_vector_y' 'angle_unit_y' l %f\n", para2); 
fprintf(fp_w, "'has unit_vector_z' 'angle_unit_z' l %f\n", para3);

 rêve create records for 6th faces

fprintf(fp_w, "create records in 'faces' with 'l' and ':'\n"); 
fprintf(fp_w, "face l %s\n", ptr_faceslist -> face6_id); 
fprintf(fp_w, "'has normal_vector' 'vector' l vs6_%s\n", ptr_faceslist -> face6_id);
fprintf(fp_w, "'has face_type' 'type' 1 %s\n", ptr_faceslist->face6_ty);
fprint(fp_w, "'has surface_finish' 'surface_finish' 1 %s\n", ptr_ftlist->surf_finish);

if ((ptr_faceslist->parent_face) != NULL)
    fprintf(fp_w, "'has parent_face' 'face' 1 %s\n", ptr_faceslist->parent_face);

if (strcmp((ptr_faceslist->face6_ty), "i") == 0)
    {
        fprintf(fp_w, "create records in 'vectors' with 'l' and 'n':\n"
        fprintf(fp_w, "vector %e %e %e\n", ptr_ftlist->feature_id);

        para1 = acos ((ptr_faceslist->normal_x6) / sqrt ((ptr_faceslist->normal_x6) * (ptr_faceslist->normal_x6) + (ptr_faceslist->normal_y6) * (ptr_faceslist->normal_y6) + (ptr_faceslist->normal_z6) * (ptr_faceslist->normal_z6)));

        para2 = acos ((ptr_faceslist->normal_y6) / sqrt ((ptr_faceslist->normal_y6) * (ptr_faceslist->normal_y6) + (ptr_faceslist->normal_x6) * (ptr_faceslist->normal_x6) + (ptr_faceslist->normal_z6) * (ptr_faceslist->normal_z6)));

        para3 = acos ((ptr_faceslist->normal_z6) / sqrt ((ptr_faceslist->normal_z6) * (ptr_faceslist->normal_z6) + (ptr_faceslist->normal_x6) * (ptr_faceslist->normal_x6) + (ptr_faceslist->normal_y6) * (ptr_faceslist->normal_y6)));

        fprintf(fp_w, "'has unit_vector_x' 'angle_unit_x' 1 %f\n", para1);
        fprintf(fp_w, "'has unit_vector_y' 'angle_unit_y' 1 %f\n", para2);
        fprintf(fp_w, "'has unit_vector_z' 'angle_unit_z' 1 %f\n", para3);
    }

/**end of 6th face**/

/**
 **ptr_faceslist = ptr_faceslist -> next;**/
 }

/*****next feature**/

ptr_ftlist = ptr_ftlist -> next;
ptr_faceslist = ptr_faceslist -> next;

} /*end while***/

} /*end viod****/
Appendix VII

The Part Data Output from the Feature-Based Design System
### Appendix VII: The Part Data Output from the Feature–Based Design System

<table>
<thead>
<tr>
<th>Item</th>
<th>Data 1</th>
<th>Data 2</th>
<th>Data 3</th>
<th>Data 4</th>
<th>Data 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>part8 cast</td>
<td>3.000000</td>
<td>120.000000</td>
<td>300.000000</td>
<td>200.000000</td>
<td>100.000000</td>
</tr>
<tr>
<td>blk1 boss</td>
<td>3.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nslot1</td>
<td>40.000000</td>
<td>25.000000</td>
<td>80.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>hole1 holecir</td>
<td>0.000000</td>
<td>0.000000</td>
<td>75.000000</td>
<td>8.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>pocket</td>
<td>0.000000</td>
<td>0.000000</td>
<td>50.000000</td>
<td>10.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>blk1 blk1.f1.bl</td>
<td>0.000000</td>
<td>0.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>nslot1</td>
<td>150.000000</td>
<td>0.000000</td>
<td>100.000000</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>hole1</td>
<td>110.000000</td>
<td>0.000000</td>
<td>75.000000</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>pocket</td>
<td>0.000000</td>
<td>100.000000</td>
<td>50.000000</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>rela1</td>
<td>nslot1</td>
<td>nslot1.f5.re</td>
<td>hole1 hole1.f1.im</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **part8**: Part data for cast 3.000000-120.000000-300.000000-200.000000-100.000000-4-0-1
- **blk1**: Boss block data 3.000000-0.000000
- **nslot1**: Slot data 40.000000-25.000000-80.000000-0.000000-0.000000-0.000000
- **hole1**: Hole data 0.000000-0.000000-75.000000-8.000000-0.000000-0.000000
- **pocket**: Pocket data 0.000000-0.000000-50.000000-10.000000-0.000000-0.000000
- **blk1**: Block data for blk1.f1.bl-0.000000-0.000000-0.000000-1.000000-0.000000-0.000000-1.000000-0.000000-0.000000
- **nslot1**: Slot data for nslot1-150.000000-0.000000-100.000000-1.000000-0.000000-0.000000-0.000000
- **hole1**: Hole data for hole1-110.000000-0.000000-75.000000-1.000000-0.000000-0.000000
- **pocket**: Pocket data for pocket-0.000000-100.000000-50.000000-1.000000-0.000000-0.000000
- **rela1**: Relation data for nslot1-nslot1.f5.re-hole1-hole1.f1.im
Appendix VIII

The Re-formed Part Data Readable by Generis
Appendix VIII: The Re-formatted Part Data
Readable by Generis

create records in 'global_part' with 'l' and ':

part | part
'has description' | 'description' | product_of_GEC
'has manufacturing_code' | 'part_id' | c6207725
'has raw_material' | 'block_part' | blk1
'has material' | 'material_id' | cast
'using heat_treatment' | 'heat_treatment' | anneal

create records in 'parts' with 'l' and ':

part | part
'consists_of' | 'feature' | blk1

: fact blk1 is a boss

create records in 'features' with 'l' and ':

feature | blk1
'has face' | 'face' | blk1.f1.bl

: feature | blk1
'has face' | 'face' | blk1.f2.bl

: feature | blk1
'has face' | 'face' | blk1.f3.bl

: feature | blk1
'has face' | 'face' | blk1.f4.bl

: feature | blk1
'has face' | 'face' | blk1.f5.bl

: feature | blk1
'has face' 'face' | blk1.f6.bl
:
create records in 'origins' with 'l' and ':'
origin | og_blk1
'along_axis' 'axis_g' | z
'has coordinate' 'coordinate' | 0.000000 :
origin | og_blk1
'along_axis' 'axis_g' | x
'has coordinate' 'coordinate' | 0.000000 :
origin | og_blk1
'along_axis' 'axis_g' | y
'has coordinate' 'coordinate' | 0.000000 :
create records in 'positions' with 'l' and ':'
feature | blk1
'has local_origin' 'origin' | og_blk1 :
create records in 'faces' with 'l' and ':'
face | blk1.f1.bl
'has normal_vector' 'vector' | vs1_blk1.f1.bl
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 3.000000
'has parent_face' 'face' |
create records in 'vectors' with 'l' and ':'
vector | vs1_blk1.f1.bl
'has unit_vector_x' 'angle_unit_x' | -90
'has unit_vector_y' 'angle_unit_y' | -90
'has unit_vector_z' 'angle_unit_z' | 0 :
create records in 'faces' with 'l' and ':'
face | blk1.f2.bl
'has normal_vector' 'vector' | vs2_blk1.f2.bl
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 3.000000
'has parent_face' 'face' |

create records in 'vectors' with 'l' and ':'
vector | vs2_blk1.f2.bl
'has unit_vector_x' 'angle_unit_x' | -90
'has unit_vector_y' 'angle_unit_y' | -90
'has unit_vector_z' 'angle_unit_z' | -180

create records in 'faces' with 'l' and ':'
face | blk1.f3.bl
'has normal_vector' 'vector' | vs3_blk1.f3.bl
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 3.000000
'has parent_face' 'face' |

create records in 'vectors' with 'l' and ':'
vector | vs3_blk1.f3.bl
'has unit_vector_x' 'angle_unit_x' | 10
'has unit_vector_y' 'angle_unit_y' | -90
'has unit_vector_z' 'angle_unit_z' | -90

create records in 'faces' with 'l' and ':'
face | blk1.f4.bl
'has normal_vector' 'vector' | vs4_blk1.f1.bl
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 3.000000
'has parent_face' 'face' |

create records in 'vectors' with 'l' and ':'
vector | vs4_blk1.f1.bl
'has unit_vector_x' 'angle_unit_x' | -90
'has unit_vector_y' 'angle_unit_y' | -180
'has unit_vector_z' 'angle_unit_z' | -90

create records in 'faces' with 'l' and ':
face | blkl.f5.bl
'has normal_vector' 'vector' | vs5_blkl.f5.bl
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 3.000000
'has parent_face' 'face' |

create records in 'vectors' with 'l' and ':
vector | vs5_blkl.f5.bl
'has unit_vector_x' 'angle_unit_x' | -180
'has unit_vector_y' 'angle_unit_y' | -90
'has unit_vector_z' 'angle_unit_z' | -90

create records in 'faces' with 'l' and ':
face | blkl.f6.bl
'has normal_vector' 'vector' | vs6_blkl.f6.bl
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 3.000000
'has parent_face' 'face' |

create records in 'vectors' with 'l' and ':
vector | vs6_blkl.f6.bl
'has unit_vector_x' 'angle_unit_x' | -90
'has unit_vector_y' 'angle_unit_y' | 0
'has unit_vector_z' 'angle_unit_z' | -90

create records in 'parts' with 'l' and ':
part | part8
'consists_of' 'feature' | nsrlot1

:
fact nslot1 is a nslot

create records in 'features' with 'I' and ':'
feature | nslot1
'has face' 'face' | nslot1.f1.im
'has ead' 'vector' | e1_nslot1
:
feature | nslot1
'has face' 'face' | nslot1.f2.re
:
feature | nslot1
'has face' 'face' | nslot1.f3.im
'has ead' 'vector' | e3_nslot1
:
feature | nslot1
'has face' 'face' | nslot1.f4.re
:
feature | nslot1
'has face' 'face' | nslot1.f5.re
:
feature | nslot1
'has face' 'face' | nslot1.f6.re
:

create records in 'origins' with 'I' and ':'
origin | og_nslot1
'alongs_axis' 'axis_g' | z
'has coordinate' 'coordinate' | 100.000000
:
origin | og_nslot1
'alongs_axis' 'axis_g' | x
'has coordinate' 'coordinate' | 150.000000
:
origin | og_nslot1
'alongs_axis' 'axis_g' | y
'has coordinate' 'coordinate' | 0.000000
:
create records in 'positions' with 'l' and ':
feature | nslot1
'has local_origin' 'origin' | og_nslot1
:
create records in 'faces' with 'l' and ':
face | nslot1.f1.im
'has normal_vector' 'vector' | vs1_nslot1.f1.im
'has face_type' 'type' | i
'has surface_finish' 'surface_finish' | 1.500000
'has parent_face' 'face' | blk1.f3.bl
:
create records in 'vectors' with 'l' and ':
vector | e1_nslot1
'has unit_vector_x' 'angle_unit_x' | 0
'has unit_vector_y' 'angle_unit_y' | -90
'has unit_vector_z' 'angle_unit_z' | -90
:
create records in 'faces' with 'l' and ':
face | nslot1.f2.re
'has normal_vector' 'vector' | vs2_nslot1.f2.re
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 1.500000
'has parent_face' 'face' |
:
create records in 'vectors' with 'l' and ':
vector | vs2_nslot1.f2.re
'has unit_vector_x' 'angle_unit_x' | 180
'has unit_vector_y' 'angle_unit_y' | 90
'has unit_vector_z' 'angle_unit_z' | 90
:
create records in 'faces' with 'l' and ':
face | nslot1.f3.im
'has normal_vector' 'vector' | vs3_nslot1.f3.im
'has face_type' 'type' | i
'has surface_finish' 'surface_finish' | 1.500000
'has parent_face' 'face' | blk1.f1.bl

create records in 'vectors' with 'l' and ':'
vector | e3_nslot1
'has unit_vector_x' 'angle_unit_x' | 90
'has unit_vector_y' 'angle_unit_y' | 90
'has unit_vector_z' 'angle_unit_z' | 10

create records in 'faces' with 'l' and ':'
face | nslot1.f4.re
'has normal_vector' 'vector' | vs4_nslot1.f1.im
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 1.500000
'has parent_face' 'face' |

create records in 'vectors' with 'l' and ':'
vector | vs4_nslot1.f4.re
'has unit_vector_x' 'angle_unit_x' | 90
'has unit_vector_y' 'angle_unit_y' | 0
'has unit_vector_z' 'angle_unit_z' | 90

create records in 'faces' with 'l' and ':'
face | nslot1.f5.re
'has normal_vector' 'vector' | vs5_nslot1.f5.re
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 1.500000
'has parent_face' 'face' |

create records in 'vectors' with 'l' and ':'
vector | vs5_nslot1.f5.re
'has unit_vector_x' 'angle_unit_x' | 90
'has unit_vector_y' 'angle_unit_y' | 90
'has unit_vector_z' 'angle_unit_z' | 180

create records in 'faces' with 'l' and ':'
face | nslot1.f6.re
 'has normal_vector' 'vector' | vs6_nslot1.f6.re
 'has face_type' 'type' | r
 'has surface_finish' 'surface_finish' | 1.500000
 'has parent_face' 'face' |

create records in 'vectors' with 'l' and ':'
vector | vs6_nslot1.f6.re
 'has unit_vector_x' 'angle_unit_x' | 90
 'has unit_vector_y' 'angle_unit_y' | 180
 'has unit_vector_z' 'angle_unit_z' | 90

create records in 'parts' with 'l' and ':'
part | part8
 'consists_of' 'feature' | hole1

fact hole1 is a hole

create records in 'features' with 'l' and ':'
feature | hole1
 'has face' 'face' | hole1.f1.im
 'has ead' 'vector' | e1_hole1

feature | hole1
 'has face' 'face' | hole1.f2.im
 'has ead' 'vector' | e2_hole1

feature | hole1
 'has face' 'face' | hole1.f3.re

feature | hole1
'has face' 'face' | hole1.f4.re

create records in 'origins' with 'l' and ':'
origin | og_hole1
'along_axis' 'axis_g' | z
'has coordinate' 'coordinate' | 175.000000

origin | og_hole1
'along_axis' 'axis_g' | x
'has coordinate' 'coordinate' | 110.000000

origin | og_hole1
'along_axis' 'axis_g' | y
'has coordinate' 'coordinate' | 10.000000

create records in 'positions' with 'l' and ':'
feature | hole1
'has local_origin' 'origin' | og_hole1

create records in 'faces' with 'l' and ':'
face | hole1.f1.im
'has normal_vector' 'vector' | vs1_hole1.f1.im
'has face_type' 'type' | l
'has surface_finish' 'surface_finish' | 0.950000
'has parent_face' 'face' | nslot1.f5.re

create records in 'vectors' with 'l' and ':'
vector | e1_hole1
'has unit_vector_x' 'angle_unit_x' | -90
'has unit_vector_y' 'angle_unit_y' | -90
'has unit_vector_z' 'angle_unit_z' | 0

create records in 'faces' with 'l' and ':'
face | hole1.f2.im
'has normal_vector' 'vector' | vs2_hole1.f2.im
'has face_type' 'type' | i
'has surface_finish' 'surface_finish' | 0.950000
'has parent_face' 'face' | blk1.f2.bl

create records in 'vectors' with 'l' and ':'
vectors | e2_hole1
'has unit_vector_x' 'angle_unit_x' | -90
'has unit_vector_y' 'angle_unit_y' | -90
'has unit_vector_z' 'angle_unit_z' | -180

create records in 'faces' with 'l' and ':'
faces | hole1.f3.re
'has normal_vector' 'vector' | vs3_hole1.f3.re
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 0.950000
'has parent_face' 'face' | l

create records in 'faces' with 'l' and ':'
faces | hole1.f4.re
'has normal_vector' 'vector' | vs4_hole1.f1.im
'has face_type' 'type' | r
'has surface_finish' 'surface_finish' | 0.950000
'has parent_face' 'face' | l

create records in 'parts' with 'l' and ':'
parts | part8
'consists_of' 'feature' | pock1

fact pock1 is a pocket

create records in 'features' with 'l' and ':'
features | pock1
'has face' 'face' | pock1.f1.im
'has ead' 'vector' | e1_pock1
create records in 'vectors' with 'l' and ':'
vector l e1_pock1
 'has unit_vector_x' 'angle_unit_x' 90
 'has unit_vector_y' 'angle_unit_y' 10
 'has unit_vector_z' 'angle_unit_z' 90
 :

create records in 'faces' with 'l' and ':'
face l pock1.f2.re
 'has normal_vector' 'vector' l vs2_pock1.f2.re
 'has face_type' 'type' r
 'has surface_finish' 'surface_finish' 3.800000
 'has parent_face' 'face'
 :

create records in 'vectors' with 'l' and ':'
vector l vs2_pock1.f2.re
 'has unit_vector_x' 'angle_unit_x' 90
 'has unit_vector_y' 'angle_unit_y' 180
 'has unit_vector_z' 'angle_unit_z' 90
 :

create records in 'faces' with 'l' and ':'
face l pock1.f3.re
 'has normal_vector' 'vector' l vs3_pock1.f3.re
 'has face_type' 'type' r
 'has surface_finish' 'surface_finish' 3.800000
 'has parent_face' 'face'
 :

create records in 'faces' with 'l' and ':'
face l pock1.f4.re
 'has normal_vector' 'vector' l vs4_pock1.f1.im
 'has face_type' 'type' r
 'has surface_finish' 'surface_finish' 3.800000
 'has parent_face' 'face'
 :
feature | pock1
  'has face' | 'face' | pock1.f2.re

feature | pock1
  'has face' | 'face' | pock1.f3.re

feature | pock1
  'has face' | 'face' | pock1.f4.re

create records in 'origins' with 'l' and ':'
origin | log_pock1
  'along_axis' | 'axis_g' | z
  'has coordinate' | 'coordinate' | 50.000000

origin | log_pock1
  'along_axis' | 'axis_g' | x
  'has coordinate' | 'coordinate' | 10.000000

origin | log_pock1
  'along_axis' | 'axis_g' | y
  'has coordinate' | 'coordinate' | 100.000000

create records in 'positions' with 'l' and ':'
feature | pock1
  'has local_origin' | 'origin' | log_pock1

create records in 'faces' with 'l' and ':'
face | pock1.f1.im
  'has normal_vector' | 'vector' | vs1_pock1.f1.im
  'has face_type' | 'type' | i
  'has surface_finish' | 'surface_finish' | 3.800000
  'has parent_face' | 'face' | blk1.f6.bl

:
Appendix IX

The Published Papers Associated With This Research Project


Feature-based component model for computer-aided process planning systems

N. N. Z. GINDY, X. HUANG and T. M. RATCHEV

Abstract. This paper describes a hierarchical structure for form features definition and classification, a planning data 'reference model' and an information structure used for developing process plans for prismatic machined components. Some of the methods used for representing the form-generating capabilities of machine tools are also outlined. These models form the basis for decision making in the prototype process planning system 'GENPLAN'. The paper reports on how the models are being used for reasoning about component geometry during plan generation and process plan optimization strategy at the feature and component levels for the determination of component set-ups.

1. Introduction

Integrating and structuring product information into one logical context by using models is becoming a powerful methodology that has an important role to play in many manufacturing applications (see Sata (1989) for examples). In such models information concerning geometry, topology, functionality, equipment and processes can be logically linked to serve as a basis for several manufacturing applications, such as process planning, costing and scheduling (Krause 1989).

Adopting a modelling approach to integrating the information requirements of an application such as process planning can have many advantages. The models developed can help simplify the planning logic and facilitate the development of structured and modular planning systems which are easy to update and maintain. Realizing such benefits, however, is dependent on the development of an integrated information structure for the planning domain and adequate quantitative models for describing component requirements and the functional capabilities of the processing systems in a compatible and integrated format that can be processed by using computers.

Some of the key issues here are the development of a framework for structuring the planning information, maximizing the information-carrying capacity of the product and processing systems models, and the selection of the generic data and parameters that are relevant for the application domain.

2. Information structure in process planning

Process planning is an example of an application in which the decision logic is primarily based on matching component requirements to the capabilities of the manufacturing processes and equipment used for their production. The relevant component characteristics are normally its geometry, topology, dimensions, material, surface finish, and accuracy requirements. These characteristics represent a set of requirements to be obtained from the machine tools making up the processing system. Machine tool capabilities can be represented by their form generating functions, work envelope and their technological output in terms of accuracy and surface finish, etc.

An information structure, based on absolute and constrained types of knowledge for components and processing systems, used in our prototype process planning system 'GENPLAN' is shown in Figure 1. At the planning stage, consideration of component requirements and processing system capabilities takes place at two basic levels. At the highest level, the absolute knowledge is initially investigated. The objective here is to establish, on a feature by feature basis, if the component can be produced by using the available system capabilities. The investigation is performed regardless of component size, material, accuracy, cost, etc. This is the first attempt at finding a match between component geometry and topology and the form-generating capabilities of the available machine tools. The output from this investigation is the possible processing methods capable of producing component geometry.
The investigation then moves to the lower level of considering the feasible matches with respect to their constrained knowledge. Here, the system is trying to take account of the technological requirements of the component and technological constraints of the processing system. The feasible solutions at the feature and component levels are optimized and an outline process plan is generated.

3. Feature-based component model

Recently, the concept of using component features for design and manufacturing applications has received much attention and research effort (see Shah et al. (1988) for examples). Features give a higher conceptual meaning to component characteristics by dissecting component geometry into recognizable and meaningful forms. The management, control and use of these groups of basic geometric entities is seen as a practical means of converting designs into manufacturable products. Thus, features are considered to be a suitable communication medium between design and manufacture (Butterfield et al. 1987).

For features to succeed in being a true communication medium in CAD/CAM applications, and to aid their integration, the 'feature-based component model' created during the design stage should be capable of conveying sufficient useful information for use in the downstream manufacturing activities. To achieve this, the component model should be capable of describing not only the individual features and their attributes, but also the structural aspects of component geometry, i.e. how the features are connected together to form the component. The feature connectivity aspects of the component model are crucial in an application such as process planning, which places heavy demands on the ability to reason about component geometry in developing the component process plans.

In general, a form feature can be defined as a geometric entity that represents a shape pattern that has some significance. In this work, form features are considered as predefined geometric primitives that can be used as component building blocks during the design stage. In GENPLAN, features are treated as component regions that have significance in the context of machining operations. Examples of such machining features are HOLE, STEP, SLOT, NOTCH, etc, each of which needs to be described and classified uniquely within the component model.
3.1. Form feature taxonomy

Component form features are treated as volumes enveloped by entry/exit and depth boundaries. A component feature is described by determining the number of imaginary faces included in its definition, i.e. its external access directions (EADs) (0, 1, 2, 3, 4, 5, 6), its boundary type (open, closed), its exit boundary status (through, not through) and its form variation with respect to its depth axis.

Based on their geometric attributes, features are uniquely classified into categories, classes and subclasses, which may be followed by secondary forms that fully describe the component features (Gindy 1989).

An overview of the main divisions in the form feature taxonomy is shown in Figure 2 and examples of individual component features are shown in Figure 3.

3.2. Feature connectivity

Form features are geometric entities that relate to local component regions. For reasoning about component geometry during process planning, it is important to capture the structural aspects of component geometry, i.e. to describe the relationships (connectivity) that determine how the component is constructed from its constituent features.

Feature connectivity can be regarded as a component representation based on a type of parent–child relationship between its constituent features. Connectivity is specified by using inheritance rules between the real and imaginary surfaces of the component. A free surface of the component is the parent of all the features that have imaginary surfaces within its boundary. A child feature can become a parent of all the adjacent features that have
Component model for CAPP systems

imaginary surfaces contained within the boundaries of any of its real surfaces. Features connectivity information is shown as a directed graph in Figure 3.

4. Process planning reference model

The planning data 'reference model' which describes the logic of process plan generation is shown in Figure 4. In this model, a component is treated as a group of connected features that are to be machined by using the available processing system capabilities. Feature connectivity decides how the features relate to the component's potential approach directions (PADs) and feature relationships are the geometric and technological relationships that define the constraints that may exist on the formation of feature clusters when machining from component PADs.

Each feature has its transition diagram (FTD), which is a representation of all the technological solutions at the feature level (TSFs), as ordered sets of operations capable of producing the feature form and satisfying its technological requirements.

1. Processing system capability model

As shown pictorially in Figure 5, a processing system machining facility can be viewed as representing the collective capabilities of the set of machine tools and tools available for component processing. A machine tool's structural configuration can be described as kinematic chains of elementary motions (translations and rotations) and rules for grouping these motions to constitute the machine form-generating system. The machine form-generating capabilities can then be represented as sets of elementary form-generating schemas, i.e. tools of specific geometry and relative motions between each tool and the workpiece (Portman 1981).

The machine's structural configuration decides the possibilities for linking the form-generating schemas during the processing of components. Analysing the elementary schemas and their relationships can lead to a representation of the operations the machine is capable of performing, which can be linked to its feature-producing capabilities.

The processing capabilities of a machining facility can be represented by a set of resource elements (REs) where each RE is a set of operations that can be obtained from one or more machine tools.

The breakdown of machine tools into resource elements allows process plan generation to be based on less than a whole-machine basis. It therefore provides better links with the machine shop scheduling system and helps to minimize the ad hoc re-planning of component on the shop floor. It also allows greater flexibility to take into account any company-specific machining strategy (flow line, machining centre, conventional machine tools, etc.) that may be required for organizing manufacturing activities.
6. Component set-up determination

The determination of the final component set-ups involves a two-stage optimization procedure. In the first stage, optimization is performed at the feature level, and in the second, the component as a whole is considered. An overview of the SET-UP determination strategy adopted in GENPLAN is shown in Figure 5.

6.1. Feature-level optimization

The objective here is to select a single TSF from among the equally feasible equally weighted alternatives for each component feature. All the resources needed for the operations required to produce all component features are considered.

Based on the feature's technological attributes, its state diagram is parsed to determine all the feasible solutions for feature processing. A TSF is divided into states, each of which can be matched to one or more resource elements obtained from the analysis of the form-generating capabilities of the available machine tools.

The set of resources needed to produce each component feature, based on its technological solutions, is determined. The total resource set to produce the component is then minimized and the appropriate TSF is attached to each component feature.

6.2. Component-level optimization

This stage involves some reasoning about component geometry. Based on feature connectivity, it is possible to
Figure 5. Set-up determination strategy in GENPLAN.
N. N. Z. Gindy et al.

determine all the component potential approach directions (PADs), the features that can be machined from each direction and the set of operations attached to each component feature. This represents the maximum number of potential component set-ups and the work content to be performed from each component PAD before any optimization.

Determination of the component set-ups can be considered initially as a process of clustering (Chang 1990) of feature states in order to effect the required machining operations using common resources from common component approach directions (PADs). All feature states are grouped in clusters in which feature states can appear in more than one cluster in different component PADs.

The final component set-ups are selected by minimizing the number of clusters. This is based on step-by-step selection of the cluster containing the maximum number of feature states and then removing those states from the remaining clusters in the other component PADs. It is important that clustering is performed while observing the precedence relationships that may exist between component features. It is almost impossible, however, to predefine all the possible feature relationships (centricity, parallelism, perpendicularity, etc) and the implications that multi-relationships between features may have on the component process plan. Many of the issues involved in making these decisions are company-specific and experience-based.

The important factor as far as system implementation is concerned, however, is that the planning system should be able to perform feature clustering within the constraints that feature relationships may impose on feature grouping. The currently implemented version of GENPLAN recognizes sequential, parallel and same set-up as constraints to be observed during feature clustering.

7. Conclusions

Modelling of component characteristics and processing system capabilities provides a good basis for solving many of the tasks involved in process planning. The process planning reference model used for plan generation in GENPLAN, described in this paper, provides an appropriate framework not only for process plan generation, but also for linking the plans to a machining strategy and machine-shop scheduling/loading systems.

The feature-based model for describing component geometry and connectivity, and the model for representing the form-generating capabilities of machine tools outlined in this paper, are proving very useful for geometric reasoning and SET-UP determination tasks in our computer-aided process planning system.

Acknowledgement

The research reported here is supported by grants provided by the ACME Directorate of the SERC. Their financial contribution is gratefully acknowledged.

References


FEATURE-BASED PLANNING DATA MODEL FOR GENERATIVE PLANNING SYSTEMS

N.N.Z. GINDY AND S.X. HUANG
Loughborough University of Technology

SUMMARY
The paper outlines some of the authors' ongoing research in developing generative process planning systems. It reports on a reference model used as the basis for generating process plans for prismatic machined components. A feature-based model for describing component geometry and connectivity is also outlined, as are methods used for representing the capabilities of machine tools. These models form the basis for decision making in the prototype process planning system GENPLAN. The paper also discusses how some of the planning domain knowledge is represented using the knowledge-based software GENERIS that is used for planning system development.

INTRODUCTION
Traditionally, process planning has been an experience-based activity that is performed by human planners and, more recently, with some help from computers. A large number of Computer Aided Process Planning (CAPP) systems have been developed, both in research laboratories and by commercial software vendors. Early CAPP systems are mainly variant systems which are based on Group Technology principles. A new component process plan is created by retrieving and editing an existing master plan created by a human planner for similar parts /1,2/.

More recent research efforts have concentrated on developing generative process planning systems. Such systems attempt to create component process plans from information available in a manufacturing database with little human intervention /3,4,5/. The amount and variety of data and knowledge involved in developing generative process planning systems is very large, parts have to be described, and manufacturing knowledge and decision logic have been captured and applied in order to simulate the decision-making process of a process planner.

Integrating and structuring all product information into one logical context using models is becoming a powerful methodology that has an important role to play in many manufacturing applications /6/. In such models the information concerning geometry, topology, functionality, equipment, and processes can be logically linked to serve as a basis for several manufacturing applications, such as process planning, costing and scheduling /7/.

Several schemes have been used for knowledge representation in planning systems. Adopting a modelling approach, however, can have many advantages. The developed models can help simplify the planning logic, and facilitate knowledge representation and the development of structured and modular systems which are easy to update and maintain. Realising such benefits, however, is dependent on the development of an integrated information structure for the planning domain and adequate quantitative models for describing component requirements and the functional capabilities of the processing systems in a compatible and integrated format.

Some of the key issues here are the framework to be used for structuring the planning information, maximising the information carrying capacity of the product and processing system models, and the selection of the generic data and parameters that are relevant for the application tasks.

The following sections outline our approach to process planning and report on some of our findings during the development of our prototype process planning system GENPLAN.

PROCESS PLANNING REFERENCE MODEL
The decision logic in process planning is primarily based on finding an appropriate match between the requirements of components and the capabilities of the manufacturing processes and equipment that are to be used for their production.

The relevant component characteristics are normally its geometry, topology, size, material, surface finish and accuracy requirements. The overall processing system capabilities are dependent on the capabilities of the individual machine tools making up the system in terms of their form generating functions and their technological output levels.

Recently, the concept of using component features for design and manufacturing applications has received much attention and research effort /8/. Features give a higher conceptual meaning to
component characteristics by dissecting component geometry into recognisable and meaningful forms. The management, control and use of these groups of basic geometric entities is seen as a practical means of converting designs into manufacturable products. Thus, features are considered to be the communication medium between design and manufacture /9/.

For features to succeed in being a true communication medium in CAD/CAM applications, and to aid their integration, the ‘feature-based component model” created during the design stage should be capable of conveying sufficient useful information for use in the down-stream manufacturing activities. To achieve this the component model should be capable of describing not only the individual features and their attributes, but also the structural aspects of component geometry, i.e. how the features are connected together to form the component. The feature connectivity aspects of the component model are crucial in an application such as process planning, which places heavy demands on the ability to reason about component geometry in developing the component process plans.

Figure (1) shows the process planning reference model used in our prototype process planning system GENPLAN. In this model a component is considered as a group of connected features which are to be machined using a set of available resources that represent the processing system capabilities.

---

**Figure 1 Process Planning Reference Model in GENPLAN**
Feature Taxonomy

Component form features are treated as volumes enveloped by entry/exit and depth boundaries. Feature geometry is described by deciding on its external access directions (0,1,2,3,4,5,6), i.e. the number of imaginary faces included in the feature definition, its boundary type (open, closed), its exit boundary status (through, not through), and the variation of its form with respect to its depth axis. Based on their geometric attributes, features are uniquely classified into categories, classes and subclasses which may be followed by secondary forms to fully describe compound features [10].

Feature Connectivity

Form features are geometric entities which relate to local component regions. It is necessary, therefore, to describe their relationships (connectivity) which determine how the component is constructed from its constituent features. The feature connectivity aspect is represented by two types of links: external access direction links for relating individual features to the basic component directions; and inheritance links that relate adjacent features, with some features becoming parents to other features.

An example of a feature-based component model and its connectivity information is shown in Figure (2).
Processing System Capabilities

Machine tools can be considered as kinematic chains of elementary motions (translations and rotations) and a set of rules that can be applied for grouping these motions to constitute feasible sets of form generating schemas. Adopting this approach, the form generating capabilities of a specific machine structural configuration can be represented as sets of elementary form generating schemas, i.e., tools of specific geometry and relative motions between each tool and the workpiece /11/.

The machine structural configuration decides the possibilities for linking the form generating schemas during the processing of components. Analysing the elementary schemas and their relationships can lead to a representation of the features the machine is capable of producing and possibilities for the simultaneous/parallel processing of component features.

Planning Strategy

During planning, consideration of component requirements and processing system capabilities takes place at two basic levels. At the highest level, the objective is to establish, on a feature by feature basis, if the component can be geometrically produced using the available system capabilities. The investigation is performed regardless of component size, material, accuracy, cost, etc. This is the first attempt at finding a match between component geometry and topology and the form generating capabilities of the available machine tools. The output of this investigation is all the possible processing methods capable of producing component geometry.

The investigation can then move to the lower level of considering the feasible matches with respect to their constrained knowledge (material, surface finish, size etc). Here, the planning system is trying to take into account the technological requirements of the component and technological limits of the processing system.

The determination of the final component set-ups involves a two stage optimisation procedure, as outlined below.

Optimisation at the Feature Level

At this stage all the resources needed for the operations required to produce all component features are considered. The first step to be performed here is to allocate a specific set of manufacturing operations for producing each of the component features. This is what is termed the technological solution at the feature level (TSF). A TSF is an ordered set of operations that satisfy the feature geometry and technological requirements.

TSFs are stored in the manufacturing database according to the feature transition diagrams (FTD), which represent all the available operation sets capable of producing the feature (see Figure 3). Based on the individual feature technological requirements, the manufacturing database is queried to determine all the equally weighted solutions for producing a feature with the desired technological attributes.

Each operation in the TSF is described as a form generating schema, i.e., a set of motions and tools for performing the operation. To facilitate the matching between feature requirements and machine tool capabilities, a TSF is divided into states (operation groups), each of which can be matched to one or more resource elements established from an analysis of the form generating capabilities of the available machine tools.

The set of resources needed to produce each component feature, based on its technological solutions, is determined. The total resource set to produce the component is then minimised and the appropriate TSFs are attached to the component feature (see Figure 2).

Optimisation at the Component Level

This stage involves some reasoning about component geometry. Based on feature connectivity (see Figure 2), it is possible to determine all the component potential approach directions (PADs), the features that can be machined from each direction and, as a result of the first stage optimisation, the set of operations attached to each component feature. This represents the maximum number of potential component set-ups and the work content to be performed from each component PAD before any optimisation.

Determination of the component set-ups can be considered initially as a process of clustering /12/ of feature states for machining using common resources from common component potential approach directions (PADs). All feature states are grouped in clusters in which a feature state can appear in more than one cluster in different component PADs.

The final component set-ups are selected by minimising the number of clusters. This is based on step by step selection of the cluster containing the maximum number of feature states and then removing those states from the remaining clusters in the other component PADs.

It is important that clustering is performed while observing the precedence relationships that may exist between component features. It is almost impossible, however, to predefined the implications that multi-relationships (concentricity, parallelism, perpendicularity, etc) between features may have on the component process plan. Many of the issues...
involved in making these decisions are company specific and experience based.

The important factor as far as system implementation is concerned, however, is that the planning system should be able to perform feature clustering within the constraints that feature relationships may impose on feature grouping. The currently implemented version of GENPLAN recognises sequential, parallel and same set-up as constraints to be observed during feature clustering.

![Feature Transition Diagram (FTD)](image)

**Figure 3 Technological Solutions at Feature Level**

**SYSTEM IMPLEMENTATION**

The knowledge-based system GENERIS /13/ was selected as the development tool for GENPLAN. The implemented system consists of a user interface, a knowledge acquisition function, an inference engine, an explanation of output function, and the knowledge base which includes a manufacturing database and production rulebase. The knowledge acquisition function describes the planning domain in a form that can be interpreted by GENERIS. The manufacturing database contains the feature based component data and the machining capability data.

The planning logic is represented as production rules in the rule base to be executed by the inference engine. The control program in the inference engine, which constructs queries for extracting information from the database and linking the database to the front end, is written in GENERIS high level language.
The Manufacturing Database

Figure (4) shows a partial view of the entity-relationship diagram used for designing the manufacturing database. The rectangles represent the entity sets, the circles represent attributes and the diamonds represent the relationships. Both part representation and machining knowledge representation are included in the entity-relationship diagram.

The entities, attributes and relationships used in the system are selected to achieve a level of detail sufficient for solving the various planning tasks. For example, to deduce information about component connectivity for set-up determination, it becomes necessary to define a local co-ordinate frame for each feature in which real and imaginary faces are treated as separate attributes. Figure (5) shows part of the relational tables used for data input into the database.

The Rulebase

The rule base contains the rule sets which capture the planning logic and reasoning processes involved in many of the planning tasks. These are included as separate rule sets for reasoning about component geometry, the selection of machining directions, optimisation procedures, etc.

As an example, the knowledge to be used for defining the inheritance relationships between features (parent-child relationships) to be used later by the system for set-up determination can be stated as:

Each feature can have any number of real faces and a fixed number of imaginary faces defined by its classification class. An imaginary face is an access direction through which a tool can pass to machine the feature and is linked to the real faces of adjacent features. Any real surface is considered to be the parent of any number of imaginary surfaces (child) whose boundary is contained within the boundary of that real surface. Through this knowledge, features are considered to be parent/child of other features.

Based on the entities, attributes and the relationships stored in the component database, the above knowledge can be represented in the form of a production rule as:

IF
feature.1 has real_surface face.1 and
feature.2 has imaginary_surface face.1
THEN
has_parent feature.2.

Similar format production rules are used to complete the reasoning about component geometry. For example, the following rule is used to determine the feasibility of a feature access direction:
IF
part.1 'has feature' feature.1 'has feature_code' feature_code.1 and
part.1 'has feature' feature.2 'has feature_code' feature_code.2 and
feature.1 'has parent' feature_code.2 and
feature.2 'has ead_feature' axone.1 and
feature.1 'has ead_feature' axone.2 and
axone.1 for_feature feature_code.2 'has angle_x' angle_x.1 and
angle_y.1 'has angle_x' angle_x.1 and
angle_z.1 'has angle_x' angle_x.1 and
axone.2 for_feature feature_code.1 'has angle_x' angle_x.2 and
angle_y.2 'has angle_x' angle_x.2 and
angle_z.2 'has angle_x' angle_x.2
THEN
feature_code.1 'may be machined_from' feature_code.2 with_dir axone.2.

The facts stored initially in the manufacturing database about the component, in addition to any deduced facts that may result from applying the production rule sets, form the basis for generating the component's process plan.

The currently implemented version of GENPLAN is capable of producing an outline process plan for prismatic components with non-orthogonal features to the component co-ordinate frame, taking into account component geometry and connectivity.

CONCLUSIONS

The feature-based model for describing component geometry and connectivity, the model for representing the form generating capabilities of machine tools, together with the process planning reference model outlined in this paper, are providing an appropriate level of detail which is helping to solve many of the problems faced in developing generative process planning systems.

Process planning is a domain which involves many complicated inferencing processes which are normally performed by human planners. Adopting a knowledge-based approach which separates the "knowledge" from how this knowledge is to be used is proving very useful in automating many of the tasks involved in process planning.

The system is currently limited to producing outline process plans based on component geometry and feature connectivity. A new version is under development to include more of the technological rules involved in the planning domain.

ACKNOWLEDGEMENTS

The research reported here is supported by grants provided by the ACME Directorate of SERC, their financial contribution is gratefully acknowledged.
REFERENCES


Feature-Based Component Model for Computer Aided Process Planning Systems

Dr N N Z Gindy, X Huang and Dr T M Ratchev

Department of Manufacturing Engineering, University of Technology, Loughborough, Leicestershire LE11 3TU, United Kingdom

Abstract

This paper reports on some of our ongoing research in developing generative process planning systems. A hierarchical structure for form features definition and classification, and an overall reference model for developing process plans for machined components, are presented. Some of the methods used for representing the capabilities of machine tools are also presented. These models form the basis for decision making in the prototype process planning system GENPLAN. As an example, the paper reports on how the models are being used for reasoning about component geometry and the optimisation strategies used for the determination of component set-ups.

1. INTRODUCTION

Integrating and structuring all product information into one logical context using models is becoming a powerful methodology that has an important role to play in many manufacturing applications [1]. In such models the information concerning geometry, topology, functionality, equipment, and processes can be logically linked to serve as a basis for several manufacturing applications, such as process planning, costing, and scheduling [2].

Adopting a modelling approach to integrating the information requirements of an application such as process planning can have many advantages. The developed models can help simplify the planning logic and to facilitate the development of structured and modular planning systems which are easy to update and maintain.

Realising such benefits, however, is dependent on the development of an integrated information structure for the planning domain and adequate quantitative models for describing component requirements and the functional capabilities of the processing systems in a compatible and integrated format that can be processed using computers.

Some of the key issues here are the framework to be used for structuring the planning information, maximising the information carrying capacity of the product and processing system models, and the selection of the generic data and parameters that are relevant for the application domain.

2. INFORMATION STRUCTURE IN PROCESS PLANNING

Process planning is an example of an application in which the decision logic is primarily based on matching component requirements to the capabilities of the manufacturing processes and equipment to be used for their production.

The relevant component characteristics are normally its geometry, topology, size, material, surface finish and accuracy requirements. These characteristics represent a set of requirements to be obtained from the machine tools making up the processing system. Machine tool capabilities can be represented by their form generating functions, work envelope, and their technological output in terms of accuracy and surface finish, etc.

The information structure based on absolute and constrained types of knowledge for components and processing systems used in our prototype process planning system GENPLAN is shown in Figure 1. At the planning stage, consideration of component
requirements and processing system capabilities takes place at two basic levels. At the highest level, the absolute knowledge is initially investigated. The objective here is to establish, on a feature by feature basis, if the component can be produced using the available system capabilities. The investigation is performed regardless of component size, material, accuracy, cost, etc. This is the first attempt at finding a match between component geometry and topology and the form generating capabilities of processing machine tools. The output of this investigation is all the possible processing methods capable of producing the component geometry.

![Diagram](image)

Figure 1. Information structure in GENPLAN.

The investigation then moves to the lower level of considering the feasible matches with respect to their constrained knowledge. Here, the system is trying to take account of the technological requirements of the component and technological constraints of the processing system. The feasible solutions at the feature level, followed by considering the feasible solutions for the component as a whole, are optimised, and an outline process plan is then generated.

3. Feature-Based Component Model

Recently, the concept of using component features for design and manufacturing applications has received much attention and research effort [3]. Features give a higher conceptual meaning to component characteristics by dissecting component geometry into recognisable and meaningful forms. The management, control and use of these groups of basic geometric entities is seen as a practical means of converting designs into manufacturable products. Thus, features are considered to be a suitable communication medium between design and manufacture [4].
For features to succeed in being a true communication medium in CAD/CAM applications, and to aid their integration, the "feature-based component model" created during the design stage should be capable of conveying sufficient useful information for use in the down-stream manufacturing activities. To achieve this the component model should be capable of describing not only the individual features and their attributes, but also the structural aspects of component geometry, i.e., how the features are connected together to form the component. The feature connectivity aspects of the component model are crucial in an application such as process planning, which places heavy demands on the ability to reason about component geometry in developing the component process plans.

In general, a form feature can be defined as a geometric entity that represents a shape pattern that has some significance. In this work, form features are considered as predefined geometric primitives that have attributes that can be used as component building blocks during the design stage. In GENPLAN, features are treated as component regions that have some significance in the context of machining operations. Examples of such machining features are HOLE, STEP, SLOT, NOTCH, etc., each of which needs to be described and classified uniquely within the component model.

3.1 Form Feature Taxonomy

Component form features are treated as volumes enveloped by entry/exit and depth boundaries. A component feature is described by determining the number of imaginary faces included in its definition, its external access directions (EAD's) (0, 1, 2, 3, 4, 5, 6), its boundary type (open, closed), its exit boundary status (through, not through), and its form variation with respect to its depth axis.

Based on their geometric attributes, features are uniquely classified into categories, classes and sub-classes, which may be followed by secondary forms to more fully describe the component features [5].

An overview of the main divisions in the form feature taxonomy is shown in Figure 2.

3.2 Feature Connectivity

Feature connectivity can be regarded as a component representation based on a type of parent/child relationship between its constituent features. Connectivity is specified using inheritance rules between the component real and imaginary surfaces. The inheritance tree at the component level has its origins in the component free surfaces. A free surface is the parent of all the features that have imaginary surfaces within its boundary. Any child feature becomes a parent feature of the adjacent features which have imaginary surfaces contained within the boundaries of any of its real surfaces.

An example of component connectivity information is represented as a directed graph in Figure 4.

4. Machine Tool Capability Model

A machine tool's structural configuration can be described as kinematic chains of elementary motions (translations and rotations) and rules for grouping these motions to constitute the machine form generating system. The machine form generating capabilities can then be represented as sets of elementary form generating schemas, i.e., tools of specific geometry and relative motions between each tool and the workpiece [6].
The machine structural configuration decides the possibilities for linking the form generating schemas during the processing of components. Analysing the elementary schemas and their relationships can lead to a representation of the operations the machine is capable of performing, which can be linked to its feature producing capabilities.

5. PROCESS PLANNING REFERENCE MODEL

The planning model which describe how a process plan is generated by GENPLAN is shown in Figure 3.

In this model, a component is treated as a group of features which are to be machined using the available processing system capabilities. Feature connectivity decides how the features relate to the component potential approach directions (PAD's) and feature relationships provide the constraints that may exist on the formation of feature clusters that can be machined from component PAD's.

Each feature has its transition diagram (FTD), which is a representation of all the technological solutions at the feature level (TSF's) as ordered sets of operations capable of producing the feature form and satisfying its technological requirements. A TSF is divided into states (groups of operations) which can be matched to one or more resource elements from the available machine tools (see Figure 4).

![Diagram of Form Feature Classification](image-url)
Figure (3) Process Planning Reference Model in GENPLAN
FEATURE 4: ROUND HOLE

FEATURE TRANSITION DIAGRAM

H - INITIAL STATE "HOLE"
S - INITIAL STATE "SOLID"
CD - CENTER DRILLING
D - DRILLING
M - MILLING
B - BORING
R - REAMING

FORM GENERATING SCHEMA (RESOURCE ELEMENT)

F41 : Rk + Tk + DRILL
F42 : Rk + Tk + REAMER

PROCESSING SYSTEM

Rz Tz tx ty

MOTIONS

TOOLS

TECH. OUTPUT

Rz = ..... rpm
Tx = ..... MM
Tz = ..... MM
S.F. = .......
TOL. = .......

Figure 4. Technological solution at feature level (TSF).
The breakdown of machine tools into resource elements allows process plans generation to be based on utilising parts of machine tool capabilities, it therefore provides better links with the machine shop scheduling system and minimises the ad hoc re-planning of component processing on the shop floor. It also allows greater flexibility to take into account any company specific machining strategy (flow line, machining centre, conventional machine tools, etc) that may be required for organising manufacturing activities.

6. COMPONENT SET-UP DETERMINATION

An overview of the SET-UP determination strategy adopted in GENPLAN is shown in Figures 4 and 5.

The determination of the final component set-ups involves a two stage optimisation procedure. In the first stage, optimisation is performed at the feature level, and in the second, the component as a whole is considered.

6.1. Feature Level Optimisation

At this stage all the resources needed for the operations required to produce all component features are considered. The first step to be performed here is to allocate manufacturing methods to component features. The technological solutions at the feature level (TSF) are stored in a manufacturing database as feature state diagrams (see Figure 4 for an example). Based on the individual feature’s technological requirements, its state diagram is parsed to determine all the feasible solutions for feature processing.

A TSF for a feature is an ordered set of operations that satisfy the feature geometry and technological requirements. Each operation in the TSF is described as a form generating schema, i.e. a set of motions and tools for performing the operation. To facilitate the matching between feature requirements and machine tool capabilities, a TSF is divided into states, each of which can be matched to one or more resource elements obtained from the analysis of the form generating capabilities of the available machine tools.

The set of resources needed to produce each component feature, based on its technological solutions, is determined. The total resource set to produce the component is then minimised and the appropriate TSF is attached to each component feature.

6.2. Component Level Optimisation

This stage involves some reasoning about component geometry. Based on feature connectivity (see Figure 5), it is possible to determine all the component potential approach directions (PAD’s), the features that can be machined from each direction and, as a result of the first stage optimisation, the set of operations attached to each component feature. This represents the maximum number of potential component set-ups and the work content to be performed from each component PAD before any optimisation.

Determination of the component set-ups can be considered initially as a process of clustering [7] of feature states in order to carry out the required machining operations using common resources from common component approach directions (PAD’s). All feature states are grouped in clusters in which feature states can appear in more than one cluster in different component PAD’s.

The final component set-ups are selected by minimising the number of clusters. This is based on step by step selection of the cluster containing the maximum number of feature states and then removing those states from the remaining clusters in the other component PAD’s.
Figure 5. An overview of set-up determination in GENPLAN
It is important that clustering is performed while observing the precedence relationships that may exist between component features. It is almost impossible, however, to predefine all the possible feature relationships (concentricity, parallelism, perpendicularity, etc) and the implications that multi-relationships between features may have on the component process plan. Many of the issues involved in making these decisions are company specific and experience based.

The important factor as far as system implementation is concerned, however, is that the planning system should be able to perform feature clustering within the constraints that feature relationships may impose on feature grouping.

The currently implemented version of GENPLAN recognises sequential, parallel and same set-up as constraints to be observed during feature clustering.

7. CONCLUSIONS

Modelling of component characteristics and processing system capabilities provides a good basis for solving many of the tasks involved in process planning. The process planning reference model used for plan generation in GENPLAN, described in this paper, is providing an appropriate framework not only for process plan generation but also for linking the plans to a machining strategy and machine shop scheduling/loading systems.

The feature-based model for describing component geometry and connectivity, and the model for representing the form generating capabilities of machine tools outlined in this paper, are proving very useful for geometric reasoning and SET-UP determination tasks in our computer aided process planning system.

8. ACKNOWLEDGEMENT

The research reported here is supported by grants provided by the ACME Directorate of SERC, their financial contribution is gratefully acknowledged.

9. REFERENCES


FUNCTIONAL DESCRIPTION OF MACHINES, COMPONENTS AND TOOLS FOR A GENERATIVE PROCESS PLANNING SYSTEM

Nabil N Z Gindy and Sue Huang
Department of Manufacturing Engineering
Loughborough University of Technology

1. INTRODUCTION

Process planning is an example of an application in which the decision logic is primarily based on matching component requirements to the capabilities of manufacturing processes and resources (machine tools, tool fixtures, etc) available for their processing. The relevant component characteristics are normally its geometry, topology, dimensions, material and accuracy attributes. Processing system attributes represent its form generating functions, work envelope, and its technological output in terms of accuracy, surface finish, etc. The planning domain involves vast amounts of data, knowledge and complicated inferencing processes. The identification, categorisation and structuring of planning information are some of the key issues that have to be addressed during system development. This can lead to a decomposition of the planning domain into manageable sub-tasks which can be tackled using appropriate methods for information representation and processing.

The objectives of this research are to develop planning logic, based on integrated models for describing component information, and the capabilities of machine tools - in a format suitable for manufacturing decision making - and to test the developed methods using a small scale prototype planning system.

The aim of this paper is to report on some of our ongoing research in developing a prototype generative process planning system for machined components. The paper describes the "Planning Data Reference Model", see Figure 1, that acts as the integrating framework for the large volume of diverse information needed in the planning domain. The feature-based "Component Data Model", based on our hierarchical structure for feature definition classification and component connectivity relationships, is also outlined.

2. COMPONENT DATA MODEL

Recently, the concept of using component features for design and manufacturing applications has received much attention and research effort [1]. Features give a higher conceptual meaning to component characteristics by dissecting component geometry into recognisable and meaningful forms. The management, control and use of these groups of basic geometric entities is seen as a practical means of converting designs into manufacturable products. Thus, features are considered to be a suitable communication medium between design and manufacture [2].

For features to succeed in being a true communication medium in CAD/CAM applications, and to aid their integration, the "Feature-based Component Model" created during the design stage should be capable of conveying sufficient useful information for use in the down-stream manufacturing activities. To achieve this, the component model should be capable of describing not only the individual features and their attributes, but also the structural aspects of component geometry, i.e. how the features are connected together to form the component. The feature connectivity aspects of the component model are crucial in an application such as process planning, which places heavy demands on the ability to reason about component geometry in developing the component process plans.
In "GENPLAN" a component is treated as a set of connected features, with each feature representing a component region that has some significance in the context of machining operations. Examples of such machining features are HOLE, STEP, SLOT, NOTCH, etc, each of which needs to be described and classified uniquely within the component model.

2.1 Feature Classification

A form feature is treated as a parameterised geometric entity described by any number of real surfaces and a specific number of imaginary surfaces connected together in a particular topology. The determining factors for feature classification are: the number of imaginary faces included in its definition (ie the number of its external access directions, EADs), its boundary type (open, closed), its exit boundary status (through, not through), and its form variation with respect to its depth axis. Based on their geometric attributes, features are uniquely classified into categories, classes and sub-classes, which may be followed by secondary forms to more fully describe the component [3]. The feature technological constraints describe a specific instance of a feature on the component. They represent the feature information in terms of its dimensions, accuracy, tolerance, etc, that are to be taken into account in the selection of its processing methods.

2.2 Feature Technological Solutions (TSF)

A Feature Transition Diagram (FTD) is a representation of all the possible Feature Technological Solutions (TSF) that can be used to produce a feature geometry and topology at various levels of technological requirements. A fully defined feature has a set of equally feasible and equally weighted TSFs and each TSF is an ordered set of operations capable of producing the feature form (geometry and topology) and satisfy its technological requirements (mini-plan for producing a feature).

2.3 Component Connectivity

Form features are geometric entities which relate to local component regions. To be able to reason about component geometry during process planning, it is important that the component model should capture the structural aspects of component geometry, ie to describe the relationships (connectivity) which determine how the component is constructed from its constituent features. Component connectivity can be regarded as a component representation based on a type of parent/child relationship between its constituent features. Connectivity is specified using inheritance rules between the component's real and imaginary surfaces. Component connectivity is represented as a directed graph in Figure 2.

2.4 Component Potential Approach Directions (PADs)

An aspect which is perhaps difficult to resolve during process planning is to automatically determine the Potential Access Directions (PADs) that can be used for component machining. In "GENPLAN" this is done through grouping component features based on the commonality of their External Access Directions (EADs). A component PAD is a common direction that coincides with the EAD of at least one of its features.

2.5 Component Relationships and Clustering Constraints

Component PADs represent the possible machining directions for groups of features. Feature clustering within each PAD should be reformed while observing the component relationships that may exist between its constituent features. Such relationships describe the inter-feature constraints that may be specified by the designer to guarantee some component functionality. They can, therefore, involve geometric and technological relationships on a multi-feature, multi-relationship basis.

Some component relationships lead to mandatory restrictions on feature clustering. For example, a child feature cannot be machined before its parent feature, two holes with a tight concentricity tolerance leading to a mandatory restriction that they must be
processed in the same set-up. Alternatively, some of the clustering constraints can be viewed as "good" manufacturing practice restrictions. These are restrictions on feature grouping that originate from some practical/technological preferences when processing groups of features on a component. For example, mill the slot before drilling the hole at its bottom. Many of the issues involved in making these decisions are company specific and experience based.

The approach adopted in "GENPLAN" is to define an action set that can cope with the majority of the possible feature relationships. Our current action set includes processing component features sequentially, in parallel, or in the same set-up.

3. PROCESSING SYSTEM DATA MODEL

As shown pictorially in Figure 2, a processing system or machining facility can be viewed as representing the collective capabilities of the set of available machine tools and cutting tools that can be used for component processing. A machine tool's structural configuration can be described as kinematic chains of elementary motions (translations and rotations) and rules for grouping these motions to constitute the machine form generating system. The machine form generating capabilities can then be represented as sets of elementary form generating schemas, i.e., tools of specific geometry and relative motions between each tool and the workpiece [4].

A machine tool capability can be represented by a set of resource elements (REs) and each RE is a set of form generating schemas (operations) which can be obtained from one or more of its available machine tools. The REs, therefore, provide a basis for comparing the capabilities of different machine tools and give the planning system an ability to generate alternative process plans on less than a whole machine tool basis (with obvious implications for loading/scheduling of the machining facility).

4. STRATEGY FOR PROCESS PLAN GENERATION

An overview of the SET-UP determination strategy adopted in "GENPLAN" is shown in Figure 2.

During planning, a component is treated as a group of connected features which are to be machined using the available processing system resource. Feature connectivity decides how the features relate to the component potential approach directions (PADs) and feature relationships provide the constraints that may exist on the formation of feature clusters that can be machined from component PADs. Each feature has its transition diagram (FTD) representing all the technological solutions at the feature level (TSFs) as ordered sets of operations capable of producing the feature form and satisfying its technological requirements.

4.1 Feature Level Optimisation

The technological solutions at the feature level (TSF) are stored in a manufacturing database as feature state diagrams and, based on the individual feature's technological requirements, its state diagram is parsed to determine all the feasible solutions for feature processing. Each operation in the TSF is described as a form generating schema and a TSF is divided into states, each of which represents a group of operations that can be obtained from one or more resource elements based on the form generating capabilities of the available machine tools. The set of resources needed to produce each component feature, based on its technological solutions, is determined. The total resource set to produce the component is then minimised and the appropriate TSF is attached to each component feature.

4.2 Component Level Optimisation

Taking into account the clustering constraints that may exist due to feature relationships, the objective here is to group the operations to be used for producing all component features into feasible component PADs, such that the number of component set-ups is
minimised. Based on feature connectivity (see Figure 2), all the component potential approach directions (PADs) are determined and, as a result of the first stage optimisation, the set of operations is attached to each component feature. This represents the maximum number of potential component set-ups and the work content to be performed from each component PAD before any optimisation.

Determination of the final component set-ups can be considered initially as a process of clustering [5] of feature states in order to carry out the required machining operations using common resources from common component approach directions (PADs). The final component set-ups are selected by minimising the number of clusters. This is based on step by step selection of the cluster containing the maximum number of feature states and then removing those states from the remaining clusters in the other component PADs.

5. SYSTEM IMPLEMENTATION

The knowledge-based system GENERIS [6] was selected as the development tool for "GENPLAN". The implemented system consists of a user interface, a knowledge acquisition function, an inference engine, an explanation of output function, and the knowledge base which includes a manufacturing database and production rulebase. The knowledge acquisition function describes the planning domain in a form that can be interpreted by the inference engine. The manufacturing database contains the feature based component description data and the machining capability data.

The planning logic is represented as production rules in the rule base executed by the inference engine. The control program in the inference engine, which constructs queries for extracting information from the database and linking the database to the front end, is written in GENERIS high level language.

6. CONCLUSIONS

The feature-based "Component Model" used to describe component geometry and connectivity, outlined in the paper, is proving very useful for geometric reasoning and component set-up determination tasks in generative planning systems.

The planning data "Reference Model" used for developing our prototype planning system, "GENPLAN", described in this paper, is providing a good basis for structuring and integrating the diverse information requirements of the process planning domain.

7. REFERENCES


FIGURE 1 Process planning Reference Model in GENPLAN
Figure 2 Setup Determination Strategy in GENPLAN