A CODING SCHEME BASED GENERATIVE CAPP SYSTEM FOR PRISMATIC COMPONENTS USING EXPERT SYSTEM METHODOLOGY

Bing Jiang (B.E.)

August, 1997

A Thesis Submitted to the Department of Mechanical Engineering
The University of Adelaide

for the Degree of Master of Engineering Science,

Supervisor: Associate Professor: Manfred Zockel,
Dr. Keith Baines

Table of Contents

Page

List of Figures

List of Tables

Abstract

This thesis describes a research project in which a generative computer aided process planning (CAPP) system, termed a coding scheme based generative CAPP expert system(CSB-GCAPPES), for machining prismatic components is developed. The research project is based on the principle that GT coding schemes can become the component's design representations, and that expert system technology can map the knowledge base required by a generative CAPP system. The CSB-GCAPPES aims to help the process planner produce quicker and more consistent process plans which will also optimise the machining process.

A new GT coding scheme based on existing GT classification coding schemes components has been developed in this research project which characterises engineering drawings of prismatic components. It has a hybrid code structure (monocodes and polycodes) and describes the component's design and manufacturing information. The monocodes represents the component's overall shape, maximum size and material while the polycodes represents the machined shapes such as plane surfaces, holes and slots and their attributes such as dimensions, tolerances, surface finishes and orientation. A code generated by the new coding scheme has enough design and manufacturing information to complete the process planning activity.

Using an expert system building tool, the CSB-GCAPPES is designed to generate a new GT code automatically, and capture and maintain the manufacturing knowledge of process planners for machining prismatic components. It is a rule-based program that recognises the new GT code and creates a process plan which lists all operations, sequences, machines, cutting tools and cutting conditions such as cutting speed, feed rate and depth of cut. The prototype system has demonstrated the feasibility of both the new GT coding scheme and the expert system approach for process planning in prismatic components manufacturing.

Statement

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person, except where due reference has been made in the text.

I give consent to this copy of my thesis, when deposited in the university library, being available for loan and photocopying.

Signature

Date:

Acknowledgements

The author expresses his gratitude and sincere thanks to his supervisors, Assoc. Prof. M. Zockel and Dr. Keith Baines, for their assistance, valuable advice and continuous support provided during the author's entire research at The University of Adelaide, and for the energy and the time devoted to the supervision of this research.

The author also wishes to thank Mr. Ron Jager in the Department's workshop who provided a great deal of practical information on machining procedures and processes, and Mr. Jonathan May for his assistance with computing matters. Thanks are also due to all colleagues who willingly offered their assistance at all times.

The author expresses his deep gratitude to his parents who motivated him to undertake this research, their continuous encouragement and personal example. Sincere thanks and affectionate gratitude are also given to the author's wife, Hong, for the long working hours spent in providing the author financial support, for the assistance in the preparation of this thesis, and for the patience, understanding and support that made this thesis possible.

Chapter 1. Introduction

This chapter provides an introduction to computer aided design/manufacturing (CAD/CAM), computer integrated manufacturing (CIM), and to computer aided process planning (CAPP) which is part of CIM and is the aspect mainly considered by this thesis. The aims of the research project are also introduced in this chapter.

1.1 CAD/CAM and CIM

CAD/CAM is concerned with the use of computer systems (hardware and software) to perform certain functions in design and manufacturing. These functions were usually manual tasks before computer systems were implemented in industry. For example, CAD systems such as CATIA, Pro/engineer, Unigraphic, CADDS and AutoCAD assist designers in the creation, modification, analysis, and optimisation of designs. Using 2D drafting and 3D modelling in CAD systems, designers can quickly create design drawings and databases for manufacturing. CAM systems such as MASTERCAM and SMARTCAM are used to assist manufacturing companies to plan, manage, and control the productions. There is also application software for factory management involving both inventory and production, for robot programming and simulation, for finite element analysis and for product assembly. A summary of CAD/CAM systems are shown in Table 1.1. The CAD/CAM technologies assist the companies to become more productive and agile by making communication more effective, shortening the engineering cycles and reducing lead time and costs.

There is no shortage of CAD/CAM solutions to specific manufacturing problems. However one of the biggest problems facing companies today is that of 'isolated automation' which means improving the situation at one point in the company but not in a way that benefits other departments in a company. Thus, integration of CAD/CAM systems in a manufacturing environment becomes the most important research topic currently which is termed CIM.

The key to achieving the CIM system is to computerise every activity in the manufacturing company. However, the development of computer aided process planning (CAPP) is far

behind the development of other computer systems in manufacturing such as the CAD system. Thus, the CAPP system becomes the 'bottle neck' of the development of CIM.

Table 1.1 Summary of CAD/CAM Systems

CAD/CAM System	Application	Example Software
CAD	Detail drawings(2D & 3D modelling), perspective illustrations, NC toolpath, assembly	Pro/Engineer, Pro/PDM, CADDS, CATIA, Unigraphics, AutoCAD,
CAE	Finite element analysis, kinematics design	RASNA, ANSYS, PATRAN, ADAMS
CAM	CNC machines control, inventory and production management	SMARTCAM, MASTERCAM, QUEST, CUTDATA
Robot control	Robot programming and simulation	ROBCAD

1.2 The Computer Aided Process Planning(CAPP)

The Computer aided process planning (CAPP) operation is one of the CIM technologies. CAPP systems are concerned with the use of computer systems to perform process planning functions which means that in this process, raw material size, machines, cutting tools, cutting speed, feed rate, depth of cut as well as machining sequence are selected.

Process planning is usually manual work. Designing an economic process plan needs much effort. A process planner has to examine all the combinations of operations, machine tools, cutting tools and parameters. But the human planner can be fatigued, ill, or temperamental. Thus, CAPP systems are built to assist human planners.

Since 1960s', two types of CAPP approaches have been developed:

1. Variant CAPP systems.
2. Generative CAPP systems.

1.2.1 The Variant CAPP

The variant CAPP was the first approach to CAPP. It was based on the concept that similar components should have similar process plans. In the variant CAPP system proven process plans are stored in the computer system. When a new component requires a process plan, the planner first identifies one or several similar components whose process plans are already stored in the computer. Then the planner retrieves one or more of these process plans which can be used directly or with some modifications. Thus, variant type CAPP system saves process planning time and increases planning consistency. But its efficiency relies on the quality of the process plans which were previously specified.

In variant CAPP, the planner usually uses classification and coding systems to identify similar components. In simple words, a code is used to represent the component and similar codes represent similar components.

Although variant CAPP systems have been installed in many manufacturing companies, as will be described in the next chapter, this type of system is inflexible, is unable to expand, and is difficult to be customised for a company's needs. Since the standard process plans have already been stored in the computer system, it requires extensive software rewrites and experienced programmers to maintain the system when the company products are changed or when machining equipment is upgraded. The variant CAPP system also needs extensive human interaction and can not be used in a fully automated environment.

1.2.2 The Generative CAPP

Since the late seventies another approach, generative CAPP, has been developed and become a major research topic in the field of CAPP. Generative CAPP creates a process plan from information available in a manufacturing database. The information in the manufacturing database includes the available machining equipment, the required cutting conditions and other manufacturing related information. When the system receives the new component design it is able to generate the process plan for the component. A generative CAPP system includes the following subsystems, Wang and Li(1991):

1. Component description

A subsystem must identify a series of component characteristics, including geometric features, dimensions, tolerances, surface requirements, etc.

2. A subsystem to select and sequence individual processing operations

Within this subsystem, decision logic is used to associate appropriate operations with particular features of the component. Further heuristic algorithms are used to calculate operation steps, times and sequences.

3. A database of available machines and tooling

This allows the system to associate machines and tooling with individual operations, having regard to the requirements of the operation in terms of machining accuracy, surface finish, size etc.

4. A subsystem to define the machining parameters

This subsystem uses information in reference books to define such parameters as spindle speed, feed and depth of cut.

5. A report generator which prepares the process plan report.

In generative CAPP systems, component design representations, the manufacturing database implementations and process plan creation algorithms have been developed since the late seventies. The advantage of the generative CAPP system approach is that it has the potential to produce an automated optimum plan without referring to previous process plans, and it can also be part of a computer integrated manufacturing (CIM) system. Thus, generative CAPP systems have important advantages.

1.3 Aims of the Research Work

As ElMaraghy described in 1993 (ElMaraghy, 1993), the effectiveness of CAPP systems have not been fully realised by industry and two main problems exist for the building of a CAPP system:

- lack of complete component description for input data to a CAPP system,
- lack of effective representation of process planning knowledge.

The objective of this research project is related to these two problems.

The first aim is to optimise the component description which can be used as input into a generative CAPP system. For this reason a new group technology (GT) based coding scheme has been developed which represents the machined surfaces of prismatic components. The new GT coding scheme defines the surface as a machined shape and also defines the attributes of each machined shape. The machined shapes includes plane surfaces, holes, slots, pockets, contours, etc. The attributes include size, tolerance, surface finish and orientation. The new GT coding scheme also represents the overall prismatic component by including the overall shape, overall size and material. Thus, the description of the prismatic component has three levels: overall description, machined shapes description (surfaces to be machined), and machined shape's attributes description(size, tolerance, surface finish and orientation). It will be shown that the new GT coding scheme captures enough design and manufacturing information for a generative CAPP system (Jiang et al 1997).

The second aim of this research project is to consider knowledge representation of the process planning system for milling prismatic components and particularly for automated processing. By utilising the proposed new GT code, current databases and process planning knowledge, the development of a generative CAPP system on a PC platform, using expert system methodology, will enable a company to expand and modify the system to optimise current machining facilities. The process planning knowledge in this generative CAPP expert system(CSB-GCAPPES) is represented by the declarative and procedural knowledge representation provided by an expert system shell, ART-IM. The CSB-GCAPPES has eight modules which allow the new GT code to be entered, enabling the processes and equipment to be selected and produce the process plan as a text file. The possibility of the CSB-

GCAPPES being integrated with a CAD system and CNC machine controller is also discussed in Chapter seven.

Chapter 2. Literature Review

In section 2.1 of this chapter the literature of research of GT classification and coding scheme is reviewed. In section 2.2 and section 2.3 the literature of research of variant CAPP systems and generative CAPP systems are also reviewed.

2.1 GT Classification and Coding Schemes

GT classification and coding schemes are very important for CAPP systems and are developed to group similar components into families (Offodile et al 1994). A GT code normally consists of a sequence of symbols, usually Arabic number or characters. The code numbers or characters identify a component's design characteristics and features, and/or its manufacturing attributes. Overall shape, external shape, internal shape, dimension, tolerance and raw material are some of the categories commonly included.

The three structures in classification and coding schemes are as follows (Hyer, et al 1985):
(1) monocode structure: in the monocode structure, the interpretation of each succeeding symbol depends on the value of the preceding symbols. It is also known as hierarchical structure;
(2) polycode structure: in the polycode structure, the interpretation of each symbol in sequence is fixed and does not depend on the value of the preceding digits;
(3) hybrid structure: in the hybrid structure, a combination of the above two types of structure is used.

Many coding schemes have been developed and they will be described in the following sections.

2.1.1 The Opitz Classification and Coding Scheme

The Opitz coding scheme was developed by H. Opitz (1970) of the Technical University of Aachen in Germany. It represents one of the pioneering efforts in the group technology based coding schemes and is perhaps the best known of the classification and coding schemes. The Opitz coding scheme (Table 2.1a and Table 2.1b) is made up of five main digits with four

supplementary digits and it is a hybrid coding scheme. Each digit has fixed numbers 0-9. The first five digits represent the geometric information describing different kinds of components. Thus these five digits indicate the shape or form of a component. The next four digits are of particular importance to manufacturing and relate to the dimensions, component material, raw shapes and workpiece finished accuracy.

Table 2.1a Opitz Coding Scheme (Geometrical code) for Cubic Components

1st Digit		2nd Digit		3rd Digit		4th Digit		5th Digit	
Component Class		Overall Shape		Bore, Rotational Surface Machining		Plane Surface Machining		Holes And Gear Teeth	
		0	Block Components-Rectangular Prism	0	No Rotational Machining	0	No Surface Machining	0	No holes or Gears
		1	Block Components-Rectangular With Deviations	1	One Bore, Smooth	1	Functional Chamfers	1	No Gear and Forming-Holes Drilled in One Direction Only
		2	Block Components-Compounded of Rectangular Prisms	2	One Principal Bore, Stepped to One or Both Ends	2	One Plane Surface	2	No Gear and Forming-Holes Drilled in More Than One Direction
		3	Block Components-With Location Surface	3	One Principal Bore With Shape Elements	3	Stepped Plane Surfaces	3	Holes Drilled in One Direction Only-With Drilling Pattern
		4	Block Components-With Divided Surface	4	Two Principal Bores, Parallel	4	Stepped Plane Surface at Right Angles, Inclined and/or Opposite	4	Holes Drilled in More Than One Direction-With Drilling Pattern
		5	Block Components-Other Than 0 to 4	5	Several Principal Bores, Parallel	5	Groove and/or Slot	5	Formed, No Auxiliary Holes
		6	Not Split Box Components-Rectangular Prisms	6	Several Principal Bores, Other Than Parallel	6	Groove and/or Slot and 4	6	Formed With Auxiliary Holes
		7	Not Split Box Components-Other Than 6	7	Machined Annular Surfaces, Annual Grooves	7	Curved Surface	7	Gear Teeth, No Auxiliary Holes
8	Cubic Components A/B<=3, A/C<4	8	Split Box Components-Rectangular Prisms	8	7+Principal Bore(s)	8	Guide Surface	8	Gear Teeth With Auxiliary Holes
		9	Split Box Components-Other Than 8	9	Others	9	Others	9	Others

Table 2.1b Opitz Coding Scheme (Supplementary Code) for Cubic Components

6th Digit		7th Digit		8th Digit		9th Digit	
Diameter or edge length(mm)		Material		Initial form		Accuracy	
0	<=20	0	Cast Iron	0	Round Bar, Black	0	Not Specified
1	>20<=50	1	Modular Graphite Cast Iron and Malleable	1	Round Bar, Bright Drawn	1	Accuracy Class 2
2	>50<=100	2	Steel, Not Heat Treated	2	Bar-Triangular, Square, Hexagonal	2	Accuracy Class 3
3	>10<=160	3	Heat Treatable Low Carbon Steel, Not Heat Treated	3	Tubing	3	Accuracy Class 4
4	>160<=250	4	3+Heat Treated	4	Angle, U-, T- Sections	4	Accuracy Class 5
5	>250<=400	5	Alloy Steel, Not Heat Treated	5	Sheet	5	2+3
6	>400<=600	6	Alloy Steel, Heat Treated	6	Plate and Slabs	6	2+4
7	>600<=1000	7	Non-ferrous Metal	7	Cast or Forged Components	7	2+5
8	>1000<=2000	8	Light Alloy	8	Welded Assembly	8	3+4
9	>2000	9	Other Materials	9	Pre-Machined	9	2+3+4+5

2.1.2 The KK-3 Classification and Coding Scheme

The KK-3 coding scheme (Chang and Wysk, 1985) was developed by the Japan Society for the Promotion of Machine Industry as a general purpose classification and coding scheme for components. It contains 21 digits and is a monocode scheme. Because of the increased number of digits in the code, it can carry more information than the Opitz coding scheme. However, it still has fixed numbers 0-9 for each digit and does not carried enough design and manufacturing information. For example, the dimension and surface finish information of each machined shape are missing.

The first digit of KK-3 classification and coding scheme classifies the general function of a component, such as gear, shaft, drive, moving and fixing and so on. More detailed functions are represented in the second digit. Up to one hundred functional names for rotational and non-rotational components are classified. Two digits are used to classify materials. The first digit shows the material type and the second digit describes the shape of the raw material. The KK-3 scheme also classifies dimensions and dimension ratios. The other digits description are shown in Table 2.2.

2.1.3 The MICLASS Classification and Coding Scheme

The MICLASS System was originally developed by The Organisation for Industrial Research Inc. (OIR, 1981), which is located in Waltham, Mass., USA. It is a polycode-structured code of 12 digits. The code includes both design and manufacturing information such as the main shape, shape elements, position of shape elements, main dimensions, ratio of the dimensions, auxiliary dimensions, form tolerance. The machinability of the material is also included. An additional 18 digits of code are also available for user-specified information (i.e. component function, lot size, major machining operation, etc.). These supplementary digits provide flexibility for system expansion. Several application programs based on MICLASS are currently available, such as the MIPLAN (used in the Cleveland-based Lamp Equipment Operation of General Electric Co., USA) and the MIAPP process-planning systems(Schaffer, 1980).

2.1.4 The DCLASS Classification and Coding Scheme

The DCLASS scheme was developed by Del Allen (Allen, 1979) at Brigham Young University. It is a polycode-structured scheme that can form codes for components, materials, process, machines, and tools. An eight-digit code is used which is shown in Table 2.3. In DCLASS, there are several branches. Each branch represents a condition, and a code can be found at the terminal of each branch. Multiple passes of the decision tree allow a complete code to be found.

Table 2.2 KK-3 Classification and Coding Scheme

Digit 1	Main Function
Digit 2	Detailed Functions
Digit 3	Materials
Digit 4	Material Treat
Digit 5	Edge Length
Digit 6	Edge Width
Digit 7	Primary Shape
Digit 8	Formed Shape-Bending Direction
Digit 9	Formed Shape-Bending Angle
Digit 10	External Plane Surface
Digit 11	External Curved Surface
Digit 12	Main Shaped Surface
Digit 13	Cyclic and Auxiliary Surface
Digit 14	Main Hole-Direction and Step
Digit 15	Main Hole-Screw Thread
Digit 16	Internal Surface Other Than Hole
Digit 17	Auxiliary Hole-Direction
Digit 18	Auxiliary Hole-Shape
Digit 19	Auxiliary Hole-Special Hole
Digit 20	Non-Machining
Digit 21	Accuracy

Table 2.3 DCLASS Coding Scheme

Digit 1-3	Basic shape
Digit 4	Form feature
Digit 5	Size
Digit 6	Precision
Digit 7 and 8	Material

2.2 Variant CAPP Systems

Variant CAPP systems were the first stage of CAPP research. Building variant CAPP systems was the major CAPP research activity in the 1970's and the first half of the 1980s (Chang, 1990). The method of building variant CAPP systems was based on group technology (GT) classification and coding schemes. GT classification and coding scheme groups components into families on the basis of similarity of attributes such as shape, surface finishes, materials, tolerances and required manufacturing processes.

For each component family, standard process plans of components were established and computer stored (Allen and Smith, 1980). A process plan for a new component is actually a standard process plan retrieved from computer storage which has been edited by the process planner. The procedure used in variant CAPP systems is as follows: a new component was first coded by a process planner to identify which family it belonged to; then the standard process plan of the family was retrieved from computer storage. The standard process plan retrieved could be the process plan of the new component or modified by the user (i.e. process planner) of the variant CAPP system. Thus, the GT classification and coding scheme in variant CAPP systems serves to distinguish families of components and the process planning knowledge is represented by standard process plans.

A number of variant CAPP systems have introduced into industry. In 1970s. The Organisation for Industrial Research Inc. developed a variant CAPP system, MIAPP(OIR, 1981). MIAPP is based on the MICLASS coding scheme.

Some other variant CAPP systems developed in the 1970s and the 1980s were GENPLAN by the Lockheed Co. (Tulkoff, 1981) and XPS-I which was CAM-I's (CAM-I was an international research group: Computer Aided Manufacturing-International) experimental planning system(Sack, 1983). Tulkoff (1981) showed that GENPLAN helped to reduce the clerical work of the process planner in the Lockheed company by up to 40% of process planning time. Thus, many users were pleased with the time and cost saved by the variant systems. Many variant CAPP systems are still used by different companies, e.g., MULTIPLAN (OIR, 1983).

However, variant CAPP systems have the shortcomings of being difficult to up-date and difficult to automate. Existing coding schemes all have fixed digits and do not carry enough information for detailed process planning. For detailed process planning, each individual machined shape, such as a hole or a step of a component and their attributes, such as dimension, tolerance and surface finish information, must be present. The reason may be that they are either design or variant process planning oriented. For example, the Opitz code "in its basic nine digit form is found to be inadequately detailed both with respect to component geometry description and manufacturing information" (Phillips, 1978). Clearly a more comprehensive and flexible-digit coding scheme is required which is more suitable for CAPP systems.

Variant CAPP systems which store a lot of standard process plan also cause problems. If some machining facilities were changed or up-dated the whole process plans affected in the system program must be rewritten. On the other hand, a standard plan in the variant CAPP system has been created for a family of components, but each component in a family is different. Thus, a new process plan must be created by an experienced person. Consequently the automation of process planning cannot be realised by using variant CAPP systems.

2.3 Generative CAPP Systems

The second stage of CAPP development was the introduction of generative CAPP systems which came under development in the late seventies. As described in chapter one, generative CAPP systems generate process plans from scratch for every new component description by the automatic generation of process plans. A component design representation model and process knowledge representation are the main ingredients of a generative CAPP system.

2.3.1 Component Design Representation

From the literature review (Chang, 1990), there are three component design representation models: GT coding scheme, special descriptive language and CAD models. These are described in the following sections.

2.3.1.1 GT Coding Scheme In Generative CAPP Systems

In existing GT coding schemes based on generative CAPP systems, the component design representation models mainly are GT coding schemes which are used in the variant CAPP systems. Several GT coding schemes, including the Opitz scheme (Opitz, 1970), the KK-3 scheme (Chang and Wysk, 1985), the MICLASS scheme (Houtzeed, 1976), and the COFORM scheme (Wysk, 1977) are used. Some other coding schemes such as the APPOCC scheme (Phillips, 1978) and the coding scheme used in JLG Industries (Jashi, 1994) have also been developed. However, the GT coding schemes used in generative CAPP systems are the same or similar to those used in variant CAPP systems which have fixed digits. The shortcomings of insufficient information and fixed digits still exist. Thus, a new coding scheme which carries enough information and has flexible digits is developed in this research. The new coding scheme represents every machined shape's attributes of the component (prismatic), i.e. the more the component has machined shapes, the more the code has the digits.

2.3.1.2 Special Descriptive Language

Special descriptive language is another component design representation model for generative CAPP systems (Lauwers and Kruth, 1994). It describes the machined shapes such as a hole, a slot, a pocket, a step, etc. as features. Thus, the special descriptive language is also termed feature-based component description.

One method in special descriptive language is frame-structure representation. Frame-structure representation describes a component using frames. Each frame has slots to store the feature's characteristics and the relationships with other frames. Usually, the top level of a frame is a component name. Each machined surface in the component is represented in a machined surface frame. Machined surface frames are linked to the component frame through a pointer slot. An example of a component can be shown as (Chang, 1990):

```
(T001
(A_KIND_OF (VALUE PART))
(PART_NAME (VALUE TEST_PART))
(BATCH_QTY (VALUE 10))
(ORDER_QTY (VALUE 10))
(MTL_SIZE (VALUE (4.0 3.0 1.0)))
(MTL_TYPE (VALUE STEEL))
(MTL_SPEC (VALUE SAE_1050))
(MTL_COND (VALUE TEMPERED))
(HARDNESS (VALUE (450 500)))
(SURFACE_TREAT (VALUE NIL))
(MADE_UP_OF (VALUE (F1 F3 F7 F10))))
```

It can be seen that in the above example there are slots for the components name, batch quantity, order quality, material size, type, specification, condition, hardness, and surface treatment. The last slot shows the relationship with other frames.

The advantage of the special descriptive language is the fact that it provides more detailed information of a component. The AUTAP system is a special-descriptive-language-based generative CAPP system described by Evershein and Esch (Evershein and Esch, 1983). This system uses a special descriptive language to represent a component. The component is described using both geometric and technological elements. It also uses subordinate elements such as patterns of holes. The CIMS/DEC is another component description language developed by Kakino et al (Kakino et al, 1977), as is the CIMS/PRO system built by Iwata et al (Iwata et al, 1980). Jakubowski (Jakubowski, 1982) also proposed a similar language format for describing components. GARI (Descotte and Latombe, 1981) is a generative system which also uses a component description language.

The special descriptive language is able to provide more information of a component than a fixed-digit coding scheme. Thus, a generative CAPP system using a special descriptive language can generate more detailed process plans. However, a shortcoming of the component description language based CAPP systems is that they require more effort from

the user of the system to prepare the input data. Although computer systems may assist users to automate the input procedure, but the systems are difficult to be modified because they need a special descriptive language expert.

2.3.1.3 CAD Models

Although GT coding schemes and component descriptive languages can provide information for process planning, the component design still needs to be manually converted into one of the two representations. Since a component is modelled completely on a CAD system, using CAD models as input to the process planning system can eliminate the human effort.

The CAD models used as input to the process planning system are currently 3D solid models. Most CAPP researchers used either constructive solid geometry(CSG) or boundary representation(B-Rep) or combine the two models because they are the best models which are used in conjunction with feature-based design. Feature-based design is a new design concept which aims at providing a common CAD model database. In feature based design, a feature refers to a subset of geometry on a component, which is similar to that in a special descriptive language. Unfortunately, at the moment, there is not a definition of the term *feature* which is agreed upon by everyone(Chang, 1990).

A number of CAD-model-based generative CAPP systems are: a system called Intelligent Product Design and Manufacturing(IPDM) and is described by ElMaraghy(1991), which uses a hybrid CSG and B-rep models to represent the component; PART is another example of a solid models based generative CAPP system(Van Houten, 1992); a commercial solid models based generative CAPP system called Design-To-Manufacture(DTM/CAPP) is built by a collaborative effort between Europe and North America(Jasperse, 1992). Because feature-based design is easy for simple components and extremely difficult to use for complex components, the CAD-model-based generative CAPP systems are limited to a small range of components and cannot be used in real industrial work (Yip-Hoi and Dutta, 1995).

Using a solid model database generative CAPP system is more effective because it can be developed as a fully automatic process planning system. Choi et al (Choi, 1982, Choi et al,

1984, Choi and Barash, 1985) introduced an algorithm which recognises the machined surfaces from a 3D boundary file. Henderson and Musti (Henderson and Musti, 1988) also developed a system which analysed the solid model of a component and generated a GT component code. Currently, the automatic recognition of a component from a 3D solid model can just analyse very simple surfaces and cannot be used industrially (Henderson and Musti, 1988). Furthermore, very few recognition systems are able to identify global constraints such as clamping requirements and interference (e.g. collision between the tool and workpiece), (Choi et al, 1984).

2.3.2 Process Knowledge Representation

In generative CAPP systems, process knowledge representation controls the decisions made in process planning, such as machine selection, tool selection, process sequence, etc. A generative CAPP system constructs a process plan in much the same way as a human planner because it is based on knowledge and experience.

From the 1980's expert systems began to be used in process planning (Muthsam and Mayer, 1990). Expert systems are computer programs which simulate experts making decision and solving problems. Experts solve problems by using a combination of factual knowledge and reasoning ability. In an expert system, these two essentials are contained in two separate but related components, a knowledge base and an inference engine. The knowledge base provides specific facts about the subject and the inference engine provides the reasoning ability that enables the expert system to form conclusions (Waterman, 1986).

In expert systems, production rules, semantic nets and frames are used to represent knowledge in the knowledge base. Production rules can be expressed simply in the form of IF-THEN rules: IF [conditions] THEN [actions]. Examples of systems using rule representation are TIPPS (Chang, 1982) and QTC (Chang et al, 1988). Semantic nets attempt to describe the world in terms of objects and binary relations. For example, a hole and a drill are objects and their binary relation can be described as machined-by. Some examples using semantic nets are the systems described by Inui et al (Inui et al, 1987), Henderson and Anderson (Henderson and Anderson, 1984) and Tsang (Tsang, 1987). A frame is a network of nodes and relations organised in a hierarchical structure that represents procedural and

declarative information in terms of attributes, hierarchical relations with other frames, constraints, default values and procedures. A frame consists of several named slots. Each slot can store values, pointers to other frames or to a procedure. When a slot is filled, it is said to be initialised. The slot value can be restricted to a certain range or it must come from a set of values. Some systems using frame representation are SIPP (Nau, 1985), SIPS (Nau and Gray, 1986) and GEFPOS (Vissa, 1987).

The inference mechanism of an expert system "reasons" while using the knowledge of the knowledge base and thus tries to solve the given problems. The basic idea behind an inference mechanism is to search for a combination of rules which solve a given problem when applied to that problem in a certain order. There are two different ways to find such a combination of rules (Waterman, 1986):

- forward chaining,
- backward chaining.

The forward chaining method searches for rules which are triggered by the initial state of the problem. The application of such a rule will change the state of the problem by adding information or modifying its description thus creating a new state. This process is repeated until the goal state is reached. Consequently, the solution of the problem can be derived from the selected rules.

In backward-chaining the method searches for rules whose conclusions correspond to the goal state of the problem. It subsequently tries to meet the conditions of these rules. Therefore, the inference mechanism may activate new rules until every condition of an activated rule is proven to be true. In this context "reasoning" is synonymous to applying rules from the knowledge base. Common-sense or heuristic information can then be used to determine which rules should be "fired", i.e. activated.

When building an expert system any programming language or expert system shell can be used. An expert system shell is an expert system which is stripped of its knowledge. Since the system structure is already built, the builder of an expert system need only insert the knowledge. Because they have a flexible structure and flexible inference strategy, the employment of expert system building tools is better than other methods. The principal

advantage of an expert system building tool is the relatively short period of time in which a prototype of a new expert system can be developed. Most of the system's utilities are already available and the expert system builder can concentrate on gathering, processing and entering the domain specific knowledge. The expert system shell used in this project is ART-IM.

Since the introduction of expert system technology to process planning, a few expert process planning systems have been proposed. SIPP (Nau, 1985), SIPS (Nau and Gray, 1986), TOM (Matsushima et al, 1982), FirstCut (Cutkosky et al, 1988) and QTC(Chang et al, 1988) are some examples. SIPP, SIPS, TOM and FirstCut systems work on each individual feature following the sequence given by the user. Thus, they are not complete systems. QTC system has considered the relationship of features during planning. However, it deals with only a limited number of features.

Muthsam and Mayer(1990), Smith(1990), Trmal et al(1992) and Singh and Raman(1992) also discussed the expert system applications in machining. There are some systems which have been developed for one of the process planning tasks such as process selection (Nau and Chang, 1983), tool selection (Gusti et al, 1986), fixture design and set up (Englert and Wright, 1988), process sequencing and operation planning(Barkocy and Zdeblick, 1984), machining data selection(Wang and Wysk, 1986), and CNC milling(Priess and Kaplansky, 1984).

One main problem with current generative CAPP systems is the lack of a complete component representation. Ideally, the component description would be extracted from a CAD database and the output directed to the process planning system. This method can realise automated process planning. However, as described in the previous section, current CAD systems lack semantic information in the data structures for identifying the manufacturing significance. The special descriptive languages also have problems because they have to be manually prepared for input into the CAPP systems. For complex components this can be tedious and difficult.

It seems that GT classification and coding schemes are currently the most suitable representation model. However, existing GT coding schemes cannot provide the exact shape

and size information because of their finite digits. Thus, a more comprehensive GT classification and coding scheme is required.

Besides the lack of complete component representation the effective planning knowledge representation is still missing. The complexity of the planning procedure and the number of important parameters, such as cutting speed and depth, make it difficult to build an effective generation model. This research work aims to make a contribution in this area.

Chapter 3. The New Classification and Coding Scheme

3.1 Structure of the New Coding Scheme

As discussed in Chapter 2, a new coding scheme is required to provide adequate component description for the coding scheme based generative CAPP expert system (CSB-GCAPPES) developed in the research project. The work in this project is restricted to prismatic components to contain the size of the task. To satisfy this requirement the code generated by the new coding scheme must carry information similar to an engineering drawing, such as component shape, material, dimensions, tolerance, surface finish and orientation. The code generated by the new coding scheme has four parts (divided by four brackets [ ]) which represent the overall description, external machined shapes, the holes and the internal machined shapes other than holes(see Figure 3.1). Clearly the CSB-GCAPPES has to take into account the component description to create an effective process plan for machining the component. Thus the code for a component serves as input to the CSB-GCAPPES, whereby the system analyses the code to formulate a process plan.

The new coding scheme provides three levels of design and manufacturing information to identify, define and classify the prismatic component (see Figure 3.2). These three levels are the overall description (level 1), the machined shapes description (level 2) and the machined shape's attributes description (level 3). Level 1 (overall description) describes the overall shape of the component, the maximum length and the material. Level 2 (the machined shapes) describes functional shapes such as straight hole, plane surface, step, slot, pocket, etc. Level 3 information describes the machined shape's attributes such as dimension, tolerance, surface finish and orientation.

The new code has a hybrid structure utilising both the monocode and the polycode structures. Digits are separated by commas. The digits of level 1(overall description) have the polycode structure with each digit being completely independent of all other digits. The level 2 and level 3 digits have the monocode structure with the digits in the level 3 being dependent on the digits in the level 2, i.e. the number of attributes and the type of attributes of machined shape are decided by the machined shape digit. The attributes of each machined shape have been set up in the new coding scheme.

The new GT coding scheme does not have a fixed number of digits, which is different to most existing coding schemes. One advantage of the new coding scheme over existing coding schemes is that it can describe each machined shape. The flexible digit length structure ensures the new code carries enough information for process planning. Each digit of the new code uses Arabic numbers (not just 0 to 9 which are used in most existing coding scheme) to fully present the component's design and manufacturing information. The value of the digit and its position in the code provides certain information. The following two sections gives a detailed description of the new coding scheme and typical component examples.

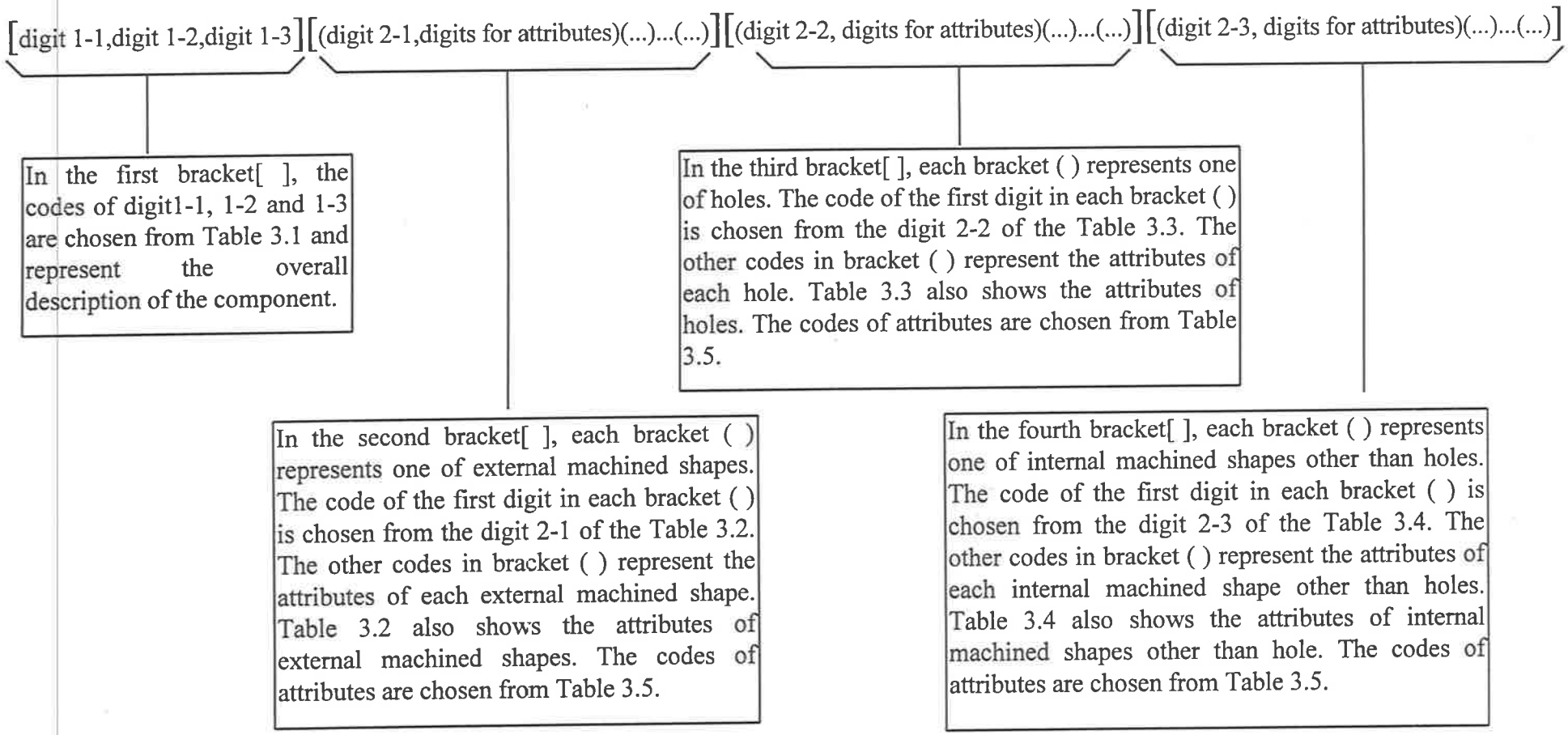

Figure 3.1 The New Coding Scheme Structure

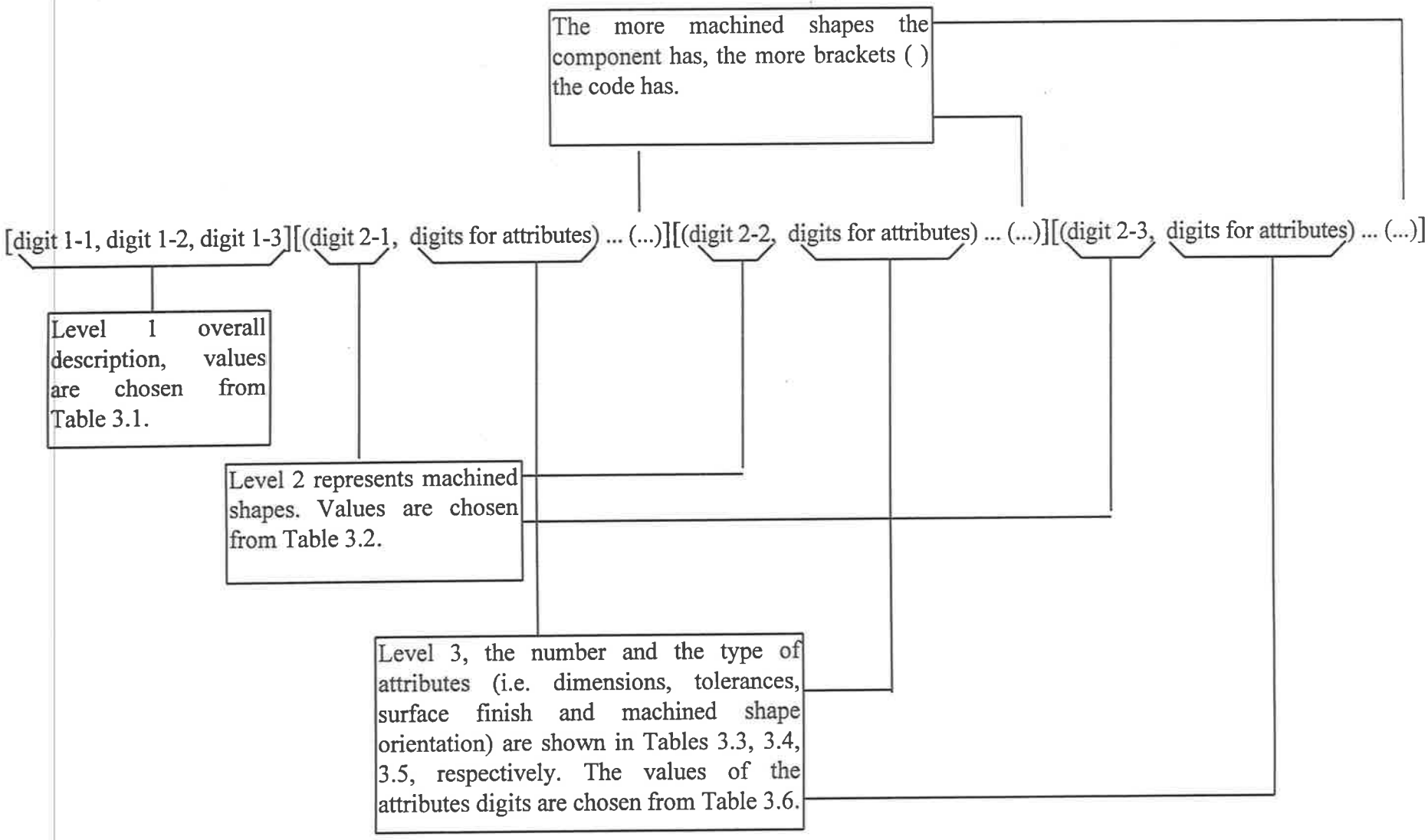

Figure 3.2 The Three Levels of the New Coding Scheme

3.2 The Proposed New Coding Scheme

3.2.1 Level One of The Coding Scheme

The first three digits of the code are the first level of component information which gives the overall description including overall shape, maximum length and material of finished component. Table 3.1 shows the meaning of each value which represent the overall shape, maximum length and material of the component.

Table 3.1 Level One: Overall Description

Digit 1-1 Overall Shape(A=length, B=width, C=height)		**Digit 1-2 Maximum Length (mm)**		**Value 1-3 Component Material**	
Code	Description	Code	Description	Code	Description
0	don't know	0	not clear	0	don't know
1	Cubic Components, A/B<=3, A/C<=4	1	A<=16	1	Mild Steel
2	Flat Components, A/B<=3, A/C>4	2	16<A<=50	2	Hardened Steel
3	Long Components, A/B>3	3	50<A<=100	3	Stainless Steel
		4	100<A<=160	4	Tool Steel
		5	160<A<=240	5	Cast Iron
		6	240<A<=360	6	Heat Resistant Alloys
		7	360<A<=600	7	Titanium
		8	600<A<=1000	8	Aluminium
		9	1000<A<=2000	9	Brass
		10	2000<A	10	Plastic

3.2.2 Level Two and Level Three of The Coding Scheme

The shape definition is the level 2 information of a component. In the second level of the coding scheme, the representation of the external machined shapes, holes and internal

machined shapes are given by the digit 2-1 of Table 3.2, the digit 2-2 of Table 3.3 and the digit 2-3 of Table 3.4.

Every machined shape has its function, dimensions, tolerance, surface finish and orientation attributes which determine the choice of machine, cutting tools and cutting conditions, etc. The attributes of each machined shape are the level 3 information of a component. In the new coding scheme, the number and the type of attributes for each code of digit 2-1, digit 2-2 and digit 2-3 are given in Table 3.2, Table 3.3 and Table 3.4, respectively. For example, the straight hole in Table 3.3 has five attributes: diameter, length, dimension tolerance, surface finish and orientation. Thus, the new coding scheme uses six digits (i.e. the straight hole digit and its five attributes) to represent the straight hole on the component. Because the number and the type of the attribute is dependent on the digit of the machined shapes, i.e. digit 2-1, 2-2 or 2-3, the level two and the level three are monocode structures.

The number, the type and the sequence of the attributes of each machined shape have been fixed in the new coding scheme as shown in Table 3.2, 3.3 and 3.4 though they may be changed when used in a different company. As required for process planning the information of attributes for each machined shape is gained from the work shop.

The codes of the attributes for the machined shapes are chosen from Table 3.5. For example, an external plane surface has a length of 138mm. From Table 3.5 the code number for the dimension is 138. When choosing the code number for tolerance, Table 3.6 may be used. For example, two holes have the distance of 88mm. The position tolerance value is 50μm. From Table 3.6 the tolerance grade is grade 8. Then, from Table 3.5 the code number for grade 8 is 10.

Table 3.2 External Machined shapes and Attributes

Digit 2-1 External Machined Shapes		**Attributes**					
Code	**Description**						
0	**No**						
1	**Plane Surface**	Width	Length	Dimension Tolerance	Surface Finish	Orientation	
2	**Step**	Width	Height	Length	Dimension Tolerance	Surface Finish	Orientation
3	**Slot**	Width	Height	Length	Dimension Tolerance	Surface Finish	Orientation
4	**Pocket**	Width	Height	Length	Dimension Tolerance	Surface Finish	Orientation
5	**Contour Flank**	Minimum Radius Of Concave	Height	Dimension Tolerance	Surface Finish	Orientation	
6	**Sculptured Surface**	Width	Length	Dimension Tolerance	Surface Finish	Orientation	

Table 3.3 Holes and Attributes

Digit 2-2 Holes		Attributes								
Code	Description									
0	No									
1	Straight Hole	Diameter	Length	Dimension Tolerance	Surface Finish	Orientation				
2	One Side Stepped	Large Diameter	Length of Large Diameter	Small Diameter	Length of Small Diameter	Dimension Tolerance	Surface Finish	Orientation		
3	Two Side Stepped	Large Diameter	Length of Large Diameter	Middle Diameter	Length of Middle Diameter	Small Diameter	Length of Small Diameter	Dimension Tolerance	Surface Finish	Orientation
4	Circular Groove	Hole Diameter	Length of the Hole	Groove Diameter	Dimension Tolerance	Surface Finish	Orientation			

Table 3.4 Internal Machined Shapes Other Than Holes and Attributes

Digit 2-3 Internal Machined Shapes		Attributes					
Code	Description						
0	No						
1	Plane Surface	Width	Length	Depth	Dimension Tolerance	Surface Finish	Orientation
2	Contour Flank	Minimum Radius of Concave	Height	Dimension Tolerance	Surface Finish	Orientation	
3	Sculptured Surface	Width	Length	Dimension Tolerance	Surface Finish	Orientation	

In Table 3.5, the codes corresponding to the machined shape attributes such as the machined shape's dimensions, tolerances, surface finish and orientation are represented. The attribute values of dimension are chosen from the first column of Table 3.5. The first column of Table 3.5 has the values from 0 to N (i.e. any integer) to represent the value of the dimension attribute. If the dimension of the machined shape is not an integer, the digit value is to the nearest integer. For example, if the dimension is 25.5mm, the digit value will be 26.

The tolerance attribute values are chosen from the second column of Table 3.5. The eighteen tolerance grades correspond with Australia Standard AS 1654 and the magnitude of tolerance depends on the basic dimension of the feature. Some tolerances for common used grades are provided in Table 3.6.

The surface finish attribute values are chosen from the third column of Table 3.5. For example, if the Ra values of the surface is 1.6μm, the digit value will be code 7.

The orientation attribute values are chosen from the fourth column of Table 3.5. In the new coding scheme, the orientation attribute is related to the orientation which is commonly used in most CAD software such as AutoCAD. For example, the orientation Front (digit code 2) is defined to include the length and the height. Thus, the new coding scheme has the ability to integrate with the CAD database. However, the task is not included in this research project because of the time limitation.

Table 3.5 Code for Machined Shape's Attribute

Code for dimension(mm)	
code	Dimension (mm)
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
...	...
N	N

Code for tolerance	
Code	Tolerance Grade
0	no requirement
1	Grade 01
2	Grade 0
3	Grade 1
4	Grade 2
5	Grade 3
6	Grade 4
7	Grade 5
8	Grade 6
9	Grade 7
10	Grade 8
11	Grade 9
12	Grade 10
13	Grade 11
14	Grade 12
15	Grade 13
16	Grade 14
17	Grade 15
18	Grade 16

Code for surface finish	
Code	Ra Value in μm
0	no requirement
1	0.025
2	0.05
3	0.1
4	0.2
5	0.4
6	0.8
7	1.6
8	3.2
9	6.3
10	12.5
11	25
12	50

Code for orientation	
Code	Orientation
0	do not know
1	Top
2	Front
3	Right
4	Left
5	Back
6	Bottom

Table 3.6 Tolerances For Common Used Grades (reprint from AS 1654)

Basic Size(mm)		Tolerance Value(μm)						
Above	Up to and Incl.	Grade 6	Grade 7	Grade 8	Grade 9	Grade 10	Grade 11	Grade 12
0	3	6	10	14	25	40	60	100
3	6	8	12	18	30	48	75	120
6	10	9	15	22	36	58	90	150
10	18	11	18	27	43	70	110	180
18	30	13	21	33	52	84	130	210
30	50	16	25	39	62	100	160	250
50	80	19	30	46	74	120	190	300
80	120	22	35	54	87	140	220	350
120	180	25	40	63	100	160	250	400
180	250	29	46	72	115	185	290	460
250	315	32	52	81	130	210	320	520
315	400	36	57	89	140	230	360	570
400	500	40	63	97	155	250	400	630
500	630	44	70	110	175	280	440	700

3.3 Two Typical Examples of the New Coding Scheme

3.3.1 A Cubic Prismatic Component Example

Figure 3.3 shows a cubic prismatic component and its code. The component has seven types of external machined shapes: top plane surface (64×138), left and right plane surfaces (44×50), front and back plane surfaces (semi-donuts), and two type steps (50×17×5, left and right). Actually on the left and right sides of the component there are two steps on each side (see Figure 3.3). However, the two steps on one side have the same size and can be considered as a one type step. Thus, in the second part of the code (parts of a code are separated by brackets [ ]) there are seven pairs of brackets ( ) and each brackets represents one type of external machined shape. Within each bracket ( ) the code digits are separated by commas and the code number represents the machined shape definition and its attributes (the first digit number always represents the machined shape definition).

Though the sequence of machined shape descriptions, i.e. the brackets ( ) within a bracket [ ]of a code, do not affect the information carried by the code, the sequence is decided by comparing the first digit of each bracket ( ). The bracket ( ) with the smaller first digit number is ahead of others. If the first digit numbers are same, the following digit numbers will be compared respectively. For example, in Figure 3.3 the (1,13,101,0,9,2) is ahead of (1,13,101,0,9,5) and the (1,44,50,0,0,3) is ahead of the (1,64,138,0,8,1).

The component also has three types of holes, a ϕ38mm semicircular hole with a R3mm groove, ϕ13mm through holes (two) and ϕ5mm blind holes (two). As described in the above paragraph the same size and orientation of the machined shapes are considered as one type, such as the two ϕ13mm holes which use the same cutting tool. Thus, there are three pairs of brackets ( ) which represents the three type holes in the third part of the code (see Figure 3.3). Similarly, within each bracket ( ) the code digits are separated by commas and the code number represents the machined shape definition and its attributes (the first digit number always represents the machined shape definition).

The cubic component is defined as one having no internal shape other than holes. The code of the cubic prismatic component using the new classification and coding scheme is as follows:

1. The code in the first bracket [1,4,1] describes the overall component. The first digit 1 means the overall shape has a length to width ratio < = 3 and the length to overall height ratio < = 4 and is chosen from the first column of Table 3.1. The second digit 4 is chosen from the second column of Table 3.1, the maximum length, while the third digit 1 is chosen from the third column of Table 3.1, the component material.

2. As seen in Figure 3.3, the second, third and fourth parts of the code (parts of a code are separated by brackets [ ]) represents the external machined shapes, holes and internal machined shapes other than holes. The second part has seven pairs of brackets ( ), i.e. (1,13,101,0,9,2) (1,13,101,0,9,5) (1,44,50,0,0,3) (1,44,50,0,0,4) (1,64,138,0,8,1) (2,17,5,50,0,0,3) (2,17,5,50,0,0,4). The first digit number in each pair of brackets ( ) is chosen from the first column of Table 3.2 and the other digit numbers in each pair of brackets ( ) represents the attributes of the machined shape. For example, in the first pair of brackets, i.e. (1,13,101,0,9,2), number 1 represents the plane surface. Also Table 3.2 shows the plane surface has five attributes: width, length, dimension tolerance, surface finish and orientation. Thus, number 13, 101, 0, 9, 2 represents the width, the length, the dimension tolerance, the surface finish and the orientation, respectively. The numbers 13 and 101 represents dimensions and are chosen from the first column of Table 3.5. The numbers 0, 9 and 2 are chosen from the second, the third and the fourth column of Table 3.5. The same procedure is followed for all machined surfaces identified in Figure 3.3.

The third part in the code of Figure 3.3 has three pairs of brackets ( ), i.e. (1,5,15,0,0,2) (1,13,50,0,8,1) (4,38,64,6,0,7,2).The first digit number in each pair of brackets ( ) is chosen from the first column of Table 3.3 and the other digit numbers in each pair of brackets ( ) represents the attributes of the machined shape. For example, in the first pair of brackets, i.e. (1,5,15,0,0,2), number 1 represents the straight hole. Also Table 3.3 shows the straight hole has five attributes: diameter, length, dimension tolerance, surface finish and orientation. Thus, numbers 5, 15, 0, 0 and 2 represents the diameter, the length, the dimension tolerance, the surface finish and the orientation, respectively. The numbers 5 and 15 represent

dimensions and are also chosen from first column of Table 3.5. The numbers 0, 0 and 2 are chosen from the second, the third and the fourth column of Table 3.5.

The fourth part in the code of Figure 3.3, i.e.[(0)], is the final part. The number 0 is chosen from the first column of Table 3.4 and means the cubic component has no internal machined shape other than holes.

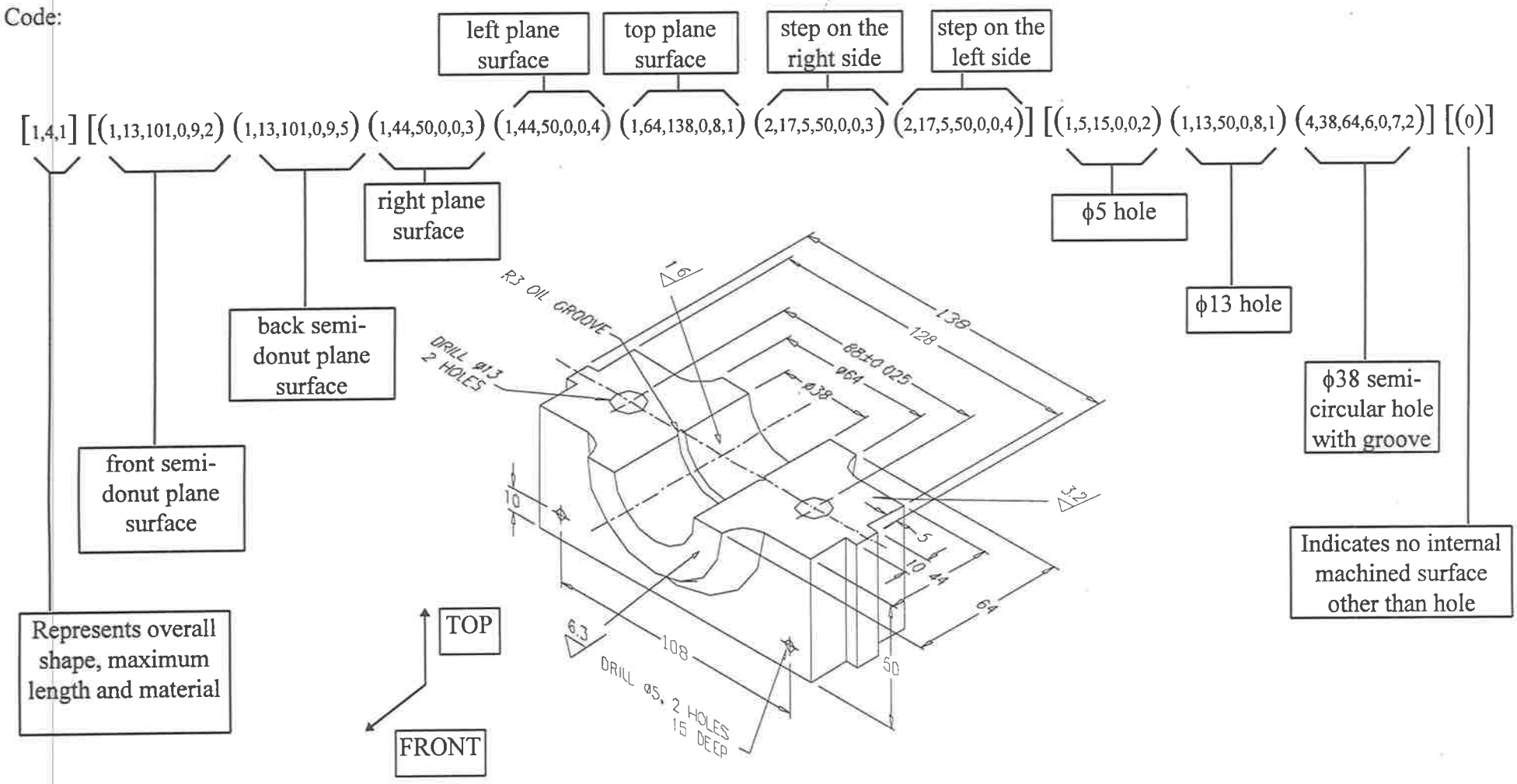

Figure 3.3 A Cubic Prismatic Component

3.3.2 A Flat Prismatic Component Example

Figure 3.4 shows a flat prismatic component and its code. The component has three types of external machined shapes: top and bottom plane surface (100×285), and external contour flank. Thus, in the second part of the code (parts of a code are separated by brackets [ ]) there are three pairs of brackets () and each bracket represents one type of external machined shape. Within each bracket ( ) the code digits are separated by commas and the code number represents the machined shape definition and its attributes (the first digit number always represents the machined shape definition).

The component also has two types of holes, ϕ15mm through holes (six) and a ϕ30mm through hole. As described in the last section the same size and orientation of the machined shapes are considered as one type, such as the six ϕ15mm holes which use the same cutting tool. Thus, there are two pairs of brackets ( ) which represent the two types of holes in the third part of the code (see Figure 3.4). Similarly, within each bracket ( ) the code digits are separated by commas and the code number represents the machined shape definition and its attributes (the first digit number always represents the machined shape definition). The component has one type of internal machined shape other than the holes, and the five internal contour flanks.

The code of the flat prismatic component using the new classification and coding scheme is as follows:

1. The code in the first bracket [2,6,1] describes the overall component. The first digit 2 means the overall shape has a length to width ratio < = 3 and the length to overall height ratio > 4 and is chosen from the first column of Table 3.1. The second digit 6 is chosen from second column of Table 3.1, the maximum length, while the third digit 1 is chosen from the third column of Table 3.1, the component material.

2. As seen in Figure 3.4, the second, third and fourth parts of the code (parts of a code are separated by brackets [ ]) represents the external machined shapes, holes and internal machined shapes other than holes. The second part has three pairs of brackets ( ), i.e. (1,100,285,0,0,1) (1,100,285,0,0,6) (5,40,12,0,0,2). The first digit number in each pair of

brackets ( ) is chosen from the first column of Table 3.2 and the other digit numbers in each pair of brackets ( ) represent the attributes of the machined shape. For example, in first pair of brackets, i.e. (1,100,285,0,0,1), number 1 represents the plane surface which is chosen from the first column of Table 3.2. Also Table 3.2 shows that the plane surface has five attributes: width, length, dimension tolerance, surface finish and orientation. Thus, number 100, 285, 0, 0 and 1 represents the width, the length, the dimension tolerance, the surface finish and the orientation, respectively. The numbers 100 and 285 represent dimensions and are chosen from the first column of Table 3.5. The numbers 0, 0 and 1 are chosen from the second, the third and the fourth column of Table 3.5, respectively. The same procedure is followed for all machined surfaces identified in Figure 3.4.

The third part in the code of Figure 3.4 has two pairs of brackets ( ), i.e. (1,15,12,0,0,1) (1,30,12,0,0,1).The first digit number in each pair of brackets ( ) is chosen from the first column of Table 3.3 and the other digit numbers in each pair of brackets ( ) represent the attributes of the machined shape. For example, in the first pair of brackets, i.e. (1,15,12,0,0,1), number 1 represents the straight hole. Also Table 3.3 shows the straight hole has five attributes: diameter, length, dimension tolerance, surface finish and orientation. Thus, number 15, 12, 0, 0 and 1 represent the diameter, the length, the dimension tolerance, the surface finish and the orientation, respectively. The numbers 15 and 12 represents dimensions and are also chosen from the first column of Table 3.5. The numbers 0, 0 and 1 are chosen from the second, the third and the fourth column of Table 3.5.

The fourth part in the code of Figure 3.4, i.e.[(2,5,12,0,0,1)], is the final part. The fourth in the code of Figure 3.4 has one pair of brackets ( ).The first digit number 2 is chosen from the first column of Table 3.4 and the other digit numbers in each pair of brackets ( ) represents the attributes of the machined shape. Also Table 3.4 shows the internal contour flank has five attributes: minimum radius of concave, height, dimension tolerance, surface finish and orientation. Thus, numbers 5, 12, 0, 0 and 1 represents the minimum radius of concave, the height, the dimension tolerance, the surface finish and the orientation, respectively. The numbers 5 and 12 represent dimensions and are also chosen from the first column of Table 3.5. The numbers 0, 0 and 1 are chosen from the second, the third and the fourth column of Table 3.5.

The code:

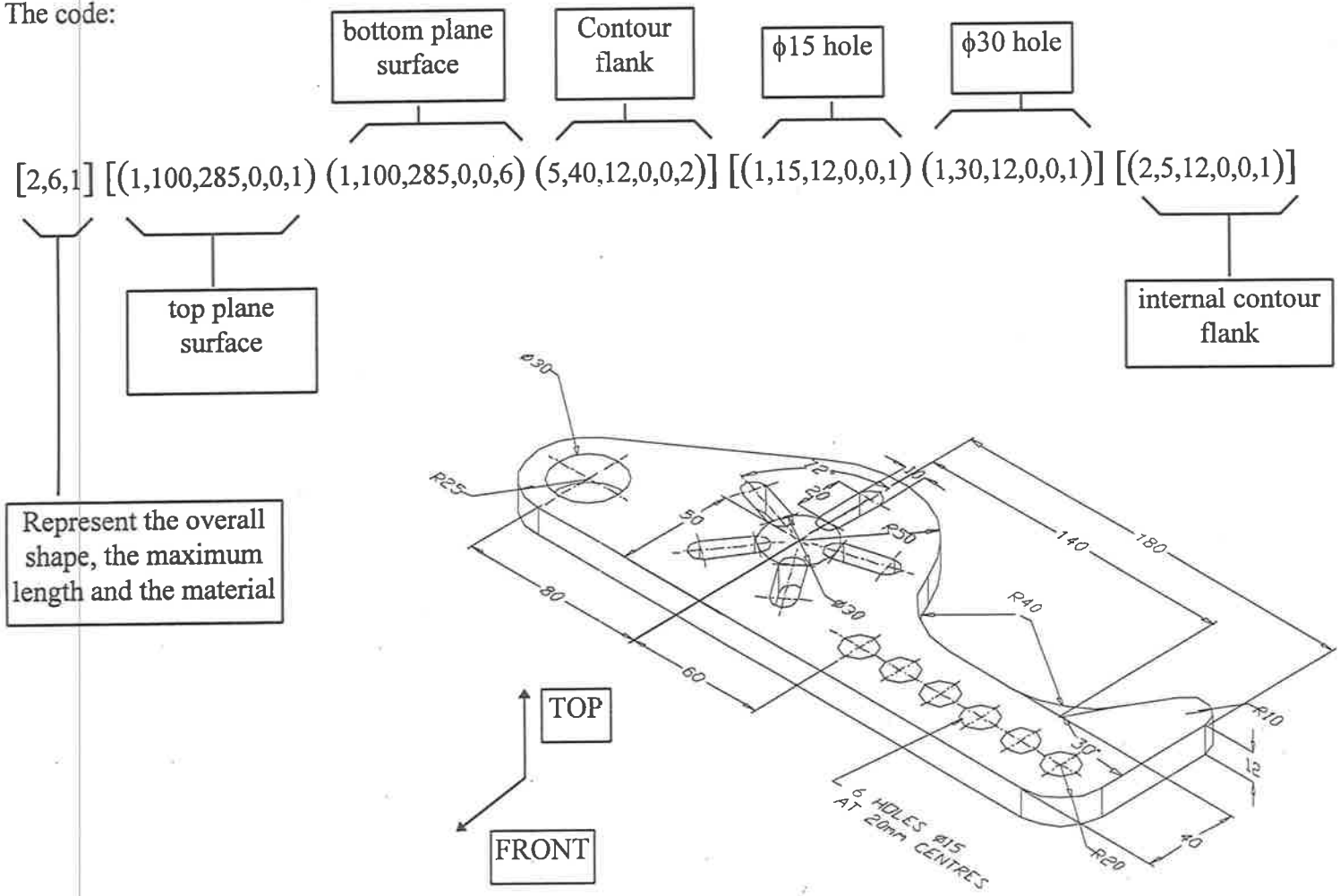

Figure 3.4 A Flat Prismatic Component

Chapter 4. The Coding Scheme Based Generative CAPP Expert System (CSB-GCAPPES)

4.1 Overview of the CSB-GCAPPES

As stated in Chapter 3 the goal of the CSB-GCAPPES is to assist the machinist in determining an optimal process plan for a prismatic component. The reason for restricting the CSB-GCAPPES to prismatic components is to make the procedure more tractable and because there are currently not many generative CAPP systems for prismatic component on the market. Compared to cylindrical components, prismatic components have more complex manufacturing features, which makes the choice of optimum cutting tools and machines more difficult. It is desirable to automate as much of the planning decision-making as possible. CSB-GCAPPES assumes that the users have the design requirements of the components and the relevant engineering drawings to hand. Also the system must have access to data like: machine limitations, tooling availability and other process-related manufacturing information. This data should be available from a computer knowledge base to develop the detailed process plan.

The CSB-GCAPPES consists of eight modules as in the Figure 4.1. The eight process planning modules which are built in CSB-GCAPPES to generate a detailed process plan are: coding, blank selection, process selection, machine selection, cutting tool selection, cutting condition(cutting speed, feed rate and cutting depth) selection, process sequencing and process documentation. A module is a part of the system with a well defined function and well defined interfaces to both the other modules of the system and the users. Every module in the CSB-GCAPPES is based on the knowledge of basic machining processes, cutting tools, NC milling machines and cutting conditions such as cutting speed, feed rate and cutting depth. This knowledge comes from work-shop experience and reference materials.

The CSB-GCAPPES generates the code of the component automatically and stores the code in the knowledge base. The blank selection module decides on a blank size based on the overall component shape and the maximum length code digit from level 1. The process selection module decides the working operations based on the level two information

carried by the code. The machine, cutting tool and cutting condition selection modules select the best choice for each operation based on the level three digit values of the code. The process sequencing module decides the best sequence of operations depending on the minimum component setups and the minimum tool changes, Finally, the process document module produces a detailed process plan for the component.

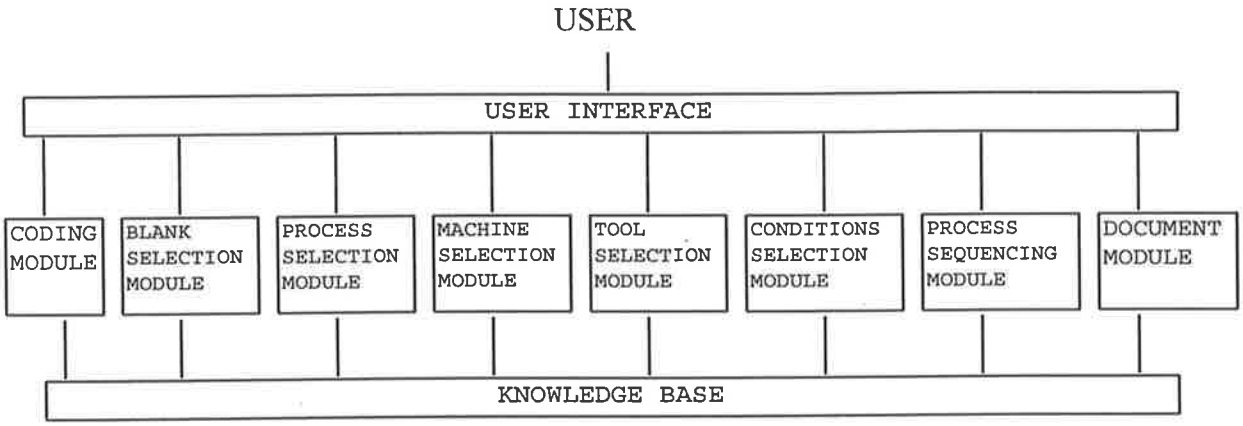

Figure 4.1 The Coding Scheme Based Generative CAPP Expert System(CSB-GCAPPES)

The knowledge base is accessible by all modules. For building the CSB-GCAPPES and its knowledge base, an expert system building tool, ART-IM, is used.

In CSB-GCAPPES, the user interface is a window-based input and output. The users can easily interface with the system using the computer keyboard. A mouse oriented environment is desired but because of the time limitation it is recommended as future work.

4.2 Knowledge Representation of the CSB-GCAPPES

The tooling knowledge including tools and machines knowledge are encoded in the expert system as *facts* and *schema*. A *fact* is a fundamental piece of knowledge sitting in the knowledge base. For example, fact number 2 says that the cutting tool is a face-milling cutter

with ϕ15mm diameter. This is encoded as: f-2 (cutting-tool face-milling-cutter diameter-ϕ15mm). Thus, a *fact* is just a simple statement linking various pieces of information.

A *schema* is a collection of *facts* that represents a single object (or class of objects). A *schema* is similar to a frame representation which includes some *facts*. A frame may include several slots (a slot is usually a piece of information) and can also be related to other frames. The *facts* in a *schema* are like slots in a frame. For instance, a group of face-milling cutters can be represented easily as:

```
(defschema face-milling-cutter
    (process-capability plane-surface)
    (minimum-diameter 40))
```

In this example, the *schema* face-milling-cutter includes the information that all face-milling cutters can produce the plane surface and that the minimum diameter of face-milling cutters is 40mm.

In CSB-GCAPPES, the operational knowledge is expressed by *rules*. A *rule* is an "IF-THEN" statement similar in principle to those in BASIC or FORTRAN programming languages. More precisely, a *rule* is defined as a set of conditions associated with a proposed set of actions. The conditions must be *facts* or *schema*. A *rule* is "fired" when the conditions are matched in the knowledge base. A *rule* is "fired" means any action specified in the rule will be taken. For example, suppose a rule exists to select a face-milling cutter to produce a plane surface, CSB-GCAPPES uses the following rule:

```
(defrule plane-surface-cutting-tool-selection
        (geometrical-shape plane-surface)
        (material mild-steel)
        =>
        (assert(mild-steel plane-surface (face-milling-cutter material-tool-steel)))
        (printout t "The cutting tool for the mild steel plane surface is a face-milling cutter
 whose material is tool steel" t))
```

The symbol (=>) is used in ART-IM to separate the conditions pattern and the action portions of a *rule*. The actions in this rule are to add the new *fact*: (mild-steel plane-surface (face-milling-cutter material-tool-steel)), to the knowledge base and print on the screen the sentence: 'The cutting tool for the mild steel plane surface is a face-milling cutter whose material is tool steel.

In the CSB-GCAPPES, the process knowledge is represented by the *facts*, the *schemas* and the *rules* and thus is knowledge about the capabilities of manufacturing processes. Machines and cutting tools decide the capabilities of a manufacturing process. Thus, the knowledge base must also store all information about the available machines and cutting tools. Though the information of the machines and the cutting tools are different from one work-shop to another, a general form of information such as type and size of machines, material and geometry of cutting tools can be found. The CSB-GCAPPES stores the knowledge in generic terms so that it can be used in different work shops.

4.3 The Inference Engine of the CSB-GCAPPES

An expert system is a computer program which encodes the knowledge symbolically and consists of a knowledge base and an inference engine. The knowledge is contained in the knowledge base as explained in section 4.2, while the inference engine controls the reasoning to reach a conclusion, i.e. the inference engine determines the sequence in which the rules are applied. In the process planning expert system, the inference engine applies the process knowledge in the knowledge base to select the process, the machine and the cutting tools, etc.

The inference engine in the CSB-GCAPPES uses an "agenda cycle" algorithm available in ART-IM. This means that when the conditions of the rules are matched with *facts* and *schemas* in the knowledge base, the rules are placed on the agenda. The rules on the agenda can have a fixed or random firing order. After a rule fires, CSB-GCAPPES executes the rule. If the knowledge base has some alterations then the rules in CSB-GCAPPES are rechecked against the *facts* and *schemas* in the knowledge base. The agenda is cleared and the new rules which are matched with *facts* and *schemas* in the knowledge base are placed on the agenda

and another rule fires. Thus, the agenda is re-evaluated each time a rule fires and the "agenda cycle" continues until the users stops the CSB-GCAPPES or until the agenda runs out of activations.

Chapter 5. The CSB-GCAPPES PROTOTYPE

The capabilities of the prototype system are presented in this chapter by using two examples. The two examples demonstrates the basic principles of the prototype system and illustrates the generation of the code of components and process plan for component production.

The eight functional modules of the generative CAPP expert system analyses the code of the component and determines which sequence of processes and cutting tools are capable of producing the component. Thus, the list of processes and cutting tools generated by the modules match the component's design and manufacturing requirements with the capabilities of the available manufacturing processes.

5.1 The Automatic Coding Module

The automatic coding (code input) module accepts the component's design information (component code value of each digit) and stores the code in the knowledge base so that other modules can generate the process plan.

The coding module automatically provides the list of each digit as shown in Figure 5.1 to the users on the screen. The users chooses one of values according to the component design description or engineering drawings and keys in each value of the code. Thus, even if the user does not know the new coding scheme he/she can generate a component code. The code selection procedure is the same as the manual procedure which is described in Chapter 3.

When all the values have been selected the coding module stores the code in the knowledge base as *facts* and *schemas* (Table 5.1) and also saves it as a text file on the hard drive for future use.

The code of the cubic component, which is shown in Figure 3.3, is [1,4,1] [(1,13,101,0,9,2) (1,13,101,0,9,5) (1,44,50,0,0,3) (1,44,50,0,0,4) (1,64,138,0,8,1) (2,17,5,50,0,0,3) (2,17,5,50,0,0,4)] [(1,5,15,0,0,2) (1,13,50,0,8,1) (4,38,64,6,0,7,2)] [(0)]. The representation of the code in the knowledge base is shown in Table 5.1.

The code of the flat component, which is shown in Figure 3.4, is [2,6,1] [(1,100,285,0,0,1) (1,100,285,0,0,6) (5,40,12,0,0,2)] [(1,15,12,0,0,1) (1,30,12,0,0,1)] [(2,5,12,0,0,1)]. The representation of the code in the knowledge base is shown in Table 5.2.

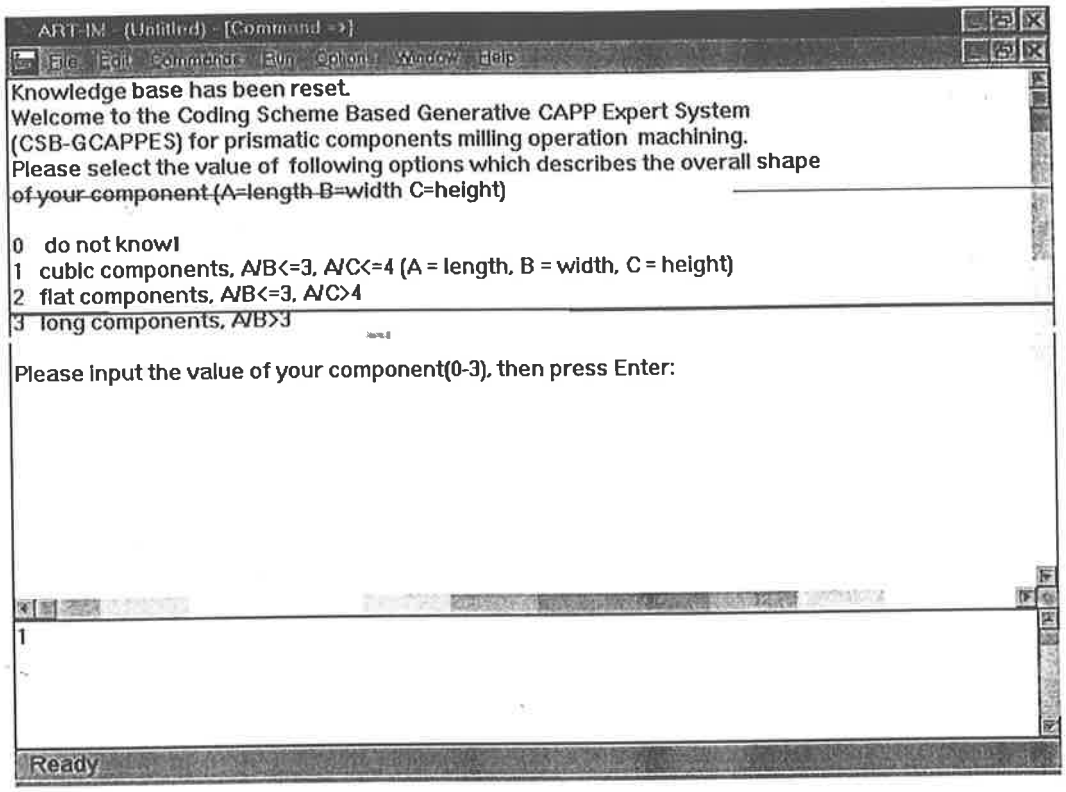

Figure 5.1 Interface windows

Table 5.1 The *Facts* of the Cubic Component Example Code Values

[1,4,1] [(1,13,101,0,9,2) (1,13,101,0,9,5) (1,44,50,0,0,3) (1,44,50,0,0,4) (1,64,138,0,8,1) (2,17,5,50,0,0,3) (2,17,5,50,0,0,4)] [(1,5,15,0,0,2) (1,13,50,0,8,1) (4,38,64,6,0,7,2)] [(0)]

The *Facts* of the Level 1 Code Digit Values
(digit-1-1-overall-shape 1), (digit-1-2-maximum-length 4), (digit-1-3-material 1)
The *Facts* of the External Machined Shapes Code Values
(digit-2-1 1 1), (digit-2-1-1-width 13 1), (digit-2-1-1-length 101 1), (digit-2-1-1-dimension-tolerance 0 1), (digit-2-1-1-surface-finish 9 1), (digit-2-1-1-orientation 2 1). (digit-2-1 1 2), (digit-2-1-1-width 13 2), (digit-2-1-1-length 101 2), (digit-2-1-1-dimension-tolerance 0 2), (digit-2-1-1-surface-finish 9 2), (digit-2-1-1-orientation 5 2). (digit-2-1 1 3), (digit-2-1-1-width 144 3), (digit-2-1-1-length 50 3), (digit-2-1-1-dimension-tolerance 0 3), (digit-2-1-1-surface-finish 0 3), (digit-2-1-1-orientation 3 3). (digit-2-1 1 4), (digit-2-1-1-width 144 4), (digit-2-1-1-length 50 4), (digit-2-1-1-dimension-tolerance 0 4), (digit-2-1-1-surface-finish 0 4), (digit-2-1-1-orientation 4 4). (digit-2-1 1 5), (digit-2-1-1-width 64 5), (digit-2-1-1-length 138 5), (digit-2-1-1-dimension-tolerance 0 5), (digit-2-1-1-surface-finish 8 5), (digit-2-1-1-orientation 1 5). (digit-2-1 2 6), (digit-2-1-2-width 17 6), (digit-2-1-2-height 5 6), (digit-2-1-2-length 50 6), (digit-2-1-2-dimension-tolerance 0 6), (digit-2-1-2-surface-finish 0 6), (digit-2-1-2-orientation 3 6). (digit-2-1 2 7), (digit-2-1-2-width 17 7), (digit-2-1-2-height 5 7), (digit-2-1-2-length 50 7), (digit-2-1-2-dimension-tolerance 0 7), (digit-2-1-2-surface-finish 0 7), (digit-2-1-2-orientation 4 7).
The *Facts* of the Holes Code Values
(digit-2-2 1 1), (digit-2-2-1-diameter 5 1), (digit-2-2-1-length 15 1), (digit-2-2-1-dimension-tolerance 0 1), (digit-2-2-1-surface-finish 0 1), (digit-2-2-1-orientation 2 1). (digit-2-2 1 2), (digit-2-2-1-diameter 13 2), (digit-2-2-1-length 50 2), (digit-2-2-1-dimension-tolerance 0 2), (digit-2-2-1-surface-finish 8 2), (digit-2-2-1-orientation 1 2). (digit-2-2 4 3), (digit-2-2-4-diameter 38 3), (digit-2-2-4-length 64 3), (digit-2-2-4-groove-diameter 6 3), (digit-2-2-4-dimension-tolerance 0 3), (digit-2-2-4-surface-finish 7 3), (digit-2-2-1-orientation 2 3).
The *Facts* of the Internal Machined Shapes Code Values
(digit-2-3 0)

Table 5.2 The *Facts* of the Flat Component Example Code Values

[2,6,1] [(1,100,285,0,0,1) (1,100,285,0,0,6) (5,40,12,0,0,2)] [(1,15,12,0,0,1) (1,30,12,0,0,1)] [(2,5,12,0,0,1)]

The *Facts* of the Level 1 Code Digit Values
(digit-1-1-overall-shape 2), (digit-1-2-maximum-length 6), (digit-1-3-material 1)
The *Facts* of the External Machined Shapes Code Values
(digit-2-1 1 1), (digit-2-1-1-width 100 1), (digit-2-1-1-length 285 1), (digit-2-1-1-dimension-tolerance 0 1), (digit-2-1-1-surface-finish 0 1), (digit-2-1-1-orientation 1 1). (digit-2-1 1 2), (digit-2-1-1-width 100 2), (digit-2-1-1-length 285 2), (digit-2-1-1-dimension-tolerance 0 2), (digit-2-1-1-surface-finish 0 2), (digit-2-1-1-orientation 6 2). (digit-2-1 5 3), (digit-2-1-5-minimum-radius 40 3), (digit-2-1-5-height 12 3), (digit-2-1-5-dimension-tolerance 0 3), (digit-2-1-5-surface-finish 0 3), (digit-2-1-5-orientation 2 3).
The *Facts* of the Holes Code Values
(digit-2-2 1 1), (digit-2-2-1-diameter 15 1), (digit-2-2-1-length 12 1), (digit-2-2-1-dimension-tolerance 0 1), (digit-2-2-1-surface-finish 0 1), (digit-2-1-1-orientation 1 1). (digit-2-2 1 2), (digit-2-2-1-diameter 30 2), (digit-2-2-1-length 12 2), (digit-2-2-1-dimension-tolerance 0 2), (digit-2-2-1-surface-finish 0 2), (digit-2-2-1-orientation 1 2).
The *Facts* of the Internal Machined Shapes Code Values
(digit-2-3 2 1), (digit-2-3-2-minimum-radius 5 1), (digit-2-3-2-height 12 1), (digit-2-3-2-dimension-tolerance 0 1), (digit-2-3-2-surface-finish 0 1), (digit-2-3-2-orientation 1 1).

5.2 Blank Selection Module

The blank is the raw material from which the component is to be made. Blank selection depends on the component's geometry. At present CSB-GCAPPES is restricted to prismatic components and hence the blanks are prismatic i.e. they can be specified by X, Y and Z dimensions. Table 5.3 shows the blank data in the CSB-GCAPPES.

Table 5.3 Blank Data

Cubic Component (A/B<=3, A/C<=4)	Flat Component (A/B<=3, A/C>4)	Long Component (A/B>3)
A=16, B=10, C=5	A=16, B=6, C=3	A=16, B=5
A=50, B=25, C=20	A=50, B=25, C=10	A=50, B=16
A=100, B=50, C=35	A=100, B=50, C=15	A=100, B=33
A=160, B=100, C=60	A=160, B=100, C=20	A=160, B=53
A=240, B=120, C=100	A=240, B=120, C=25	A=240, B=79
A=360, B=180, C=150	A=360, B=180, C=30	A=360, B=119
A=600, B=250, C=200	A=600, B=250, C=50	A=600, B=199
A=1000, B=500, C=350	A=1000, B=500, C=100	A=1000, B=333
A=2000, B= 1000, C=800	A=2000, B= 1000, C=200	A=2000, B=666

The above data is stored in the knowledge base of CSB-GCAPPES as *facts* and *schemas*. The data is set to three groups of *schemes*: (i) Cubic-Prismatic-Blank, (ii)Flat-Prismatic-Blank and (iii) Long- Prismatic-Blank. Some of the *schemas* for blank data selection are:

```
(defschema Cubic-Prismatic-Blank
        (Length $?)
        (Width $?)
        (Height $?)
        (Materials $?))
```

The "$?" stands in place of several elements of a *fact* by a segment operator "$" with a wildcard "?".

If the data is 16, 10 and 5 then the *schema* becomes

```
(defschema Cubic- Prismatic-Blank-1
        (Length 16)
        (Width 10)
        (Height 5)
        (is-a  Cubic-Prismatic-Blank))
```

The "is-a" symbol is one of system-defined inheritance relations in ART-IM. It allows 'child' *schemas* to inherit *facts* from 'parent' *schemas*. In the above example the *schema*, Cubic- Prismatic-Blank-1, is a 'child' *schema* of Cubic-Prismatic-Blank.

If the data is 100, 50, and 15, then the *schema* becomes

```
(defschema Flat-Prismatic-Blank-3
        (Length 100)
        (Width 50)
        (Height 15)
        (is-a Flat-Prismatic-Blank))
```

The selection of blanks depends on level one values of the code, i.e. the overall shape, the maximum length and the material. The rules to choose the blank from the knowledge base depends on the type of components. For the cubic component example the rule is:

```
(defrule Component-Blank-Selection
        (digit-1-1-overall-shape 1)
        (digit-1-2-maximum-length 4)
        (digit-1-3-material 1)
=>
        (assert (component-blank
                        (is-a   Cubic-Prismatic-Blank-4))))
```

The term "assert" is a system-defined function which adds new *facts* or *schemas* to the knowledge base. The component blank data for the cubic component is shown in Table 5.4. The blank data is not only stored in the knowledge base, but also saved as a text file.

Table 5.4 The Blank *Facts* for the Cubic Component Example

The Blank *Facts* for Cubic Component
(length 160)
(width 100)
(height 60)
(material mild-steel)

Similarly, the component blank *facts* for the flat component are shown in Table 5.5. Thus, the blank data of the new component has been stored in the knowledge base to be used by the rules in other modules such as the process selection module and the plan documentation module.

Table 5.5 The Blank *Facts* of the Flat Component Example

The Blank *Facts* for Flat Component
(length 360)
(width 180)
(height 30)
(material mild-steel)

5.3 The Process Selection Module

A process is defined as the ability to produce a particular component feature or to meet a particular component requirement. The process selection module selects the processes for each machined shape. For example, a plane surface may need two operations, rough cut and finish cut, depending on the surface finish. Each operation has different cutting conditions or may use different tools. Another example is a large hole which may need one drilling operation and one boring operation. Thus, the process selection module of the CSB-GCAPPES decides the operations needed.

The process selection module contains the machined shape production capability knowledge and the component geometry knowledge, which comes from the component

code level two values. The shape producing capabilities of a process is the geometrical shape a specific machining operation can produce. The level two values of the code represent the machined shapes of a component.

In this prototype system, only the machined shapes shown in Table 5.6 are considered, these are: plane surface, step, slot, pocket, contour and straight hole. The shape producing capability knowledge base is based on the information in Table 5.6.

Table 5.6 Shape Process Capability

Machined Shapes	Processes
plane surface	face milling using square shoulder facemill (if surface finish Ra is 3.2μm or less then has rough and finish cut)
step	end milling using facemill
slot	end milling using endmill
pocket	end milling using round insert endmill or endmill
contour	end milling using long edge cutter or ballnose endmill
straight hole	drilling

In the process selection module, the level two values and the surface finish values of level three are used to reason the most appropriate processes and operations. For the cubic component example, one of the reasoning rules is:

```
(defrule process-selection
        (digit-2-1 1 1)
        (digit-2-1-1-surface-finish 8 1)
=>
        (printout t "The process is plane surface rough cut and plane surface finish cut." t)
        (assert(process 1 plane-surface-rough-cut)
                (process 1 plane-surface-finish-cut)))
```

Table 5.7 and Table 5.8 give the detailed *facts* in the CSB-GCAPPES for the cubic component and the flat component for the process selection.

Table 5.7 The Facts of the Cubic Component Process Selection

The Process *Facts* for the External Machined Shapes
(process 1 plane-surface-face-milling), (process 2 plane-surface-face-milling), (process 3 plane-surface-face-milling), (process 4 plane-surface-face-milling), (process 5 plane-surface-rough-cut), (process 5 plane-surface-finish-cut), (process 6 step-end-milling), (process 7, step-end-milling).
The Process *Facts* for the Holes
(process 1 straight-hole-drilling), (process 2 straight-hole-drilling), (process 3 straight-hole-drilling), (process 3 groove-end-milling).

Table 5.8 The *Facts* of the Flat Component Process Selection

The Process Facts for the External Machined Shapes
(process 1 plane-surface-face-milling), (process 2 plane-surface-face-milling), (process 3 contour-flank-end-milling).
The Process *Facts* for the Holes
(process 1 straight-hole-drilling), (process 2 straight-hole-drilling).
The Process *Facts* for the Internal Machined Shapes
(process 1 contour-flank-end-milling).

5.4 The tool selection module

The process selection module selects a set of feasible processes and operations for each machined shape. The most appropriate cutting tools are then selected for each process. Cutting tools have the capabilities to produce different machined shapes and their choice is dependent on the dimension, the tolerance and the surface finish of the machined shapes. The capability of milling tools is shown in Table 5.9 and the cutting tools capabilities knowledge base in the CSB-GCAPPES is built from the information in this table. The knowledge comes from work-shop practice and SANDVIK (SANDVIK, 1994).

Table 5.9 Cutting Tools Capability Knowledge

Machined Shapes	Cutting Tool Type
Plane surface	Square shoulder facemills
Step	Facemill
Slot	Endmill
Pocket	Round insert endmill or endmill
Contour	Ballnose endmill or long edge cutters
Straight hole	Drills of endmill

In the tool selection module, the tool selection rules return a set of feasible tools which depends on the values from level two and level three. The tool knowledge base uses *facts* and relational *schemas*. Each knowledge record stores information on the tool number, tool type, dimension and material. Table 5.10 gives the cutting tools used in the CSB-GCAPPES prototype.

Table 5.10 The Cutting Tools In the CSB-GCAPPES

Tool No.	Tool Type	Dimension (mm)	Teeth Number	Material
T01	Face mill	Diameter ϕ100	6	Carbide
T02, T03, T04	Long edge cutter	Diameter ϕ5, ϕ15, ϕ25		HSS
T05, T06, T07	End mills	Diameter ϕ5, ϕ15, ϕ25	3	HSS
T08	Ballnose endmill	Diameter ϕ6		HSS
T09, T10	Drill	Diameter ϕ13, ϕ38		HSS

The tool material selection is based on the blank material. Suggested tool material is taken from reference handbooks and in the CSB-GCAPPES only high speed steel (HSS) and

carbide tipped tools are used. The tool geometry selected is based on the geometry of the machined shape which comes from the level three digit values of the code. The tool type is determined by the process which is selected by the process selection module and stored in the knowledge base. Table 5.11 and Table 5.12 show the cutting tools used for the cubic component example and the flat component example, respectively.

Table 5.11 The Cutting Tools Used for the Cubic Component

The Cutting Tools for the External Machined Shapes
(process 1 plane-surface-face-milling T01 face-mill-ϕ100), (process 2 plane-surface-face-milling T01 face-mill-ϕ100), (process 3 plane-surface-face-milling T01 face-mill-ϕ100), (process 4 plane-surface-face-milling T01 face-mill-ϕ100), (process 5 plane-surface-rough-cut T01 face-mill-ϕ100), (process 5 plane-surface-finish-cut T01 face-mill-ϕ100), (process 6 step-end-milling T07 end-mill-ϕ20), (process 7 step-end-milling T07 end-mill-ϕ20).
The Cutting Tools for the Holes
(process 1 straight-hole-drilling T05 end-mill-ϕ5), (process 2 straight-hole-drilling T09 drill-ϕ13), (process 3 straight-hole-drilling T10 drill-ϕ38), (process 3 groove-end-milling ball-end-mill-ϕ6).

Table 5.12 The Cutting Tools Used for the Flat Component

The Cutting Tools for the External Machined Shapes
(process 1 plane-surface-face-milling T01 face-mill-ϕ100), (process 2 plane-surface-face-milling T01 face-mill-ϕ100), (process 3 contour-flank-end-milling T04 long-edge-cutter-ϕ25).
The Cutting Tools for the Holes
(process 1 straight-hole-drilling T06 end-mill-ϕ15), (process 2 straight-hole-drilling T07 end-mill-ϕ25).
The Cutting Tools for the Internal Machined Shapes
(process 1 contour-flank-end-milling T02 long-edge-cutter-ϕ5).

5.5 Machine Selection Module

In this CSB-GCAPPES prototype only two CNC milling machines are used. Table 5.13 shows the specification of these two machines. However, this can easily be extended.

Table 5.13 Machines

TYPE	MAXIMUM COMPONENT SIZE MACHINED	SPINDLE SPEED(rpm)
OCUMA-HOWA 6M CNC MILLING MACHINE	X: 720 mm, Y: 410 mm Z: 460 mm	60 - 4200
DYNA DM2400	X: 155 mm, Y: 125 mm Z: 100 mm	420 - 5000

The machine selected depends on the values of the overall shape and maximum length codes of the component as well as the cutting tool speed. Thus, the machine used for the cubic component and the flat component is the OCUMA-HOWA 6M CNC milling machine.

5.6 Cutting Conditions Selection Module

Cutting conditions include cutting speed, feed rate and depth of cut. The knowledge of selecting the cutting conditions comes from work-shop experience. Table 5.14 shows the cutting speed for HSS cutting tool material. If carbide tipped cutters are used, the cutting speeds given in the table can be multiplied by 4. The formula to convert the cutting speed in meters/min to the spindle speed is as follows:

Spindle Speed(rpm) $\approx$ 300 *Cutting Speed(m/min)/Tool Diameter(mm)

Table 5.14 Cutting Speed for HSS Cutting Tools

Material	Cutting speed(m/min)
Mild Steel	30
Aluminium	100
Brass	60
Cast Iron	20

The feed rate is calculated by the following formula:

Feed=T*R*N

T-Chip thick(mm/tooth) (Table 5.15)

R-Spindle Speed(rpm)

N-Number of Tool Teeth

Table 5.15 Chip Thickness

	Mild Steel			Aluminium, Brass, Cast Iron		
Tool Diameter mm	<ϕ12	ϕ12-ϕ25	>ϕ25	<ϕ12	ϕ12-ϕ25	>ϕ25
Chip Thick mm/tooth	0.02	0.05	0.15	0.04	0.10	0.30

Also from work-shop experience, the depth of cut for drills and end mills should not exceed ½ the diameter of the cutter and it is recommended for finish cuts to increase cutting speeds and reduce feedrates by approx 50% and to use climb milling (i.e up milling). Thus, the cutting conditions selection rules are written and based on the component material values and the tool diameters which are both stored in the knowledge base by the tool selection module. Table 5.16 and Table 5.17 give the cutting condition *facts* for the cubic component and the flat component, respectively.

Table 5.16 The Cutting Conditions for the Cubic Component

The Cutting Conditions for the External Machined Shapes
(process 1 plane-surface-face-milling face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10), (process 2 plane-surface-face-milling face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10), (process 3 plane-surface-face-milling face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10), (process 4 plane-surface-face-milling face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10), (process 5 plane-surface-rough-cut face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10), (process 5 plane-surface-finish-cut face-mill-ϕ100 spindle-speed-720 feed-162 depth-of -cut-10 up-milling), (process 6 step-end-milling end-mill-ϕ25 spindle-speed-360 feed-54 depth-of-cut-5), (process 7, step-end-milling end-mill-ϕ25 spindle-speed-360 feed-54 depth-of-cut-5).
The Cutting Conditions for the Holes
(process 1 straight-hole-drilling end-mill-ϕ5 spindle-speed-1800 feed-36), (process 2 straight-hole-drilling drill-ϕ13 spindle-speed-690 feed-34), (process 3 straight-hole-drilling drill-ϕ38 spindle-speed-236 feed-35), (process 3 groove-end-milling ball-end-mill-ϕ6 spindle-speed-1500 feed-225).

Table 5.17 The Cutting Conditions for the Flat Component

The Cutting Conditions for the External Machined Shapes
(process 1 plane-surface-face-milling face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10), (process 2 plane-surface-face-milling face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10), (process 3 contour-flank-end-milling long-edge-cutter-ϕ25 spindle-speed-360 feed-18 depth-of-cut-5).
The Cutting Conditions for the Holes
(process 1 straight-hole-drilling end-mill-ϕ15 spindle-speed-600 feed-30), (process 2 straight-hole-drilling end-mill-ϕ25 spindle-speed-360 feed-18).
The Cutting Conditions for the Internal Machined Shapes
(process 1 contour-flank-end-milling long-edge-cutter-ϕ5 spindle-speed-1800 feed-36 depth-of-cut-5).

5.7 Sequencing of Processes Module

The sequencing of the processes module chooses the suitable operations sequence for the final plan. Generally, there is always more than one process sequence which is feasible to produce a particular characteristic of a component. The objective of the process sequencing module is to minimise the time which is needed to produce a component.

In the CSB-GCAPPES two factors are taken into consideration to minimise the time. The first factor is to minimise the number of workpiece setups. Thus, the rules in the CSB-GCAPPES search the facts of the machined shapes orientation. The machined shapes with the same orientation code value will be manufactured together.

The second factor is the availability of a cutting tool to perform the required process to minimise the number of tool changes. After considering the orientation code values, the rules in the CSB-GCAPPES will search the cutting tools facts in the knowledge base to find the processes which use the same cutting tool. Table 5.18 and Table 5.19 give the process sequence for the cubic component and the flat component, respectively.

Table 5.18 The Process Sequence for the Cubic Component

plane-surface-rough-cut 138×64 orientation-top face-mill-ϕ100
plane-surface-finish-cut 138×64 orientation-top face-mill-ϕ100
straight-hole-drilling ϕ13 orientation-top drill-ϕ13
plane-surface-face-milling 101×13 orientation-front face-mill-ϕ100
straight-hole-drilling ϕ38 orientation-front drill-ϕ38
groove-end-milling ball-end-mill-ϕ6
straight-hole-drilling ϕ5 orientation-front end-mill-ϕ5
plane-surface-face-milling 50×44 orientation-right face-mill-ϕ100
step-end-milling 17×5×50 orientation-right end-mill-ϕ20
plane-surface-face-milling 50×44 orientation-left face-mill-ϕ100
step-end-milling 17×5×50 orientation-left end-mill-ϕ20
plane-surface-face-milling 101×13 orientation-back face-mill-ϕ100

Table 5.19 The Process Sequence for the Flat Component

plane-surface-face-milling 285×100 orientation-top face-mill-ϕ100
straight-hole-drilling ϕ15 orientation-top end-mill-ϕ15
straight-hole-drilling ϕ30 orientation-top end-mill-ϕ25
contour-flank-end-milling minimum-radius-ϕ5 orientation-top long-edge-cutter-ϕ5
contour-flank-end-milling minimum-radius-ϕ40 orientation-front long-edge-cutter-ϕ25
plane-surface-face-milling 285×100 orientation-bottom face-mill-ϕ100

5.8 The Plan Documentation Module

The plan documentation generates a standard process plan and prints the result on the screen. It combines all text files generated by the selection modules and also saves the final process plan as a text file so that the user can retrieve it later. Table 5.20 and Table 5.21 show the final process plans for the cubic component and the flat component, respectively.

Table 5.20 Final Process Plan for the Cubic Component

Machine: OCUMA-HOWA 6M CNC milling machine Blank: length 160, width 100, height 60 Material: mild-steel
plane-surface-rough-cut 138×64 T01 face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10
plane-surface-finish-cut 138×64 T01 face-mill-ϕ100 spindle-speed-720 feed-162 depth-of -cut-10 up-milling
straight-hole-drilling ϕ13 T09 drill-ϕ13 spindle-speed-690 feed-34
plane-surface-face-milling 101×13 T01 face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10
straight-hole-drilling ϕ38 T10 drill-ϕ38 spindle-speed-236 feed-35
groove-end-milling T08 ball-end-mill-ϕ6 spindle-speed-1500 feed-225
straight-hole-drilling ϕ5 T05 end-mill-ϕ5 spindle-speed-1800 feed-36
plane-surface-face-milling 50×44 T01 face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10
step-end-milling 17×5×50 T07 end-mill-ϕ25 spindle-speed-360 feed-54 depth-of-cut-5
plane-surface-face-milling 50×44 T01 face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10
step-end-milling 17×5×50 T07 end-mill-ϕ25 spindle-speed-360 feed-54 depth-of-cut-5
plane-surface-face-milling 101×13 T01 face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10

Table 5.21 Final Process Plan for the Flat Component

Machine: OCUMA-HOWA 6M CNC milling machine Blank: length 360, width 180, height 60 Material: mild-steel
plane-surface-face-milling 285×100 T01 face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10
straight-hole-drilling ϕ15 T06 end-mill-ϕ15 spindle-speed-600 feed-30
straight-hole-drilling ϕ30 T07 end-mill-ϕ25 spindle-speed-360 feed-18
contour-flank-end-milling minimum-radius-ϕ5 T02 long-edge-cutter-ϕ5 spindle-speed-1800 feed-36 depth-of-cut-5
contour-flank-end-milling minimum-radius-ϕ40 T04 long-edge-cutter-ϕ25 spindle-speed-360 feed-18 depth-of-cut-5
plane-surface-face-milling 285×100 T01 face-mill-ϕ100 spindle-speed-360 feed-324 depth-of-cut-10

Chapter 6. Results

The evaluation of the results is focused on two distinct items:

- the new classification and coding scheme,
- the process planning expert system.

6.1 The New Classification And Coding Scheme

The new coding and classification scheme is capable of representing three levels of information of a prismatic component: the overall description, the machined shape definition and the attributes of the machined shape. Different types (i.e. cubic, flat and long) of prismatic component have been coded for testing. A paper (Jiang et al, 1997) about the new classification and coding scheme has been presented at the Third Asia Pacific Conference on Materials Processing, Hong Kong, November 1996 and is also published in the Journal of Materials Processing Technology.

The new classification and coding scheme has flexible digt length, i.e.the more machined shapes the component has, the more digits the code has. This structure ensures the code captures enough design and manufacturing information such as the overall shape, the raw material, the maximum size of the component and, the dimensions, the tolerance, the surface finish and the orientation of each machined shape.

In most existing classification and coding schemes, the choice of values for each digit is 0-9, i.e. the maximum number of choices for each digit is ten. The advantage of the new classification and coding scheme is that its digit value choice can have more than ten numbers. This structure enhances the ability for the new classification and coding scheme to represent more detailed information of the component for process planning.

6.2 The Process Planning Expert System

The expert process planning prototype system combines component information and process information to develop processing sequencing for prismatic components. The machining processes considered by the expert process planning system are those commonly used to

produce prismatic components but can be extended to other manufacturing components. The decision making process used by the expert process planning system is not dependent on the specific machine tools used in each of these processes, and the program can easily be expanded.

Since no two manufacturing facilities are exactly alike, it is important for the user to be able to tailor a process planning system to suit the particular facility. The expert CAPP system has been designed with the flexibility to allow such modifications to be made easily.

The primary objective of this research work is to illustrate the use of the group technology principle of component classification and coding and expert system methodology in automated process planning for machined components. The result obtained from the system have shown that it is possible to use a well designed component code to determine a process plan, and the system has therefore met the primary objective of this research.

The basic goal of this research work was achieved. A paper about the code-based process planning expert system (Jiang et al, 1998) has been accepted for publication in the Journal of Expert Systems with Applications. However, as true of most researches, there were external factors that imposed restrictions on this work.

Chapter 7. Conclusion

This research project has been undertaken to investigate the application of an automated process planning system for prismatic component manufacturing. The automated process planning system is an important part of a CIM system. However, the manufacturing industry has no current effective automated process planning system. The main problems include:

- current systems require much time and effort to prepare and enter the required data,
- current systems are complex and take longer to install in a company,
- current systems are difficult to be adapted to a company's specific product, procedures and practices.

Recently, generative process planning systems have been the main research area. In order to design a complete generative process planning system, a un-ambiguous component representation is needed, which provides the process planning system with the input data. Different modules for the various decision-making processes and a uniform user interface, a common data base and interfaces to other systems are also required.

A coding and classification scheme and an expert process planning prototype system for prismatic machined components have been developed in this research work. The new classification and coding scheme captures enough design and manufacturing data for input to the CAPP system. The automated coding module is user friendly and is integrated with the expert system for generation of the process plan. It is flexible and allows for expansion.

The expert process planning prototype system has demonstrated that the expert system methodology facilitates specification, implementation and modification of the required process planning knowledge. Modules for documenting, organising and using the available manufacturing process information were also illustrated.

Chapter 8. Future Work

8.1 Expending the Generative CAPP Expert System

Efficient, computerised procedures for detailing all manufacturing processes are necessary before complete automated process planning system can be developed. However, the CSB-GCAPPES developed in this research work has not covered all geometry which will have to be included in an effective CAPP system. The following list shows some extensions and modification which still have to be implemented in the CSB-GCAPPES:

- free form surfaces,
- threads in holes,
- relative and absolute position of machined shapes.

Continued research in the area of communication with other systems in CIM, and to provide mouse-oriented user interface and knowledge base update would be the logical extension of the research described in this thesis. Thus, the development of techniques incorporating all functions into automated process planning systems should therefore receive future attention.

8.2 Integrating the Generative CAPP Expert System with CAD systems

The effective interface between CAD database and CSB-GCAPPES systems is also a major future work. The CSB-GCAPPES has been designed for code values which are decided by the user with prompt from the expert system. This is however quite time consuming and converting the new coding scheme into a computer interactive coding system directly with a CAD system would prove to be very beneficial to potential users.

An automatic coding system can be built which can analyse the solid model of a component and generate a GT code through the identification of form features. Henderson and Musti(Henderson and Musti, 1988) have done significant research in this area. More research is, however, still needed.

References

Alting L. and Zhang H., 1989, Computer Aided Process Planning: the State of the Art Survey, International Journal of Production Research, Vol. 27, no. 4, 553 -585.

Allen, D.K., 1979, Generative Process Planning System Using DCLASS Information System, Monograph 4, Computer Aided Manufacturing Laboratory, Brigham Young University, Utah.

Allen, D.K. and Smith, Paul R., 1980, Computer-Aided Process Planning, Computer Aided Manufacturing Laboratory, Brigham Young University, Utah.

Barkocy, B.E and Zdeblick, W.J, 1984, A knowledge Based System for Machining Operations Planning, SME Technical Paper MS 84-716.

Berra, P.B. and Barash, M.M., 1968, Investigation of Automated Process Planning and Optimisation of Metal Working Processes, Report 14, Purdue Laboratory for Applied Industrial Control, West Lafayette, Ind., July.

Chang, T.C., 1982, TIPPS: A Totally Integrated Process Planning System, PhD Thesis, Virginia Polytechnic Institute and State University, Blacksbury, Va.

Chang, T.C. and Wysk, R.A., 1985, An introduction to computer aided process planning systems, Englewood Cliffs, NJ:Prentice Hall.

Chang, T.C., Anderson, D.C. and Mitchell, O.R., 1988, QTC-An Integrated Design/Manufacturing/Inspection System for Prismatic Parts, Proceedings of the ASME, Computers in Engineering Conference, San Francisco, Vol. 1, pp417-426.

Chang, T. C., 1990, Expert Process Planning for Manufacturing, Addison Wesley Publishing Company, Inc.

Choi, B.K., 1982, CAD/CAM Compatible and Tool-Oriented Process Planning for Machining Centers, PhD Thesis, Purdue University, USA.

Choi B.K., Barash M.M., Andersson D.C., 1984, Automatic recognition of machined surfaces from a 3-D solid model, Computer-Aided Design, vol.16, no. 2, pp81-86.

Choi B.K. and Barash M.M., 1985, STOPP: an approach to CADCAM integration, Computer-Aided Design, vol. 17, no. 4, May, pp162-168.

Clayton, Bruce D., 1990, ART-IM Programming Tutorial, Interence Corportation, CA, USA.

Cutkoshy, K.R., Tenenbaum, J.M. and Muller, D., 1988, Features in Process-Based Design, Proceesings of the 1988 ASME International Computers in Engineering Conference and Exhibition, ASME, July 31-August 4, pp557-562.

ElMaraghy, H.A., 1991, Intelligent Product Design and Manufacturing, Chapter 7 in Artificial Intelligence in Design, ed. D.T. Pham, Springer-Verlag, pp. 147-168.

ElMaraghy, H.A., 1993, Evolution and future perspectives of CAPP, Annals of the CIRP, vol. 42, no.2, 739-751.

Englert, P.J. and Wright, P.K., 1988, principles of part set up and workholding in automated manufacturing, Journal of manufacturing systems, Vol. 7, No. 2, pp. 147-161.

Evershiem, W. and Esch, H. , 1983, Automated generation of process plans of prismatic parts, Annals of CIRP, Vol. 32/1/83, pp. 361-364.

Gallagher, C., Southern, G. and Knight, W., 1973, Group Technology, Butterworth, London.

Gusti, F., Santochi, M., and Dini, G., 1986, COATS: An Expert System for Optimal tool Selection, Annals of CIRP, Vol. 35/1/86, pp. 337-340.

Ham, I. and Lu, S.C., 1988, Computer Aided Process Planning: the Present and the Future, Annals of the CIRP, Vol. 37, No. 2, 591-601.

Henderson, M.R. and Musti, S., 1988, Automated Group Technology Part Coding From a Three-Dimensional CAD Database, Transaction of the ASME, Vol. 110, August, 278-286.

Henderson, M.R. and Anderson, D.C., 1984, Computer Recognition and Extraction of Form Features: A CAD/CAM Link, Computers in Industry, Vol. 5, pp329-339.

Houtzeed, A., 1976, The MICLASS system, paper published by TNO, Waltham, Massachusetts.

Hyer, N.L. and Wemmerlov, U., 1985, Group Technology Oriented Coding Systems: Structures, Applications and Implementation, Production andInventory Management, v26, n2, pp55-78.

Inui, M., Suzuki, H., Kimura, F. and Sata, T., 1987, Extending Process Planning Capabilities with Dynamic Manipulation of Product Models, Proceedings, 19th CIRP International Seminar on Manufacturing Systems, Penn State University, June 1-2, pp273-280.

Iwata K., Kakino Y., Oba F., Sugimura N., 1980, Development of non-part family type computer aided production planning CIMS/PRO, in: Advanced Manufacturing Technology, Blake P. (ed.), North Holland.

Jain, Vijay K., Batra, J.L. and Garg, A.K., 1995, Computer Aided Process Planning (CAPP) for electric discharge machining (EDM), Journal of Materials Processing Technology , 48, 561-569.

Jakubowski, R., 1982, Syntactic characterization of machine parts shapes, Cybernetics and Systems: An International Journal, Vol. 13, pp. 1-24.

Jashi, Sanjay B., Hoberecht, Walter C., Lee, Jason, Wysk, Richard A. and Barrick, Dean C., 1994, Design, development and implementation of an integrated group technology and computer aided process planning system, Vol 26, No.4, IIE Transactions, pp. 2-18.

Jasperse, H.B., 1992, Process Selection and Ordering in a Design for manufacturing Approach, Proc. 24th CIRP Seminar on Manuf. Sys., Copenhagen.

Jiang, B., Baines, K. and Zockel, M., 1997, A New Coding Scheme for the Optimisation of Milling Operations for Utilisation by a Generative Expert CAPP System, Journal of Materials Processing Technology, 63, pp. 163-168.

Jiang, B., Zockel, M., Baines, K., and Lau. H., 1998, A Process Planning Expert System Based on a Flexible Digit Length Coding Scheme, accepted for publication, Expert System with Applications, Vol.14, No. 2.

Kakino, Y., Ohba, F., Moriwaki, T. and Iwata, K., 1977, A New Method Of Parts Description For Computer Aided Process Planning, In Advances In Computer-Aided Manufacturing, ed. D. McPherson, pp. 197-213, North-Holland Publishing Co., Amsterdam.

Lauwers, Bert and Kruth, Jean-Pierre, 1994, Computer-Aided Process Planning for EDM Operations, Journal of Manufacturing Systems, Vol. 13, No. 5, pp313-325.

Li, J. , Han, C and Ham, I., June 1987, CORE-CAPP A company-oriented Semi-generative Computer Automated Process Planning System, Proceedings of the 19th CIRP international seminar on manufacturing systems, Pennsylvania State University, pp. 219-225.

Matsushima, K., Okada, N., and Sata, T., 1982, The integration of CAD and CAM by application of artificial intelligence techniques., Annals of CIRP, 31, 1 , pp. 329-332.

Muthsam, H. and Mayer, C., 1990, An Expert System for Process Planning of Prismatic Workpieces, Proc. 1 st Conf. Artificial Intelligence and Expert System in Manufacturing, 211-220, March.

Nau, D.S. and Chang, T.C., 1983, Prospects for process selection using AI, Computers in Industry, Vol. 4, No. 3, pp. 253-263.

Nau, D.S. and Gray, M., 1986, SIPS: An Approach of Hierarchical Knowledge Clustering to Process Planning, Bound Volume, The ASME Winter Annual Meeting, Anaheim, Calif., December.

Nau, D.S., 1985, SIPP Reference Manual, Technical Report 1515, Computer Science Department,University of Maryland, June.

Niebel, B.W., 1965, Mechanised Process Selection for Planning New Designs, ASTME Paper 737.

Offodile, O. Felix, Mehrez, Abraham and Grznar, John, 1994, Cellular Manufacturing: A Taxonomic Review Framework, Journal of Manufacturing Systems, Vol. 13, No. 3, pp196-220.

OIR, 1981, TNO, Introduction to MIPLAN, Waltham, MA: Organization for industrial research, Inc.

OIR, 1983, MULTIPLAN, Organization for Industrial Research, Inc., Waltham, Mass.

Opitz, H., 1970, A Classification System to Describe Workpieces, Pergamon Press Ltd.,Oxford, England.

Phillis, R.H., 1978, A Computerized Process Planning System Based on Component Classification and Coding, Ph.D. thesis. Purdue University, West Lafayette, Ind.

Preiss, K. and Kaplansky, E., 1984, Automated CNC Milling by Artificial Intelligence Methods, in Proceedings of AUTOFACT 6, pp. 240-259.

Sack, C.F., 1983, CAM-I's Experimental Planning System, XPS-I, Proc. of AUTOFACT'5, Detroit, Michigan, Nov.

SANDVIK, 1994, Rotating Tools, Stibo Datagrafik Press, Denmark.

Schaffer, G. ,1980, GT via Automated process planning, American Machinist, May, pp. 119-122.

Scheck, D.E., 1966, Feasibility of Automated Process Planning, Ph.D. Thesis, Purdue University, West Lafayette, Ind.

Singh, R. and Raman, S., 1992, METEX-An Expert System For Maching Planning, Int. J. Prod. Res., Vol. 30, No. 7, 1501-1516.

Smith, G., 1990, An Expert Planning System for Complex Surface Manufacture, Proc. 1 st Conf. Artificial Intelligence and Expert System in Manufacturing, 201-210, March.

Trmal, G.J., Zhu, C.B. and Midha, P.S., 1992, An Expert System For Grinding Process Optimisation, Journal of Materials Processing Technology, 33, 507-517, Elsevier.

Tsang, J.P., The Propel Process Planner, 1987, Proceeding of the 19th CIRP International Seminar on Manufacturing Systems, Penn State University, pp71-77.

Tulkoff, J., 1981, Lookheed's GENPLAN, Proceedings of 18th numerical control society annual meeting and technical conference, Dallas, Texas, pp. 417-421.

Van Houten, F.M. and Van't Evre, A.H., 1992, PART, A Feature Based Computer Aided Process Planning System, Int. J. of CAD/CAM and Computer Graphics, Vol. 7/3, pp. 335-368.

Vissa, N., 1987, A Frame-Based Generative Process Planning System for Machining Prismatic Parts, MS Thesis, Purdue University.

Wang, H-P. and Wysk, R.A., 1985, Micro-GEPPS: A microcomputer based process planning system, Proceedings of 1985 ASME Winter Meeting, PED-Vol. 19, ASME, New York, pp. 139-150.

Wang, H-P, and Li, J., 1991, Computer-Aided Process Planning, Amsterdam: Elsevier.

Wang,H-P and Wysk, R.A., 1986, An Expert System For Machining Data Selection, Computers and Industrial Engineering, vol. 10, no.2, pp. 99-107.

Wang, H-P. and Wysk, R.A., 1987, Intelligent reasoning for process planning, Computers in Industry, 8, 4, , pp. 293-309.

Waterman, D.A., 1986, A Guide to Expert System, Addison-Wesley.

Weindhal, H.P. and Schmidt, B., 1993, ESPRIT Project 6805: the COMPLAN System.

Wysk, R.A., 1977, An Automated Process Planning and Selection Program: APPAS. Ph.D. thesis, Purdue University, West Lafayette, Ind.

Wysk, R.A., Chang, T.C. and Ham, I., 1985, Automated Process Planning Systems-An Overview of Ten Years of Activities, Proc of 1th CIRP Workshop on CAPP.

Yip-Hoi, Derek and Dutta, Debasish, 1995, Data Extraction from Geometric Models for Process Planning for Parallel Machines, Journal of Manufacturing Systems, Vol. 14, No. 5, pp307-318.

Appendix : Part Program of the CSB-GCAPPES

```
(defglobal ?*digit1.1* = "
0   do not know
1   cubic components, A/B<=3, A/C<=4 (A = length, B = width, C = height)
2   flat components, A/B<=3, A/C>4
3   long components, A/B>3
" )

(defglobal ?*digit1.2* = "
0  do not know
1  A<=16 (A = length)
2 16<A<=50
3 50<A<=100
4 100<A<=160
5 160<A<=240
6 240<A<=360
7 360<A<=600
8 600<A<=1000
9 1000<A<=2000
10 2000<A " )

(defglobal ?*digit1.3* = "
0 do not know
1 mild steel
2 hardened steel
3 stainless steel
4 tool steel
5 cast iron
6 heat resistant alloys
7 titanium
8 aluminium
9 brass")

 (defglobal ?*digit2.1* = "
0 no external feature
1 plane surface
2 step
3 slot
4 pocket
5 contour
6 sculptured surface ")

(defglobal ?*digit2.2* = "
0 no holes
1 straight hole
2 one side stepped hole
3 two sides stepped hole
4 screw hole
5 tapered hole
6 hole with circular groove
7 hole with axial groove (keyway) ")

(defglobal ?*digit2.3* = "
0 no internal feature(other than hole)
1 plane surface
2 contour flank
3 sculptured surface ")



(defglobal ?*digit3.1* =  "
0  0
1  1
2  2
3  3
4  4
5  5
6  6
7  7
8  8
9  9
 10 10
 11 11
```

```
...
N N ")

(defglobal ?*digit3.2* = "
0 no tolerance requirement
1 Grade 01
2 Grade 0
3 Grade 1
4 Grade 2
5 Grade 3
6 Grade 4
7 Grade 5
8 Grade 6
9 Grade 7
10 Grade 8
11 Grade 9
12 Grade 10
13 Grade 11
14 Grade 12
15 Grade 13
16 Grade 14
17 Grade 15
18 Grade 16")


(defglobal ?*digit3.3* = "
0 no surface finish requirement
1 0.025
2 0.05
3 0.1
4 0.2
5 0.4
6 0.8
7 1.6
8 3.2
9 6.3
10 12.5
11 25
12 50 ")

(defglobal ?*digit3.4* = "
0 do not know
1 Top
2 Front
3 Right
4 Left
5 Back
6 Bottom
")


**********************************************************
(def-art-fun code-text (?input)
(close ?input)
(bind ?input1 (open "code.txt" "r"))
(bind ?input2  (read-line ?input1))
(printout t ?input2 t  )
(close ?input1))

**********************************************************************
(def-art-fun plan-text (?plan)
(close ?plan)
(bind ?plan1 (open "plan.txt" "r"))
(bind ?plan2  (read-line ?plan1))
(printout t ?plan2 t  )
(close ?plan1))

**************************************************************
(DEFRULE digit-1.1
   (INITIAL-FACT)
  =>
   (PRINTOUT T "Welcome to the Generative CAPP Expert System(GCAPPES) for prismatic
components milling operation  machining." T
"Please select the value of  following options which describes the overall shape
```

```
of your component (A=length B=width C=height)"
T ?*digit1.1* T "Please input the value of your component(0-3), then press Enter:"
T)
   (BIND ?digit-1.1    (READ))
   (ASSERT  (digit-1.1 ?digit-1.1))
   (bind ?input  (open "code.txt" "a"))
 (printout ?input
"Your component code is:" "[" ?digit-1.1 ",")
   (code-text ?input)
   )
*****************************************************************

(DEFRULE digit-1.2
   (digit-1.1 ?digit-1.1)
  =>
   (PRINTOUT T "Please select the value from the following
options which describe the maximum length of  your component" T
?*digit1.2*  T  "Please input the value of your component code:
" T)
   (BIND ?digit-1.2 (READ))
   (assert (digit-1.2 ?digit-1.2))
   (bind ?input  (open "code.txt" "a"))
   (printout ?input   ?digit-1.2 ",")
   (code-text ?input)
   )


*****************************************************************

 (DEFRULE digit-1.3
   (DIGIT-1.2 ?DIGIT-1.2)
  =>
   (PRINTOUT T "Please input the  value of the following
options which describe the material of your component: " T
?*digit1.3* T "Please input the value of your component code :
" T)
    (BIND ?DIGIT-1.3  (READ))
    (ASSERT  (DIGIT-1.3 ?DIGIT-1.3))
   (bind ?input  (open "code.txt" "a"))
   (printout ?input   ?digit-1.3 "] [")
   (code-text ?input)
   )
 ***************************************************************

 (DEFRULE digit-2.1
   (DIGIT-1.3 ?DIGIT-1.3)
=>
   (PRINTOUT T "Please input the digit 2.1 of the code
 which describe the external feature of your component:" T
?*digit2.1* T "Please input the digit of  your part code:"
T)
    (BIND ?DIGIT-2.1  (READ))
       (ASSERT ( DIGIT-2.1 ?DIGIT-2.1))
 (bind ?input  (open "code.txt" "a"))
   (printout ?input   "(" ?digit-2.1 )
   (code-text ?input))

 ************************************
 (DEFRULE digit-2.1-0
 (digit-2.1 ?digit-2.1)
 =>
 (if (= ?digit-2.1 0)
    then  (bind ?input  (open "code.txt" "a"))
    (printout ?input   ")" )
    (code-text ?input)
    ))
 ******************************************
    (DEFRULE digit-2.1-1
 (digit-2.1 ?digit-2.1)
 =>
 bind (?var 10)
 (while (>?var 0)
 do(if ( = ?digit-2.1 1)
    then (printout T
 "Please input the value of the code which describe the width of
```

```
your plane surface feature(mm):" T ?*digit3.1* T
"Please input the value of the code which describe the width of
your plane surface feature(mm):" T )
        (BIND ?DIGIT-2.1-1-width  (READ))
        (ASSERT (?var DIGIT-2.1-1-width ?DIGIT-2.1-1-width))
         (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-1-width )
    (code-text ?input)

       (printout T
"Please input the value of the code which describe the length of
your plane surface feature(mm): " T
?*digit3.1*  T
"Please input the value of the code which describe the length of
your plane surface feature(mm):" T )
    (BIND ?DIGIT-2.1-1-length  (READ))
        (ASSERT (?var DIGIT-2.1-1-length ?DIGIT-2.1-1-length))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-1-length )
    (code-text ?input)



    (printout T
"Please input the value of the code which describe the dimensions
tolerance of your plane surface feature: " T ?*digit3.2* T
"Please input the number of the code which describe the
dimensions tolerance of your plane surface feature:" T )
    (BIND ?DIGIT-2.1-1-dimension-tolerance  (READ))
    (ASSERT ( ?var DIGIT-2.1-1-dimension-tolerance
?DIGIT-2.1-1-dimension-tolerance ))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-1-dimension-tolerance )
    (code-text ?input)

    (printout T
"Please input the number of the code which describe the position
tolerance of your plane surface feature: " T ?*digit3.2* T
"Please input the number of the code which describe the position
tolerance of your plane surface feature:" T )
    (BIND ?DIGIT-2.1-1-position-tolerance  (READ))
        (ASSERT ( ?var DIGIT-2.1-1-position-tolerance
?DIGIT-2.1-1-position-tolerance ))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-1-position-tolerance )
    (code-text ?input)


    (printout T
"Please input the number of the code which describe the geometry
tolerance of your plane surface feature:" T ?*digit3.2* T
"Please input the number of the value which describe the geometry
tolerance of your plane surface feature:" T )
    (BIND ?DIGIT-2.1-1-geometry-tolerance  (READ))
     (ASSERT ( ?var DIGIT-2.1-1-geometry-tolerance
?DIGIT-2.1-1-geometry-tolerance ))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-1-geometry-tolerance )
    (code-text ?input)


    (printout T
"Please input the number of the code which describe the surface
finish of your plane surface feature: " T ?*digit3.3* T
"Please input the number of the code which describe the surface
finish of your plane surface feature:" T )
    (BIND ?DIGIT-2.1-1-surface-finish  (READ))
     (ASSERT ( ?var DIGIT-2.1-1-surface-finish
?DIGIT-2.1-1-surface-finish ))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-1-surface-finish ")")
    (code-text ?input)
```

```
(printout T
"Please input the number of the code which describe the orientation
 of your plane surface feature: " T ?*digit3.4* T
"Please input the number of the code which describe the orientation
of your plane surface feature:" T )
    (BIND ?DIGIT-2.1-1-orientation  (READ))
     (ASSERT ( ?DIGIT-2.1-1-orientation
?DIGIT-2.1-1-orientation ))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-1-orientation ")")
    (code-text ?input)
   (assert (?var 1 ?DIGIT-2.1-1-width ?DIGIT-2.1-1-length
?DIGIT-2.1-1-dimension-tolerance ?DIGIT-2.1-1-position-tolerance
?DIGIT-2.1-1-geometry-tolerance ?DIGIT-2.1-1-surface-finish ?DIGIT-2.1-1-orientation))
```

)

```
(if ( = ?digit-2.1 2)
    then (printout T
"Please input the number of the code which describe the width of
your step feature(mm):" T ?*digit3.1* T
"Please input the number of the code which describe the width
of your steps feature(mm):" T)
        (BIND ?DIGIT-2.1-2-width  (READ))
        (ASSERT (?var DIGIT-2.1-2-width ?DIGIT-2.1-2-width))
     (bind ?input  (open "code.txt" "a"))
     (printout ?input   "," ?DIGIT-2.1-2-width)
     (code-text ?input)

     (printout T
"Please input the number of the code which describe the length
of your steps feature(mm):" T ?*digit3.1* T
"Please input the number of the code which describe the length
of your steps surface feature(mm): "T )
     (BIND ?DIGIT-2.1-2-length  (READ))
     (ASSERT (?var DIGIT-2.1-2-length ?DIGIT-2.1-2-length))
     (bind ?input  (open "code.txt" "a"))
     (printout ?input   "," ?DIGIT-2.1-2-length)
     (code-text ?input)

     (printout T
"Please input the number of the code which describe the height
of your steps feature(mm): " T ?*digit3.1* T
"Please input the number of the code which describe the height
of your steps surface feature(mm):" T)
     (BIND ?DIGIT-2.1-2-height  (READ))
     (ASSERT (?var DIGIT-2.1-2-height ?DIGIT-2.1-2-height))
     (bind ?input  (open "code.txt" "a"))
     (printout ?input   "," ?DIGIT-2.1-2-height)
     (code-text ?input)

     (printout T
"Please input the number of the code which describe the
dimensions tolerance of your step feature:" T ?*digit3.2* T
"Please input the number of the code which describe the
dimensions tolerance of your step feature:" T )
     (BIND ?DIGIT-2.1-2-dimension-tolerance  (READ))
     (ASSERT ( ?var DIGIT-2.1-2-dimension-tolerance
?DIGIT-2.1-2-dimension-tolerance ))
   (bind ?input  (open "code.txt" "a"))
     (printout ?input   "," ?DIGIT-2.1-2-dimension-tolerance)
     (code-text ?input)

     (printout T
"Please input the number of the code which describe the position
tolerance of your step feature: " T ?*digit3.2* T
"Please input the number of the code which describe the position
tolerance of your step feature:" T )
     (BIND ?DIGIT-2.1-2-position-tolerance  (READ))
     (ASSERT ( ?var DIGIT-2.1-2-position-tolerance
?DIGIT-2.1-2-position-tolerance ))
   (bind ?input  (open "code.txt" "a"))
     (printout ?input   "," ?DIGIT-2.1-2-position-tolerance)
```

```
    (code-text ?input)

    (printout T
"Please input the number of the code which describe the geometry
tolerance of your step feature:" T ?*digit3.2* T
"Please input the number of the code which describe the geometry
tolerance of your step feature: "T )
    (BIND ?DIGIT-2.1-2-geometry-tolerance  (READ))
    (ASSERT ( ?var DIGIT-2.1-2-geometry-tolerance
?DIGIT-2.1-2-geometry-tolerance ))
(bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-2-geometry-tolerance )
    (code-text ?input)

    (printout T
"Please input the number of the code which describe the surface
finish of your step feature:" T ?*digit3.3* T
"Please input the number of the code which describe the surface
finish of your step feature(mm):" T )
    (BIND ?DIGIT-2.1-2-surface-finish  (READ))
    (ASSERT ( ?var DIGIT-2.1-2-surface-finish
?DIGIT-2.1-2-surface-finish ))
 (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-2-surface-finish ")")
    (code-text ?input)

     (printout T
"Please input the number of the code which describe the orientation
 of your step feature:" T ?*digit3.4* T
"Please input the number of the code which describe the orientation
of your step feature(mm):" T )
    (BIND ?DIGIT-2.1-2-orientation  (READ))
    (ASSERT (?var  DIGIT-2.1-2-orientation
?DIGIT-2.1-2-orientation ))
 (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-2-orientation ")")
    (code-text ?input)  )
```

```
(if ( = ?digit-2.1 3)
    then (printout T
"Please input the number of the code which describe the width of
your slot feature(mm): " T ?*digit3.1* T
"Please input the number of the code which describe the width of
your slot feature(mm):" T)
        (BIND ?DIGIT-2.1-3-width  (READ))
        (ASSERT (?var DIGIT-2.1-3-width ?DIGIT-2.1-3-width))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-3-width)
    (code-text ?input)


    (printout T
 "Please input the number of the code which describe the length
 of your slot feature(mm):" T ?*digit3.1* T
 "Please input the number of the code which describe the length
 of your slot surface feature(mm):" T )
    (BIND ?DIGIT-2.1-3-length  (READ))
    (ASSERT (?var DIGIT-2.1-3-length ?DIGIT-2.1-3-length))
  (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-3-length)
    (code-text ?input)

    (printout T
 "Please input the number of the code which describe the height
 of your slot feature(mm):" T ?*digit3.1* T
  "Please input the number of the code which describe the height
 of your slot feature(mm):" T )
    (BIND ?DIGIT-2.1-3-height  (READ))
    (ASSERT (?var DIGIT-2.1-3-height ?DIGIT-2.1-3-height))
 (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-3-height)
    (code-text ?input)

    (printout T
```

```
"Please input the number of the code which describe the
dimensions tolerance of your slot feature(mm):" T ?*digit3.2*
T
"Please input the number of the code which describe the
dimensions tolerance of your slot feature: "T )
    (BIND ?DIGIT-2.1-3-dimension-tolerance  (READ))
    (ASSERT ( ?var DIGIT-2.1-3-dimension-tolerance
?DIGIT-2.1-3-dimension-tolerance ))
 (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-3-dimension-tolerance)
    (code-text ?input)

    (printout T
"Please input the number of the code which describe the position
tolerance of your slot  feature: " T ?*digit3.2* T
"Please input the number of the code which describe the position
tolerance of your slot feature: "T )
    (BIND ?DIGIT-2.1-3-position-tolerance  (READ))
    (ASSERT ( ?var DIGIT-2.1-3-position-tolerance
?DIGIT-2.1-3-position-tolerance ))
 (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-3-position-tolerance)
    (code-text ?input)

    (printout T "Please input the number of the code which
describe the geometry tolerance of your slot feature: " T
?*digit3.2* T
 "Please input the number of the code which describe the
geometry tolerance of your slot feature: "T )
    (BIND ?DIGIT-2.1-3-geometry-tolerance  (READ))
    (ASSERT ( ?var DIGIT-2.1-3-geometry-tolerance
?DIGIT-2.1-3-geometry-tolerance ))
(bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-3-geometry-tolerance)
    (code-text ?input)

    (printout T "Please input the number of the code which
describe the surface finish of your slot feature: " T ?*digit3.3*
T
 "Please input the number of the code which describe the surface
finish of your slot feature: "T )
    (BIND ?DIGIT-2.1-3-surface-finish  (READ))
    (ASSERT ( ?var DIGIT-2.1-3-surface-finish
?DIGIT-2.1-3-surface-finish ))
 (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.1-3-surface-finish ")")
    (code-text ?input)  )
```

```
(printout t "Do you have more external feature?(Y or N)" t)
    (bind ?more (read))
    (assert (more ?more))
       (if (= ?more y)
    then (bind ?var (- ?var 1))
    else ( bind ?var 0)
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "]")
    (code-text ?input) )))


*************************************************************************
(DEFRULE digit-2.2
   (digit-2.1 ?digit-2.1)
=>
    (bind ?var 10)
    (while (>= ?var 1) do
     (PRINTOUT T "Please input the digit 2.2 of the code which
describe the hole feature of your component:" T
?*digit2.2* T "Please input the digit number of  your component code:"
T)
    (BIND ?DIGIT-2.2  (READ))
```

```
        (ASSERT (?var DIGIT-2.2 ?DIGIT-2.2))
(assert (?var ?digit-2.2))
    (bind ?input  (open "input.txt" "a"))
    (printout ?input   " [(" ?digit-2.2 )
    (code-text ?input)
```

```
    (if (= ?digit-2.2 0)
    then  (bind ?input  (open "input.txt" "a"))
    (printout ?input   ")" )
    (code-text ?input)
    )
```

```
    (if ( = ?digit-2.2 1)
    then (printout T "Please input the number of the code
which describe the diameter of your straight function hole
feature(mm):
" T ?*digit3.1* T "Please input the number of the code which
describe the diameter of your straight function hole
feature(mm):
"T )
        (BIND ?DIGIT-2.2-1-diameter  (READ))
        (ASSERT (?var DIGIT-2.2-1-diameter ?DIGIT-2.2-1-diameter))
         (bind ?input  (open "input.txt" "a"))
    (printout ?input   "," ?DIGIT-2.2-1-diameter )
    (code-text ?input)



    (printout T "Please input the number of the code which
 describe the length of your straight function hole
 feature(mm): " T
 ?*digit3.1*  T "Please input the number of the code which
 describe the length of your  straight function hole
 feature(mm): "T )
    (BIND ?DIGIT-2.2-1-length  (READ))
       (ASSERT (?var DIGIT-2.2-1-length ?DIGIT-2.2-1-length))
    (bind ?input  (open "input.txt" "a"))
    (printout ?input   "," ?DIGIT-2.2-1-length )
    (code-text ?input)



    (printout T "Please input the number of the code which
 describe the dimensions tolerance of your  feature: " T
 ?*digit3.2* T "Please input the number of the code which
 describe the dimensions tolerance of your  feature: "T )
    (BIND ?DIGIT-2.2-1-dimension-tolerance  (READ))
    (ASSERT ( ?var DIGIT-2.2-1-dimension-tolerance
 ?DIGIT-2.2-1-dimension-tolerance ))
    (bind ?input  (open "input.txt" "a"))
    (printout ?input   "," ?DIGIT-2.2-1-dimension-tolerance )
    (code-text ?input)

    (printout T "Please input the number of the code which
 describe the position tolerance of your  feature: " T
 ?*digit3.2* T
  "Please input the number of the code which describe the
 position tolerance of your  feature: "T )
    (BIND ?DIGIT-2.2-1-position-tolerance  (READ))
       (ASSERT ( ?var DIGIT-2.2-1-position-tolerance
 ?DIGIT-2.2-1-position-tolerance ))
    (bind ?input  (open "input.txt" "a"))
    (printout ?input   "," ?DIGIT-2.2-1-position-tolerance )
    (code-text ?input)


    (printout T "Please input the number of the code which
 describe the geometry tolerance of your  feature: " T
 ?*digit3.2* T "Please input the number of the code which
 describe the geometry tolerance of your  feature: "T )
    (BIND ?DIGIT-2.2-1-geometry-tolerance  (READ))
     (ASSERT ( ?var DIGIT-2.2-1-geometry-tolerance
 ?DIGIT-2.2-1-geometry-tolerance ))
    (bind ?input  (open "input.txt" "a"))
```

```
   (printout ?input   "," ?DIGIT-2.2-1-geometry-tolerance )
   (code-text ?input)


   (printout T "Please input the number of the code which
describe the surface finish of your  feature: " T ?*digit3.3*
T  "Please input the number of the code which describe the
surface finish of your  feature: "T )
   (BIND ?DIGIT-2.2-1-surface-finish  (READ))
    (ASSERT ( ?var DIGIT-2.2-1-surface-finish
?DIGIT-2.2-1-surface-finish ))
   (bind ?input  (open "input.txt" "a"))
   (printout ?input   "," ?DIGIT-2.2-1-surface-finish ")")
   (code-text ?input)


(printout T "Please input the number of the code which
describe the orentation of your  feature: " T ?*digit3.3*
T  "Please input the number of the code which describe the
orientation of your  feature: "T )
   (BIND ?DIGIT-2.2-1-orientation  (READ))
    (ASSERT ( ?var DIGIT-2.2-1-orientation
?DIGIT-2.2-1-orientation ))
   (bind ?input  (open "input.txt" "a"))
   (printout ?input   "," ?DIGIT-2.2-1-orientation ")")
   (code-text ?input)
 )
```

```
(printout t "Do you have more function hole feature?(Y orN)"
t)
   (bind ?more (read))
   (assert (more ?more))
      (if (= ?more y)
   then (bind ?var (- ?var 1))
   else ( bind ?var 0)
   (bind ?input  (open "input.txt" "a"))
   (printout ?input   "]")
   (code-text ?input) )))



**************************************************
(DEFRULE digit-2.3
(digit-2.2 ?digit-2.2)
=>
bind (?var 10)
(while (>?var 0)
do(PRINTOUT T "Please input the digit 2.3 of the code which
describe the internal feature of your component:" T
?*digit2.3* T "Please input the digit number of  your componcnt code:"
T)
   (BIND ?DIGIT-2.3  (READ))
      (ASSERT (DIGIT-2.3 ?DIGIT-2.3))
(assert (?var ?digit-2.3))
    (bind ?input  (open "input.txt" "a"))
   (printout ?input   " [(" ?digit-2.3 )
   (code-text ?input)
```

```
   (if (= ?digit-2.3 0)
   then  (bind ?input  (open "input.txt" "a"))
   (printout ?input   ")" )
   (code-text ?input)
   )
```

```
if ( = ?digit-2.3 1)
    then (printout T
"Please input the value of the code which describe the width of
your plane surface feature(mm):" T ?*digit3.1* T
"Please input the value of the code which describe the width of
your plane surface feature(mm):" T )
       (BIND ?DIGIT-2.3-1-width  (READ))
       (ASSERT (?var DIGIT-2.3-1-width ?DIGIT-2.3-1-width))
```

```
        (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.3-1-width )
    (code-text ?input)

      (printout T
"Please input the value of the code which describe the length of
your plane surface feature(mm): " T
?*digit3.1* T
"Please input the value of the code which describe the length of
your plane surface feature(mm):" T )
    (BIND ?DIGIT-2.3-1-length  (READ))
       (ASSERT (?var DIGIT-2.3-1-length ?DIGIT-2.3-1-length))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.3-1-length )
    (code-text ?input)

     (printout T
"Please input the value of the code which describe the dimensions
tolerance of your plane surface feature: " T ?*digit3.2* T
"Please input the number of the code which describe the
dimensions tolerance of your plane surface feature:" T )
    (BIND ?DIGIT-2.3-1-dimension-tolerance  (READ))
    (ASSERT ( ?var DIGIT-2.3-1-dimension-tolerance
?DIGIT-2.3-1-dimension-tolerance ))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.3-1-dimension-tolerance )
    (code-text ?input)

    (printout T
"Please input the number of the code which describe the position
tolerance of your plane surface feature: " T ?*digit3.2* T
"Please input the number of the code which describe the position
tolerance of your plane surface feature:" T )
    (BIND ?DIGIT-2.3-1-position-tolerance  (READ))
       (ASSERT ( ?var DIGIT-2.3-1-position-tolerance
?DIGIT-2.3-1-position-tolerance ))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.3-1-position-tolerance )
    (code-text ?input)


    (printout T
"Please input the number of the code which describe the geometry
tolerance of your plane surface feature:" T ?*digit3.2* T
"Please input the number of the value which describe the geometry
tolerance of your plane surface feature:" T )
    (BIND ?DIGIT-2.3-1-geometry-tolerance  (READ))
     (ASSERT ( ?var DIGIT-2.3-1-geometry-tolerance
?DIGIT-2.3-1-geometry-tolerance ))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.3-1-geometry-tolerance )
    (code-text ?input)


    (printout T
"Please input the number of the code which describe the surface
finish of your plane surface feature: " T ?*digit3.3* T
"Please input the number of the code which describe the surface
finish of your plane surface feature:" T )
    (BIND ?DIGIT-2.3-1-surface-finish  (READ))
     (ASSERT (?var  ?DIGIT-2.3-1-surface-finish
?DIGIT-2.3-1-surface-finish ))
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "," ?DIGIT-2.3-1-surface-finish ")")
    (code-text ?input)
```

```
(printout T
"Please input the number of the code which describe the orientation
 of your plane surface feature: " T ?*digit3.4* T
"Please input the number of the code which describe the orientation
of your plane surface feature:" T )
    (BIND ?DIGIT-2.3-1-orientation  (READ))
     (ASSERT ( ?var ?DIGIT-2.3-1-orientation
?DIGIT-2.3-1-orientation ))
    (bind ?input  (open "code.txt" "a"))
```

```
    (printout ?input   "," ?DIGIT-2.3-1-orientation ")")
    (code-text ?input)
   (assert (?var 1 ?DIGIT-2.3-1-width ?DIGIT-2.3-1-length
?DIGIT-2.3-1-dimension-tolerance ?DIGIT-2.3-1-position-tolerance
?DIGIT-2.3-1-geometry-tolerance ?DIGIT-2.3-1-surface-finish ?DIGIT-2.3-1-orientation)))
```

```
(printout t "Do you have more internal feature?(Y or N)" t)
    (bind ?more (read))
    (assert (more ?more))
        (if (= ?more y)
    then (bind ?var (- ?var 1))
    else ( bind ?var 0)
    (bind ?input  (open "code.txt" "a"))
    (printout ?input   "]")
    (code-text ?input) )))
```