ARTIFICIAL INTELLIGENCE APPROACH TO INTEGRATION OF FEATURE-BASED MODELING AND MANUFACTURING TASKS PLANNING

By

© PEIHUA GU, B. ENG., M. ENG.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

July 1989

INTEGRATION OF FEATURE-BASED MODELING AND MANUFACTURING

DOCTOR OF PHILOSOPHY (1989)
(Mechanical Engineering)

McMASTER UNIVERSITY
Hamilton, Ontario

TITLE : Artificial Intelligence Approach to Integration of Feature-Based Modeling and Manufacturing Tasks Planning

AUTHOR : Peihua Gu, B. Eng. (Tianjin University)
M. Eng. (Tianjin University)

SUPERVISOR: Professor H.A. ElMaraghy, Ph.D., P.Eng.
Director, Centre for Flexible Manufacturing
Research and Development

NUMBER OF PAGES: xvi, 259.

ABSTRACT

Two important deficiencies have been identified for the integration of CAD and automated process planning. These deficiencies stem from the lack of a uniform representation scheme of parts and products, and an effective communication for CAD and process planning. This thesis presents a new approach and original knowledge regarding the integration and individual aspects of feature-based design, cellular manufacturing planning, inspection planning and assembly sequence planning.

A high-level new language called Feature-based Design Description Language (FDDL) has been proposed and designed with a feature representation scheme. Its syntax, semantics and vocabulary have been defined with consideration given to the user, the engineering terminology, and the computer implementation. The FDDL system consists of a number of lexical analyzers, a parser and three code generators. Once the products or parts modeled by the FDDL, or by a feature-based modeler, are processed using the FDDL system, inputs are created for manufacturing tasks planning systems.

A feature-based modeling and manufacturing tasks planning system has been designed and implemented, and consists of a prototype of a feature-based modeler, the FDDL system, a feature-based cellular manufacturing planning system, a feature-based automated inspection task planner, and a prototype assembly sequence planner. The prototype feature-based modeler is used to model components using features. An expert tolerancing consultant module has been included in the modeler to

assist the user. Cellular manufacturing planning deals with group formation and parts assignment to cells. A clustering-based optimization approach has been proposed and implemented for the formation of machine cells and part families. A feature-based assignment system has been developed to integrate the feature-based design and the formed cells. Automata and pattern recognition techniques, in combination with manufacturing knowledge, are used in the system. The feature-based inspection planner has been developed to integrate the feature-based design and a Coordinate Measuring Machine (CMM). Original inspection strategies and knowledge have been developed for CMM, based on the analysis of CMM characteristics, tolerancing theories, features representation, part structure and geometry. A knowledge-based approach has been presented to integrate CAD with the assembly sequence planning. A prototype of such an assembly sequence planner has been developed for generating the assembly sequence for products from the design directly.

ACKNOWLEDGEMENT

I wish to express my sincere appreciation to my supervisor, Dr. H.A. ElMaraghy, for her support, guidance and supervision during the course of this work. Special thanks are extended to Prof. J.N. Siddall, Dr. M. Shoukri and Dr. F.S. Poehlman, members of my Ph.D supervisory committee, for their continuous interest and encouragement.

I also gratefully thank Mr. L. Hamid, research engineer at the Centre for Flexible Manufacturing Research and Development, for his assistance during the development of the FDDL language, and to my friend, Mr. T. Papazafiriou, for reading the initial draft of the thesis.

Thanks are extended to fellow colleagues and graduate students in the Department of Mechanical Engineering, with whom I have shared a pleasant experience.

The understanding and patience of my parents, my wife and son, and families will be remembered forever.

Financial support provided by the Canadian International Development Agency (CIDA), the Natural Science and Engineering Research Council of Canada (NSERC), the Government of the Province of Ontario through the Ontario Graduate Scholarship (OGS) is gratefully acknowledged.

TABLE OF CONTENTS

NOMENCLATURE

A	:	Finite State Automata
C_i	:	Machines Number of the ith Cell
D	:	Overall Average Distance of Cluster Domains
D_j	:	Average Distance of Components in Cluster Domain S_j
D_{ij}	:	Pair-Wise Distance between Cluster Centers i and j
e	:	Convergency Criteria
(e1,e2,e3)	:	Directional Parameters (Angles)
f_i	:	Utilization Factor
G	:	Grammar
I	:	Maximum Number of Iterations Allowed
J	:	Square Error in Pattern Space
K	:	Number of Cluster Domains Desired
L	:	Maximum Number of Pairs of Cluster Centres
M	:	Number of Saved Questions
N	:	Number of Components
N_c	:	Number of Cluster Domains
N_j	:	Number of Samples in S_j
P	:	A Set of Productions
P_i	:	Part Number on ith Machine
q	:	Initial State
Q	:	A Finite Set of States
Q_c	:	Lumping Parameter
Q_n	:	Minimum Number of Components in One Domain

Q_s	:	Standard Deviation Parameter
S	:	Starting Point of Productions
S_j	:	A Set of Samples of jtn Domain
T	:	Threshold Value
T_i	:	Occupying Times per Part on ith Machine
U	:	Objective Function (Total Machines)
Vt	:	Terminals
Vn	:	Non-terminals
X	:	Sample Vector (Component)
(x,y,z)	:	Coordinates
Z_j	:	Jth Cluster Centre
$ß_j$	:	Given Quantity to the Components of Z_j
$\phi(i)$	:	Constraints Function
Σ	:	A Finite Set of Input Symbols
δ	:	A Mapping
μ_j	:	Components Mean Vector of Set S_j

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Computer-aided design systems and computer-controlled machines are heavily used in industry. This high technology has improved both productivity and quality of products design and manufacturing. However, further increases in productivity and the reduction of costs for smaller batch size production require flexible automation. Much research related to flexible manufacturing is currently being conducted, and one of the fundamental issues is the integration of design and manufacturing. Currently, CAD systems and manufacturing facilities are separate. In order to improve this situation, the automated process planning, as part of whole manufacturing tasks planning, has progressed through intensive research interest during the last decade. Different approaches and systems have resulted. Recently, much effort has been focused on the application of artificial intelligence techniques to automated process planning. It has resulted in the improvement of several aspects of development in automated process planning systems, such as the system structure, the knowledge representation, and planning efficiency. However, due to the limits of a commercially available geometric modeling system, automation of the design and the process planning still remains a difficult task.

This thesis presents a novel approach to the integration of feature-based design and manufacturing tasks planning.

1.2 Literature Survey

1.2.1 Computer-Aided Geometric Modeling

The main representation schemes used for geometric modeling and drafting systems include wireframe, surface and solid modeling.

The wireframe representations are simple and can be used in interactive 2-D systems (Newman and Sproul, 1979). The internal representation are simple lines and arcs. The 3-D wireframe systems exhibit some serious deficiencies, the obvious one is the ambiguity in the interpretation of the model. A low level graphic system, such as GKS, can be used for 3-D wireframe modeling. While this system is used for engineering analysis, the calculations of moment inertia, stress, etc. are difficult without human input.

Surface modeling is an improvement on wireframe representation in that it removes the ambiguity of interpretation and is capable of modeling certain types of complex curved surface such as car bodies. It too is used for engineering analysis but some calculations are difficult to perform based on these models, such as mass property, stress-strain, and the like. Thus, it is desired that surface modeling is included in a solid modeling system so that the advantages of both representations can be maintained.

Solid modeling is the best approach and is distinguished by the use of the unambiguous representation of solids. There are two main methods being used, these are Constructive Solid Geometry (CSG) and Boundary Representation (B-rep). In the CSG scheme, some simple building primitives and boolean operators are used to create an object. The primitives include cubes, cylinders, cones, spheres and other solids bounded by quadric surfaces. This representation is in an ordered binary tree. In

the boundary representation scheme, an object is represented by its faces, edges, and vertices. The geometric data and the topological relationships among these geometric entities must be defined in the data structure so as to define a valid object. It is possible that a solid modeler allows the computation of various mass properties for the objects. Many solid modelers have been developed, such as PADL1 and PADL2 (Brown and Voelcker, 1979) and TIPS-1 (TIPS-1 System Manual). A good theoretical exploration and modeling schemes survey can be found in the references Requicha and Voelcker (1982 and 1983), Requicha (1980) and Eastman and Henrion (1979).

This current generation of geometric modelers (solid modelers, surface modelers and wireframe modelers) do not include technical information in their data base and, therefore, cannot drive manufacturing application systems such as process planning systems. Some researchers have examined the tolerance problems for geometric modeling (Requicha, 1983 and 1984), but much more work is needed to arrive at a solution. In order to utilize these modelers in the manufacturing field, different modeling systems have been developed such as (Tang, R. et al., 1987), (Tang, Z et al, 1987) and (Wang, 1984). Instead of using a solid modeler directly with manufacturing applications, many researchers define their own specific data format, e.g. features, such that the gap is bridged between the geometric modeling and manufacturing applications.

Design-with-features research is currently being conducted by several researchers such as Dixon (1988) and Cuningham and Dixon (1986). They have designed an architecture for a design-with-features system. The features are stored in a design-with-feature library, and the user can

design a component by adding, modifying or deleting operators. The features are application-oriented, which means that different features can be defined for various applications. Features are classified into two classes, static and kinetic types. The static features are primarily structural in their functional intent. The kinetic features entail motion or the transfer of energy to meet their functional intent. They also have a constructed taxonomy of design-with-feature. The design-with-features system will impose limitations on designers; the design-with-features library will be finite; and not all operations will be possible. The advantages include the abilities to capture a designer's knowledge and to encourage the standardization of design. They may provide assistance for a designer, such as design for easy manufacturing, and so forth. Several types of features follow, with examples to explain their meanings.

Casper (Luby et al., 1986) is a feature-based design aid and has been used to design aluminum castings. The design is represented symbolically in features that are defined as related to casting. Casper builds with two types of features, macro-features and co-features. Macro-features are classes, such as boxes. Co-features are attachments or details which can be added to macro-features, such as holes. The design approach is iterative. The designer can select features on the menu. Macro-features must be selected first and values are entered. Then co-features can be added. In order to allow evaluation of manufacturability, the geometric data base must express the features in terms corresponding to the available manufacturability knowledge. This system is developed using an object-oriented programming language (Stefik and Bobrow, 1986).

A feature-based modeling system has been proposed by Shah and

Rogers (1988a and 1988b) which consists of two shells, one for modeling and the other for mapping and applications. The feature modeling shell provides all necessary facilities for the creation of a product data base. It can be customized by a user organization to define the features needed by their designers. In order to use the design data base, relevant information must be extracted from the data base by the application systems. The feature mapping shell provides the facilities for extracting and reformalizing the product data base. The application knowledge can be incorporated into the shell using frames. The feature definition is fairly broad, and includes geometric features and technical features, such as tolerances features, material features, and the like. Currently, the system is still under development.

Ranyak and Fridshal provide a hierarchical approach for feature modeling (Rangak and Fridshal, 1988). They have designed a prototype of Dimension and Tolerance (D&T) in conjunction with a solid modeler. Three layers are defined in the D&T model. These are the solid geometric layer, the feature layer, and the tolerance layer. The purpose of dividing the part model is to allow separate implementation at each layer with standard interfaces for access. In the D&T model there are three types of entities: features, tolerances, and datum reference frames (DRFs). The features are defined as geometric elements and are divided into various classes such as surface features, size features, and compound features. A feature is described in a system name (ID), a user name (type), a class and the attached data. The tolerances are also divided into classes such as form, orientation, location, size, and so forth. Each of them has a specific application and interpretation. Three major systems have been integrated:

the Test Bed Solid Modeler, the D&T modeler, and the Process Planning Features system.

Turner and Anderson (1988) describe the development of an object-oriented feature-based design environment. Defined features are not only used in the design of a part, but also for process planning and visual inspection. The basic representation of a feature is that of a data object with a list of parameters. These parameters are used to describe the features. Each feature consists of two levels of information. The first level contains information common to all features. The second level of feature information includes private data to describe itself. Thus, information is different for each feature type. The tolerances are included in the descriptions. By selecting the desired stock geometry, the designer can build the object. The features are placed on the workpiece. The wireframe model is drawn on the workpiece, and an object-oriented language is used to code the system. This feature-based design is used in the process planning and in visual inspection (Chang et al., 1988).

A general description regarding features in mechanical engineering can be found in Tikerpuu and Ullman (1988) and Pratt (1988). The features for particular applications, such as process planning, are given in Unger and Ray (1988), Bound and Chang (1988) and Cutkosky and Tenenbaum (1988).

1.2.2 Automated Process Planning

Process planning has been defined by the Society of Manufacturing Engineers (Tulkoff, 1987) as " the systematic determination of the methods by which a product is to be manufactured economically and competitively." Process plans are used in industry to specify the proper sequence of

production operations, and also to specify the tools and facilities requested. The use of documentation such as operation sheets and route cards are employed. Today, most industries still use the manual method to prepare the process plans, not yet taking advantages of newly developed computer-aided methods. Generally, these methods can be divided into two basic classes, variant and generative approaches.

The variant approach is an information retrieval procedure based on Group Technology (GT). All parts are grouped into part families according to similarity of design and manufacturing. Standard process plans are created for each part family. To generate the process plan for a new part the procedure is to assign a code to the part, and this code number is used to retrieve the standard process plan. Then, the standard plan is modified for the specific part. Thus, a great deal of preliminary work is required to establish the part families and standard plans. Also, the user of the system must be experienced and competent in order to make decisions during the modification of the standard plans. Currently, most of the existing process planning systems are based on the variant approach, such as CAPP of CAM-I (Link, 1976), GENPLAN (Tulkoff, 1981) and MIPLAN (Schaffer, 1980).

Instead of retrieving standard plans, the generative approach generates a plan for the new part based on built-in logic and algorithms. The system consists of information concerning the features of the part, available machines, tools, fixtures, and so on, as well as sophisticated decision algorithms. This approach requires that the knowledge of manufacturing be captured and encoded into efficient software. In order to plan for any component, a lower level of geometric description regarding

the component must be used, such as primitives or features. These form elements and associated machines, tools, and so on are defined and included in the data base. In combination with other knowledge and algorithms for determining the optimum cutting parameters, the generative system can create the required operations and their sequences for a new component. Several such systems can be found in references AUTAP (Eversheim and Fuchs, 1980), AUTOPLAN (Vogel and Adlard, 1981), TIPPS (Chang and Wysk, 1983). However, due to the complicated decision-making process involved in this approach, the generative process planning systems are still in the experimental stages.

Recently, artificial intelligence techniques have been introduced into the process planning field, and this has resulted in a significant improvement to the current generative process planning systems structure and logic inferences.

1.2.3 Expert Systems for Process Planning

The problems of process planning make it a good candidate for the application of expert systems, since process planning requires human knowledge and is also ill defined. Thus, a number of researchers are applying expert system techniques to the problem.

TOM (Technostructure Of Machining) developed at the University of Tokyo, Japan [Matsushima et al., 1982) is a system for process planning the machining of holes. It is a rule-based process planner, written in PASCAL on a VAX computer. Inputs for TOM include an interactive input by the user and data transfer from a COMPAC system. The control structure is the backward chaining with an Alpha-Beta search strategy (Nilsson, 1980). This algorithm can significantly increases the efficiency of

a search if it can be applied to the search space. TOM uses this strategy to generate the search tree and then evaluates each node by calculating the machining time. This requires that both the sequence of hole-making and machining time are predictive. Based on the available information, the system can create a optimum machining sequence for a given hole feature. Currently, the system is only concerned about hole features and makes the assumptions of a fixed number of cuts. For the system to become practical, the number of features must be defined and a feasibility study must be conducted for the search algorithm.

GARI (Descotte and Latombe, 1984) is an AI-based process planning system. It generates the process plans based on the part description in terms of form features, such as hole, grooves, faces, etc. GARI consists of a specialized knowledge base and a planner. The knowledge base is represented by production rules. The left-hand side of a rule is a set of conditions, such as parts, available machines, and/or plan to be produced. On the right-hand side advises are in the form of weak constraints which apply to the plan once the conditions on the left-hand side are satisfied. Each advise is weighted by an integer between 1 and 10. An important advantage of GARI over earlier process planning systems is that it makes use of a specialized incremental knowledge base which is independent from the planner. Therefore, knowledge can be modified and extended to suit the needs of a particular company. This system is implemented in MACLISP language on the HP-68 computer under the MOLTICS operating system. The current system is not connected with any geometric modelers.

Van't Erve and Kals (1986a and 1986b) have reported the development of a knowledge-based generative process planning system

named XPLANE (eXpert process PLANning Environment). XPLANE includes a link to CAD systems and selection for jigs and fixtures, NC part-program generation, tools management and capacity planning. This system is not only capable of generating an adequate process plan but it is also capable of evaluating a number of alternative solutions and of selecting the optimum alternative. Also, XPLANE uses the backward chaining control algorithm, and a cost-equation is used for the evaluation. This system consists of several modules, these are a feature-recognition module, knowledge-based editor and explanatory facility. The tool-management module and the capacity-planning module exist only in concept. Van't Erve and Kals understand that a product must be completely described and must contain all relevant information needed by the other systems members. Thus, it is required that a solid modeler should include an exact mathematical geometric representation and the possibility to store additional, non-geometrical data such as material specifications, surface-finish, and so forth. The solid modeler being used in their research is based on the G.P.M. solid modeler, which is currently under development.

Hi-Mapp (Hierarchical and Intelligent Manufacturing Automated Process Planning) is a development from the University of Southern California (Berenji and Khoshnevis, 1986). This system uses form features to represent parts. The initial state of the planner consists of information about the geometric description of a part in addition to the characteristics of the available machines, tools and materials. The goal state consists of a partial ordering of the features to be processed. A revised form of a planner called Deviser by Vere (1983) was used as its core. Deviser is a general purpose planner and scheduler developed at the Jet Proposion

Laboratory. Hi-Mapp has 45 production rules and the feature number is very limited. Since Deviser is used, the control structure applies the backward chaining. The system is implemented on VAX/750 in Interlisp. The input is in the problem file which includes the part and statements of the desired goal, machines tools and the like. The feature definitions are similar to those of the GARI. Current system is not linked with a design system or a geometric modeler.

EXCAP (an EXpert Computer-Aided Process planning system) is a development from UMIST, England (Davies and Darbyshire, 1984). It is written in PASCAL, and runs on a VAX 11/750. This system is designed to plan rotational parts. The components and blank are defined in terms of an ordered sequence of dimensioned features such as face, cylinder or taper. This control structure is also the backward chaining. Fuzzy logic is used to handle uncertainty. The EXCAP forms a tree of possible operation sequences. Nodes in the tree represent various intermediate workpiece configurations; the root node represents the finished part, the terminal nodes of the tree represent the blank. Branches between nodes represent the operations used. This system can provide automatic, generative sequencing for a very limited range of turning operations. The latest version has been improved to include a comprehensive range of turning operations. The knowledge base and control structure for the improved version of the system are written in Prolog (Wright et al., 1987).

TURBO-CAPP (Wang and Wysk, 1987a and 1987b), developed at Pennsylvania State University, consists of several modules for manufacturing surface identification, knowledge base, process selection and sequence, NC program generation, knowledge acquisition and data base

management. The system is complete and is implemented on an IBM PC in PROLOG. Since AutoCAD is used as a geometric modeler, technical data, such as tolerances, surface-finish, and such must be interactively input by the user. The control strategy is the pattern-matching algorithm. This method provides the convenience for knowledge acquisition and data base modification. The system also contains some supplementary modules, such a Queries/Answers Processor, a Tolerances Input Module, and a Machine Description Module. The Q/A module is a interface between the user and the machine to acquire knowledge and allows the user to select other commands from the menu. The Tolerance Input Module is used to enter tolerances and other technical information. The Machine Description Module is also a knowledge acquisition interface which provides the facility for the user to perform specific tasks using the menus. In this way, a new file for the machine description can be created or an existing file can be reviewed and modified.

Core-CAPP (Computer-Oriented, Rule based Expert-CAPP) is a development from Pennsylvania State University (Li et al., 1987). This system is semi-generative for a particular company specializing in the manufacturing of large forged metal products. By semi-generative it is meant that some decisions can be made generatively, while others are obvious or only have limited choices and, thus, do not need to be generative. The system consists of three main modules - GT coding, part family search, and automated process planning. An interactive coding program assigns the GT code to the new part. Then, a part family number is sought out based on the GT code. Since the GT classification and coding system is mainly a brief description of the part, more detailed part

information is required. A part feature interpreter is developed, which interprets the part features according to the GT code and part family number. Also, a series of additional features, necessary for developing the process plan for the particular part, are selected. These features, which include the detailed geometric shape, shape elements, manufacturing requirements, and detailed dimensions, are interactively input. The process planning rule base stores the planning knowledge in production rules which are used to infer the operations and their sequences, the required machine tools and departments, the required cutting tools and machining conditions. Currently, the system is implemented on a IBM PC computer.

1.2.4 Integration of CAD and Process Planning

The current process planning systems are not directly integrated with the CAD systems. In order to automate the design and manufacturing, the integration of CAD and the process planning is a fundamental issue.

Lee and Fu (1987) proposed an new approach to recognize the features directly from Constructive Geometric Modeler(CSG). This method consists of two steps, namely, feature extraction and unification. Since primitives used in CSG can all have an associated local coordinate frame, Lee and Fu use the principal axis to deal with the problem of feature extraction. Primitives such as cones, cylinders, and tori can all be characterized by a single axis. Sphere can also be described using an axis with arbitrary orientation. Cubes, however, must use 12 axes because of their axis-asymmetry. This approach uses a CSG tree as an input to generate all principal axes. The axes are partitioned into several clusters based on spatial relationships. Therefore, within each cluster, axes involved in a particular feature can be located, according to the conditions defined

by the feature. Then the feature representation is unified by rebuilding the CSG tree.

Joshi and Chang (1987) have developed an interface for integration of a solid modeler and the automated process planning system. The CAD model using an internal boundary representation scheme is used to make inferences about the part and reasons, geometrically, to extract information to drive the process planning system. Some issues being considered are the determination of materials to be machined, the identification of machined faces, grouping the machined faces into features, tool approach directions for the machining process, and the development of machining precedence. A graph based features recognizer has been developed to recognize features existing in the part. These modules have been implemented. The ROMULUS solid modeler is used to model the parts and the raw material. The internal representation of ROMULUS is a Boundary Representation. The stored part description of the solid modeler is used as an input to the interface which, in turn, enhances computer understanding and analysis of the parts. It is expected that once technical information such as tolerances, surface-finish, and so forth are included in the solid modeler data base, the interface can be further improved to include this information and to enhance the reasoning process.

The product modeling approach proposed by Inui et al. (1987), Kimura et al. (1984) and Sata, et al (1985) is used to integrate design and manufacturing activities. The product model is a computer-internal model which contains a wide range of engineering information concerning the product. A solid modeling system, GEOMAP-III (Kimura, 1984), is used to

represent the geometry of a product. In order to drive the process planning system, form features are defined with all necessary attributes such as dimensions, tolerances, and the like. Based on the concept of the product model, an expert system, XMAPP (eXpert Model based Assistant for Process Planning), is developed to plan the machining process. This approach is not only concerned with the integration of CAD and the process planning, but also may be used in other engineering applications.

An approach has been proposed by Peklenik et al (1985) to integrate CAD/CAPP/CAM and GT using pattern recognition techniques. This approach is based on the GT concept and part spectrum (Peklenik and Grum, 1980). The new development of part classification for the GT is carried out using pattern recognition techniques whereby the geometric elements of the parts are developed and organized in a matrix. Once a part is coded in a binary matrix, the system can automatically determine its class using potential functions. As compensation for the lack of detailed information for the process planning and the NC programming, the form primitives are defined and some relations are provided to construct a part. This integration scheme has been tested with a rotational part.

Phillips et al (1987) have proposed an integrated design and process planning system in which a single part data base is used for all tasks. This research effort has focused on two areas, the development of a generative process planning system and an interface between CAD and the planning system. The objective of the interface is to extract the detailed part representation from a CAD data base, and convert it into a symbolic representation for input to the process planning system. The resulting system, called MICROPLAN, has been developed for rotational parts. This

system is linked with commercial CAD and CAM packages and consists of several modules. These are the conceptual design, data translation and process planning modules. A current prototype of the system has been tested on a small representative sample of rotational parts.

1.2.5 Assembly Sequence Planning

The approaches to automated assembly sequence generation can be generally divided into four classes: mating conditions to create the sequences; an assembly sequence determination by disassembly; an assembly sequence generated by a question and answer process; and a user-defined robotic assembly sequence.

Ko and Lee (1987) used the mating conditions, such as against, fits, tight-fits and contact to describe the relationships between components. An assembly can be expressed through a mating graph of components, having a hierarchical structure which is generated by developed algorithms. A bell assembly example is used to provide a assembly procedure generation process showing how the technique can be used. In fact, it is simply the ordering of components in an assembly by question and answers.

Lee and Gossard (1985) provide a hierarchical data structure for representing assemblies in a data base which is divided into two parts. The first part involves the data structure being used to store topological and geometric information on each component in an assembly. The second part deals with the data structure being used to store information on how all the components in an assembly are connected. A tree structure using the concept of a virtual link is created to represent the relationships between the components in an assembly.

Lee and Andrews (1985) developed a method to infer the position of objects in an assembly, based upon the representation of assembly by spatial relationships imposed on the component in an assembly. By using mating conditions, the system can reduce the computational time and give final solutions. Rochelean and Lee (1987) continue to discuss the so-called interactive assembly modelling in which this approach requires the user to interactively input the mating conditions of the components using the virtual link concept in combination with an inference method for position computations.

The second approach of assembly sequence planning is to generate the assembly procedure using disassembly. Sekiguchi et al (1983) developed a prototype software for the automatic verification of assembly drawings and the automatic generation of the assembly sequence. This technique is based on the assumption that the sequence of assembly is the reverse to the sequence of disassembly. Some connective relations between two parts are defined such as fit, taper contact, and so forth. These relations can be expressed by a matrix comprised of the codes arranged certain order. The groups are then generated, based on the connective relations. The disassembly sequence is determined by first, the disassembly sequence among the groups proceeding from the outside to inside, and second, selecting the part to be disassembled in a group.

Sedas and Talukdar (1987) developed a algorithm for planning disassembly based on the same assumption made by Sekiguchi et al. This work uses a combination of the geometric and the graph representations of objects. The algorithm is designed to divide an object into a pair of sub-objects (subassemblies) and verify that the subassemblies can be separated.

Two examples are provided draw and flash to demonstrate the concept. Basically, this approach is a simulation of interference checking which dictates that a part (or group) can be disassembled if no other objects are preventing it.

The third approach in automated sequence planning is the question and answer method. Bourjault (1984) presented an approach to generate all possible and valid assembly sequence for a set of parts that constitute an assembly. His algorithm is based on the rules which are derived from the answers to a series of questions concerning the mating of part pairs and multiples of parts. Bourjault begins by using the information contained in the part list and assembly drawings to characterize the assembly by a network wherein nodes represent the parts, and lines between the nodes represent any of the user-defined relations between the parts called "liaisons".

Defazio and Whitney (1987) realized that the question and answer method can be a serious problem when the number of relations are large, say more than 7. Real product assembly typically require many more than 7 parts. Thus, they proposed a simple technique based on two points: First, that a production engineer or assembly mechanic faced with an unfamiliar product to assemble asks fewer questions, not necessarily more complicated but more involved, than Bourjault. This is a set of questions which can directly evoke relationships; Second, that valid liaison sequences can follow algorithmically directly from those equivalent relationships. As with Bourjault's network, the nodes are the parts and the liaisons are the relations between the parts.

A programming system for automatically generating robotic assembly

sequence has been developed (Laperriere and ElMaraghy, 1989) in which the user interactively describes the geometry and topology of the components of an assembly and models the relationships between these components. The relation diagrams are used to specify the initial (disassembled) and final (assembled) states of the product in the assembly world. The geometric and topological data is used to validate the physical connections supplied in the relation diagrams. A robotic assembly sequence plan is generated by reasoning about the spatial constraints between components in the assembled states. In comparison with Bourjault's and Defazio's and Whitney's approach, in this system the relations are input graphically, not as a result of answering question, and are checked for validity. Also it only produces one feasible robotic assembly sequence.

Some dedicated approaches have been developed employing robots or other mechanical devices to perform the assembly such as Lieberman and Wesley (1977), Wesley et al. (1980) and Chang and Wee (1988).

1.3 Statement of Problem

A process planning system can translate a design into manufacturing instructions, however, the problem remaining is how to integrate a process planning system with a design system. Based on the analysis of both traditional and computerized design and manufacturing processes, such as machining, inspection, assembly, and previous work, two important gaps between CAD and automated process planning have been identified. These gaps are the lack of a uniform representation scheme for parts and products, and an effective communication medium for design and automated process planning systems. Most current automated process planning systems focus only on the machining process and therefore, only

the information for machining is concerned. Obviously, such information may not be enough for planning inspection and assembly. This tasks planning has not yet gained enough research attention. These processes are also important for flexible automation and computer-integrated manufacturing. Also, previous researchers have dealt primarily with the individual aspects of design and automated process planning.

1.4 Objective and Research Approach

The main objective of this project is, in general, to explore a new approach for the integration of CAD and manufacturing tasks planning, and, specifically, to develop new knowledge and techniques for various aspects of design and manufacturing tasks planning so that a new level of integration can be achieved.

The artificial intelligence approach is proposed for the integrated treatment of computer-aided design and manufacturing tasks planning. A feature-based modeling and manufacturing tasks planning system is designed. As the main theme of design and various manufacturing tasks planning, a high-level new design language called Feature-based Design Description Language (FDDL), associated with a feature representation scheme has been proposed and implemented to serve as the communication medium between the design and manufacturing. The FDDL system allows the user to directly model the product in the text description of the FDDL syntax or by using the feature-based modeler. This prototype modeler has been developed in order that the user designs the components using the features. The output of the design has a graphic representation of the design for the human user's view and a data file containing the full description of the design in FDDL syntax. After the data file of the design

is processed by FDDL system, either from the user's input or from the feature-based modeler, inputs with error-free syntax are created for the cellular manufacturing planning system, the inspection planner and assembly sequence planner. Two aspects of the cellular manufacturing planning are considered: cells formation module and knowledge-based assignment system. A new and flexible approach has been developed to form the cells and corresponding part families. This approach is based on a cluster-seeking algorithm and an optimization technique. A knowledge-based assignment system integrates the formed cells and the feature-based design and assigns a designed part to an appropriate cell. A generative inspection planner has been implemented for Coordinate Measuring Machine based on developments in original inspection knowledge and strategies. With the output from the FDDL system, the planner uses the target language to plan the inspection procedure for the designed component. An initial assembly sequence planner has been developed to generate the assembly sequence for the modeled product in the FDDL system, using developed assembly knowledge and strategies. In this way, the feature-based design is integrated with various manufacturing tasks planning.

1.5 Organization of the Work

The thesis consists of 7 chapters. The main contents of these chapters are summarized in the following paragraphs:

In Chapter 1, a general literature survey has been provided, which includes geometric modeling, process planning and an expert system for automated process planning, integration of CAD and process planning systems, and assembly sequence planning. The current situation and available methods for geometric modeling and integrating of CAD with

automated process planning are analyzed. The important gaps between the CAD systems and automated process planning systems have been identified. The main objective and research approach are discussed.

The development of the FDDL and the background of creating the language is presented in Chapter 2. This language provides the communication medium between the user, the modeling process, and manufacturing tasks planning. Thus, the design and various manufacturing tasks are analyzed to determine what information the language should provide. Based on this analysis, the language and its system are designed. The implementation of the language system is discussed in detail in second chapter.

In Chapter 3 a experimental prototype of a Feature-Based Modeler is presented as an input to the language system. This system consists of three modules, namely, the feature base, the interactive modeler, and the Expert Toleranoing Consultant (ETC). The development of this prototype system indicates that expert systems can be integrated with the geometric modeler to assist the user, and can make the design system more intelligent. Also, the geometric data for graphic presentation is separated from the feature description which is on a higher level and represented in an abstract way.

Chapter 4 deals with two issues of cellular manufacturing, the design of machine cells and part families, and the assignment of a part to machine cells. The cluster-seeking approach and optimization technique are used to form the cells and part families and the results are compared with previous works to determine the advantages of this approach. Based on the formed cells, the feature-based assignment system is developed using

finite state automata and pattern recognition techniques. This module is integrated with the feature-based modeler and the language system, and thus, can assign a new part to an appropriate cell directly from the design data base.

In Chapter 5, inspection planning is described for the Coordinate Measuring Machine (CMM). Systematic analysis of the characteristics for CMM is carried out. Based on the feature representations, CMM characteristics, the feature accessibility, and the geometric tolerancing theory, the original inspection strategies for CMM are presented in order to operate the machines efficiently. These strategies and other inspection knowledge are implemented in an expert planner, which is also integrated with the feature-based modeler and the language system.

Chapter 6 deals primarily with the assembly sequence planning problem. Based on the analysis of current available methods and assembly sequence planning, a knowledge-based approach is proposed to plan assembly sequence of a product from a CAD data base. The assembly strategies regarding the base parts and associated sub-assemblies, part function on the structure, and sequencing principles are presented and implemented in the planner. The development of this prototype system shows that the FDDL can not only be used for an individual part, but also can be used for whole products.

In Chapter 7, the conclusions and recommendations are presented.

CHAPTER 2

FDDL: A FEATURE-BASED DESIGN DESCRIPTION LANGUAGE

2.1 Introduction

The current generation of geometric modeling systems have their own data structures and geometric databases containing the representations of components in term of geometric entities such as points, lines and other primitives. They do not include essential technical information, such as tolerance, surface-finish, and material conditions, and thus, cannot drive process planning systems which depend on such information.

The motivation for creating a new language is to develop a method which will completely represent the individual parts and products in a consistent and abstract way such that the geometric and technical data are included in the CAD data base and an effective communication medium for various stages from the design to different manufacturing tasks planning (Gu et al., 1989). This information can also be retrieved and interpreted. By using the language, the user can interact with the design and manufacturing tasks planning systems, and the design and manufacturing tasks planning systems are thereby integrated.

In developing the design language, it is necessary to create a vocabulary to describe the design at the symbolic level. It is essential that users understand and use this vocabulary and, thus, the vocabulary must also be consistent with engineering representations of parts and products. At the same time, the vocabulary must be translatable to data which may

be used for inference mechanisms. The syntax for the language must be defined in such a way that the sentences easy for humans to create, yet it must also be possible to translate the elements of the language automatically. In addition, the language should be general enough to describe many products and be task independent. Finally, it must be integrated with a CAD environment.

2.2 Design of Feature-Based Design Description Language (FDDL)

2.2.1 Analysis of Design and Manufacturing

Before the language is designed and its syntax is specified, the systematic analysis of traditional design and various manufacturing tasks are required to determine the requirements of the language.

With the exception of concept generation and synthesis, a large part of the design process is routine, well-understood and consists of detailing, analysis and preparation for manufacturing. The traditional engineering drawings, used to represent parts and products, are considered to be an engineering language which currently links design and manufacturing activities. The information given by the part drawings include:

1. part geometric shape and dimensions,
2. material and heat treating requirements,
3. geometric and dimensional tolerances,
4. surface finish.

The geometric shapes represent the whole part and its individual features. These are not precisely stated, but are visually interpreted by human experts who look at the drawings and use their knowledge to understand the geometric shapes. The material is generally consistent throughout the

whole part. Tolerances and surface finish specifications are determined for individual features.

The information given by the assembly drawings contain:

1. the relations between the parts,
2. the relative positions for each individual part in the product,
3. the parts function within the product.

The relations between parts within a product indicate their connection properties, such as free contact or tight fit. The location of each part within the product structure is indicated by position information. The part function in an assembly is included in product documentation and is not specified explicitly on the drawings.

The machining process assumes that each individual part is produced from blank material and machined until finished. A machining process is planned based on the part material condition, and the geometric and technical constraints such as material hardness, shape, dimensions, tolerances and surface finishes. All this information is usually provided by engineering drawings. The machining activities include the selection of machine tools and cutting tools, and decisions regarding cutting parameters, machining sequences, and the selection of fixtures. A whole part is produced by machining individual features in a certain sequence. When a feature is associated with different tolerance and surface finish specifications as defined in Section 2.2.2, it results in a different machining process. The machining process is not reducible to something below the feature level, thus, the features which make sense for manufacturing are atomic entities.

The inspection process is discussed here in the context of

inspection conducted by Coordinate Measuring Machines (CMM). The inspection is performed by CMMs based on the geometric features which can be represented by points, lines, circles, cylinders, cones, planes, and spheres. But points, lines and circles are mathematical objects which are not manufacturable. From this point of view, the part's inspection must be decomposed into feature inspection and real world features must be represented by the geometric elements recognizable by CMMs. Relationships between features and geometric elements, representation of parts, inspection strategies, inspection knowledge development and inspection knowledge representation are discussed in detail in Chapter 5. The important information for inspection is that pertaining to geometric features and tolerance specifications.

The assembly process is only performed after all individual parts are produced. The final product is formed by assembling the parts together based on assembly drawings. Parts are associated by connecting relations which are described in terms of features. For example, fitting a shaft in a hole is a relation between two features; the solid cylinder of the shaft, and the hollow cylinder of the hole. Two contacting parts can be defined by the two specific planes which touch. The assembly sequence must consider both the relations and positions which may cause interference during the assembly process. These are the main reasons for the failure of an assembly. Therefore, a successful assembly requires information regarding the connection of features, the spatial relations and locations, and the correct sequence necessary to satisfy all the relations. The requirements for design and manufacturing are summarized as follows:

1. Information regarding the whole part must contain material,

heat-treating and part function class if necessary,

2. Information regarding individual features includes the feature geometry, tolerances, surface finish, and relations with other parts.

Generally speaking, the vocabulary of the language will express real and integer numbers, part names, part types, part classes, material conditions, feature types, feature names, dimensional and technical feature parameters, location, orientation, tolerances, surface-finishes and the connecting relations.

2.2.2 Features Definition

Features are atomic geometric entities with boundaries and technical constraints. They cannot be directly manufactured if they are further decomposed.

Current feature definitions are application oriented, and thus different sets of features are defined for different processes such as casting or inspection. In order to be used in machining, inspection, assembly and even in geometric modeling, the features must be defined in a more consistent manner.

This definition implies that a feature is basic and can be machined individually, that is, formed by the relative motions of cutter and workpiece. This is the lowest level of the definition of features from the manufacturing point of view. Different fabrications, such as casting, forging, machining, planning and forming, may require information which can only be provided by combining a few primitive features. This need not affect the language's function since code generators may be developed to handle higher-level features than those of their underlying primitives. For

a certain casting process planned, a primitive may be a box with a given length, thickness and height. It is described in the language with six instances of the features "box_face" having different dimensions, locations and orientations. A code generator can recognize one box_face and understand that more of the features may follow. Their locations, orientations and dimensions can be identified and calculated also. Another example is a cylinder which is defined by three features; two faces and one cylindrical surface (see Figure 2.1). Thus, the language is flexible.

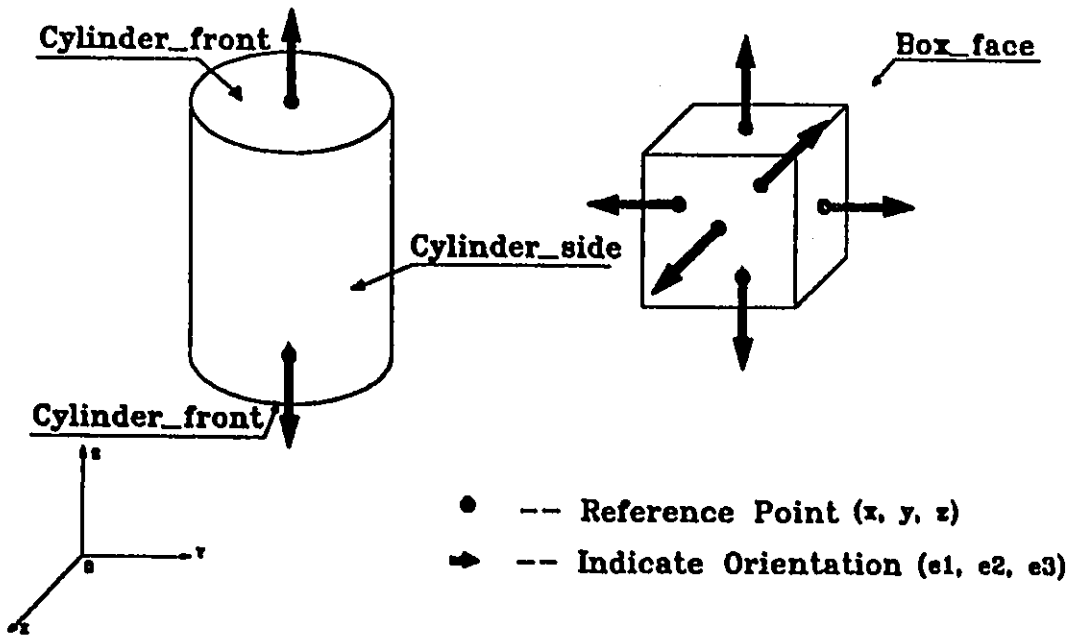


Figure 2.1 Samples of Features Definition

2.2.3 Source Language

The proposed FDDL is a source language which is used to describe the design of parts or products. It must be consistent with common engineering terminology and easily used by designers and engineers. It has to provide details and complete information for various aspects of design and manufacturing tasks. Different tasks may require different information. For example, surface-finish may not be necessary for tolerances inspection but would be essential for choosing manufacturing procedures and techniques. Therefore, it seems appropriate that a specific target language be defined for each type of tasks.

This language is application independent and is used to describe any product. The products are divided into individual interconnected components. In turn, the individual components are represented by a number of features with specific geometric and technical parameters. Thus, the language is designed having two levels, part level and feature level. The part level is concerned only with global information regarding the whole part such as material condition, classification and heat-treating. At the feature level, all local feature information, such as geometric shapes, dimensions, tolerances, surface-finish and relations as well as locations in the certain coordinate system are provided.

PART LEVEL	FEATURE LEVEL
name	name
class	type
material	parameters
heat-treatment	location
<terminator>	tolerances
	relations
	surface-finish
	<terminator>

They are further explained as follows:

part name: the user defines this with a unique identifier. Other parts in the same design must have different names.

part class: the parts can be divided into a number of classes if required. This may provide some information for following application systems.

material: based on standards, such as AISI and SAE, the material type is given, for example, AISI_1108.

heat-treating: the heat-treating is specified by the type name such as "annealing".

feature name: the user defines this with a unique identifier within the part in which it is being defined. Other parts may use the same feature name for at most one of their features.

feature type: the possible feature types are all pre-defined (see later definitions of features).

parameters: these include dimensions and technical data.

location: location is specified with a position given in three dimensional cartesian coordinates, and three orientation parameters.

tolerances: geometric and dimensional tolerances are specified using tolerance types and values and identifiers of other features if required.

surface-finish: micro inches or micro meters for the height of a rough surface for the given feature.

relations: a number of relations defined in this thesis (for example, Appendix D, relation "pfit" which means fit with

pressure).

A component may be represented using the language if it can be described by the features. However, the features defined in this body of work should be learned by the users. Other parameters, such as tolerances, surface-finish, material, heat-treating have been kept consistent with engineering definitions, thereby making the FDDL vocabulary reasonably easy to learn. A product assembly is specified by simply describing all the components and their relations with one another. The sequence of parts definition is not important. A design can be fully described using the language since the language can allow representations of both part and assembly drawings.

The source language should be easy to learn and, therefore, the specified grammar should be as natural as possible. The context-free grammar is used to specify the FDDL language. The number of design features may be very large and material types may be numerous, however, it is possible to develop a number of features which cover a reasonably wide range of components used in specific applications. For complicated parts, special features may be developed to simplify the representation process. Engineering materials are standard, and a number of types can be initially defined and a material data base may be integrated with the language. An initial set of relations have been defined and will be expanded upon as research progresses. For tolerance, surface-finish, and heat-treating, they can be defined based on handbook data. In this manner, the FDDL vocabulary can be kept at a reasonable size. The grammar specifications and lexical definitions are provided in Appendix A. Simple example of FDDL statements is given below:

```
part (name (shaft1),
                    class (2),
                    material (aisi_1330),
                    heat-treating (aging)
                    )
        (feature (name(hole1),
                    type (bore,2.4,5.),
                    location (26.0,30.0,22.0,90.0,0.0,-90.0),
                    tolerance (diameter,1.8,0.0),
                        .
                        .
                        .
                    )
                feature (name (face2),
                    type (box_face),
                        .
                        .
                        .
                    )
                )
```

2.2.4 Target Language

Currently, target languages have been developed for a machine cells assignment system, an inspection planner, and an assembly planner. These systems require information specific to their own tasks and each system generally uses only the portion of the information which is provided in a design specified using the language. The design source code is checked by a syntax checker before any translation is done. The individual task characteristics are analyzed to decide on the translation schemes to be used by the code generators.

The target languages are mainly task-oriented. Their format and style are not important as long as they are accepted by the subsequent systems. But they are still languages, and their syntax and semantics must be well formed. The FDDL language system was designed and implemented based on the above requirements.

2.3 Development of the Language System

2.3.1 Implementation Consideration of the Language

As discussed previously, the features may be defined based on the application purpose. In the mean time, the associated attributes may be increased. As rich sources of material types become available and as research develops, some new definitions may be added. Therefore, the language system must be carefully designed so that it can adapt to these situations.

1. There are a large number of special words or "keywords" in the language. Each keyword has its own specific meaning. For example, there are a multitude of words used to describe various kinds of tolerances, geometric attributes, and the relations among parts and features.

2. It is expected that the number of keywords will always increase as the need arises.

3. Usually, with each keyword there is a list of parameters (although there are other types, most are numeric). The number of parameters and the meaning of each generally depend on the associated keyword (context sensitive).

4. Due to 1, 2 and 3, the definition of the FDDL language (that is, the specifications of syntax and semantics) is best done in dictionary form, where each "sentence" or "word" is written down and defined.

The translation scheme involves the source language and the target (or object) language and specifies how elements of the source language are interpreted to be elements of the target language.

It is desireable to create a simple but very general grammar for

the parser, which ignores keyword instances. The rest of the language system must be defined and developed using a collection of keyword specific translators. The method of developing the language consists of following phases:

1. Lexical Analysis
2. Syntax Analysis
3. Code Generation

2.3.2 Lexical Analysis

The first phase of the language processing system is the lexical analysis. This process contains two stages, the first of which involves formatting the original source code. The second stage groups together the strings of characters denoting a feature, the feature identifier, part, tolerances, its relations, materials and numbers into single representing symbols. All these strings can be generated by regular expressions. For example, a real number might be generated by the regular expression:

(+|-|)digit*.digit digit*(e(+|-|)digit digit*|)

Regular expressions are equivalent to type-3 grammars and there is a one to one correspondence between the regular expressions and finite automata (Aho et al., 1985). The relevance to lexical analysis is that to each type-3 language there is a corresponding deterministic finite automaton which recognizes the strings of the language. Thus, writing a lexical analyzer consists, in part, of simulating the various automata to recognize these features, identifiers, tolerances, relations, real numbers or integers. After the transformation is done by lexical analyzers, arbitrary length identifiers, numbers and keywords will be replaced by fixed length symbols. For example, 'feature' is replaced by F. Numbers are replaced by N, Keywords

by K Identifiers by I and Part by P, etc. After the source language is processed by lexical analyzer, it is ready to be analyzed by the parser for syntactical checking.

2.3.3. Syntax Analysis

The parser processes the intermediate language code (grammar symbol or token) based on the grammar given by the language, and accepts the input language if it is correct. The grammar is specified as context-free grammar shown in Appendix A. Here, formal definitions of context-free grammar are given below (Fu, 1982):

A grammar is defined to be quadruple:

$$G = (Vt, Vn, P, S) \tag{2.1}$$

where Vt is an alphabet whose symbols are known as terminals, Vn is an alphabet whose symbols are known as nonterminals, and Vt and Vn have no symbols in common:

$$Vt \cap Vn = 0 \tag{2.2}$$

P is a set of productions and S is the starting point in the productions. The terminals are all grammar tokens and brackets in the language system. The nonterminals, productions and starting rule are defined in Appendix A. Since all character strings are replaced by tokens and their types are reasonable small, the parser becomes simple and effective. The productions for the language are specified such that the parser only performs the simple and high-level syntax checking and the rest of the detailed checking with context-sensitivity is performed by a code generator.

2.3.4 Code Generation

As mentioned before, the processing is divided into three phases.

After the lexical and syntactical analysis, the source language is considered as syntax error free. But due to the approach, it is not the general method which uses a parsing tree and a symbol table created by a lexical analyzer and which generates the language. The syntax-semantics checking is still required in the following phase of code generation. For example, following a diameter tolerance are the two real numbers which represents the lower and upper bounds of the diameter tolerances. Thus, the code generator processes the diameter tolerances' keyword and will expect two real numbers to follow.

2.4 Implementation of Feature-Based Design Description Language

2.4.1 Language System Structure

The system structure is shown in Figure 2.2. There are two ways to allow the user to enter the components descriptions, through the direct language source text file or through the Feature-Based Modeler. The Feature-Based Modeler is discussed in Chapter 3. The language is designed and implemented with the assistance of Mr. L. Hamid, software engineer at the Centre for Flexible Manufacturing Research and Development, McMaster University (Gu and ElMaraghy, 1989a). The LEX and YACC language tools on a SUN 3/260 were used in implementing the FDDL.

2.4.2 Lexical Analyzers and Parser

The lexical analyzers are composed of two main parts, the lexical scanner and the recognizer. The scanner removes blanks, tabs, carriage returns and line feeds from the input file. Then, it appends a special symbol to the end of every alphanumeric string beginning with a character. This is to allow keywords to be easily distinguishable from identifiers that happen to contain keywords as substrings. The lexical recognizers identify

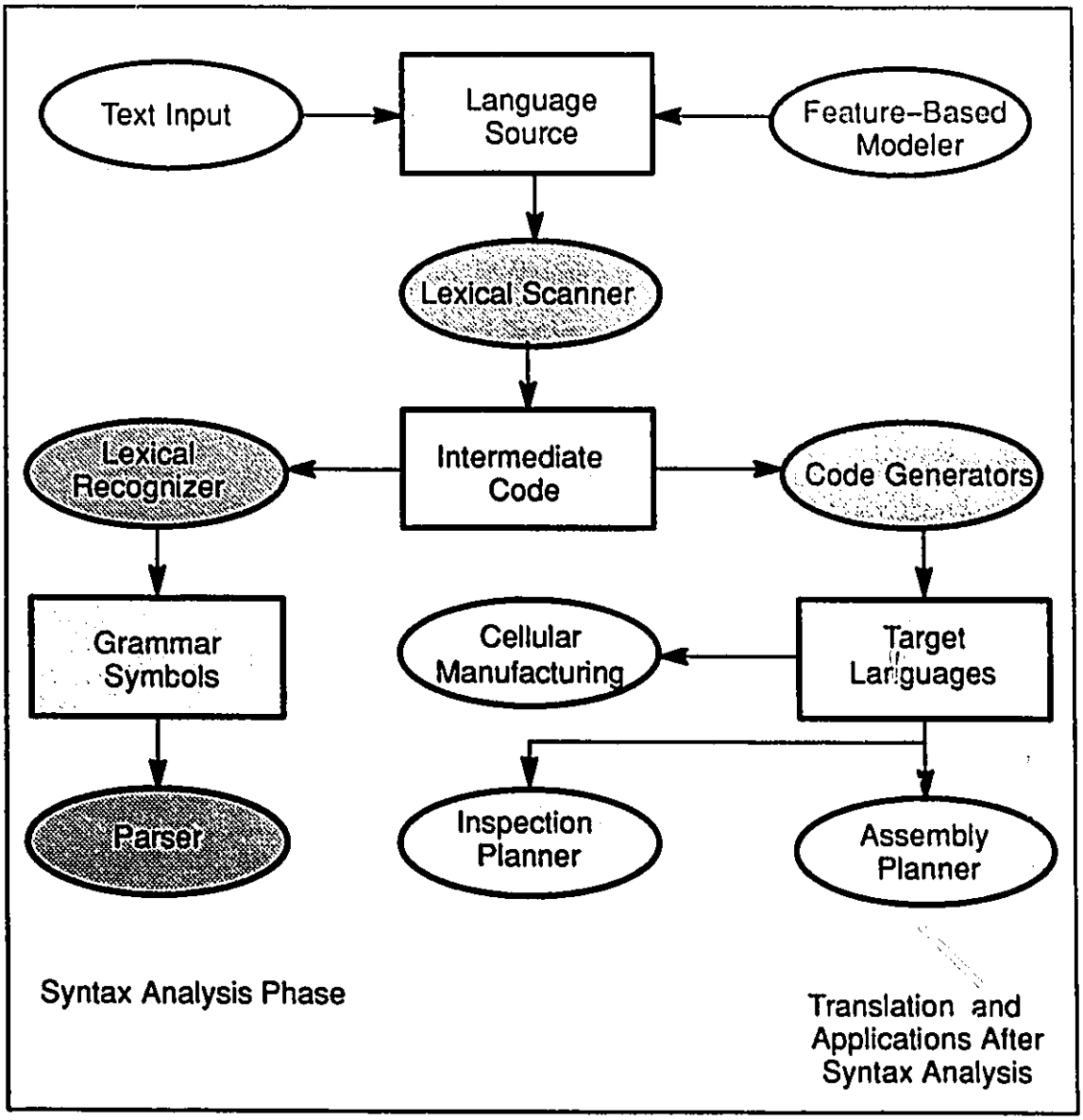


Figure 2.2 Feature-Based Modeling and Manufacturing Tasks Planning System

all the keywords numbers (both integer and real) and identifiers, and replace them by grammar tokens.

The parser accepts the input or stops when an error is detected and indicates which input token was at fault (by giving the number of the token as it appears in the input). A utility lexical analyzer can be used to indicate the location of the syntax error in the source code.

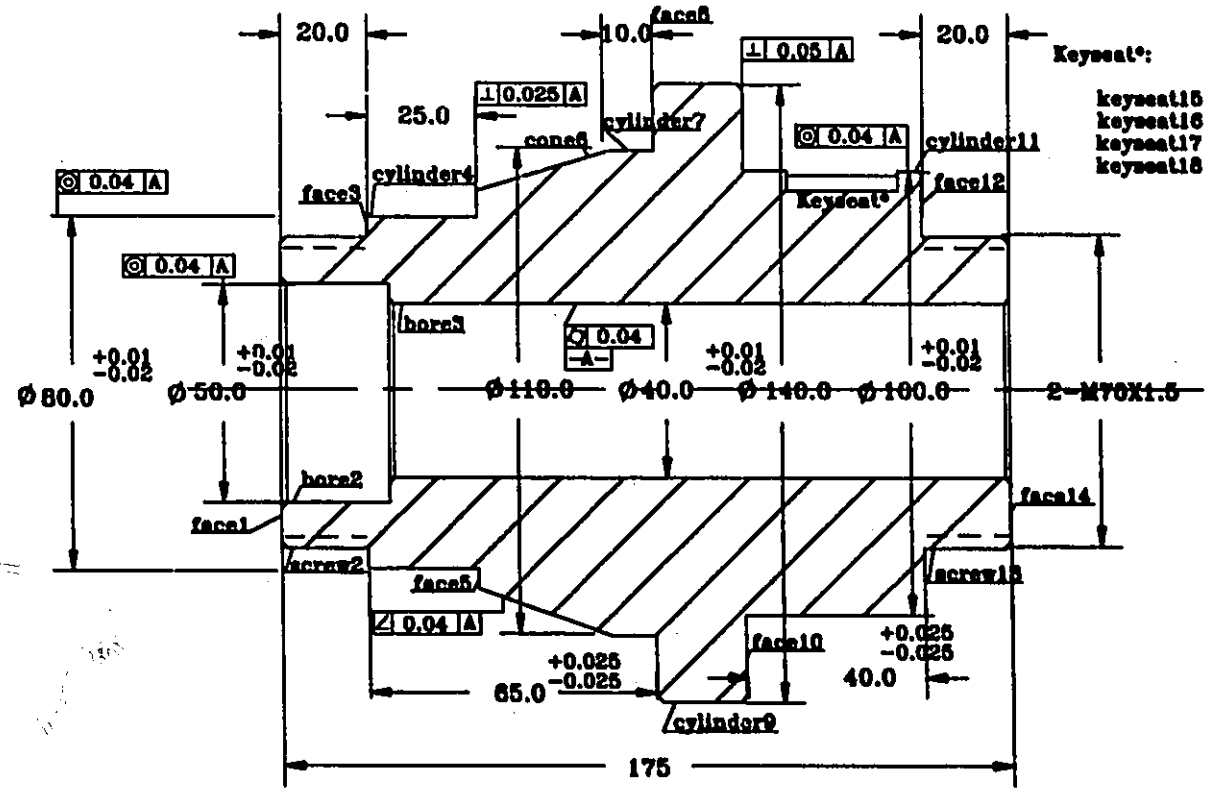
2.4.3 Code Generators

The lexical analyzers and the parser are used to check the vocabulary and general syntax. Once this phase is passed, the detailed syntax-semantics should be checked and the code is generated for the expert systems which follow. The language is a link which integrates the modeling and manufacturing tasks planning systems such as, cells planning, inspection planning and assembly planning. Each code generator assumes that lexical definitions and general syntax are correct, that detailed syntax-semantics are examined, and that codes are created in the target language syntax if no errors are detected. As discussed before, the source code contains all the design information. Since applications are different, they require different information from the source language and have differing target languages. Thus, a code generator is developed for each application.

2.4.4 Examples

A sample part, consisting of twenty features, is shown in the Figure 2.3. The FDDL source code is give in Figure 2.4. The format is already processed which means that the input format is not readable as Figure 2.4. The user simply inputs the text without concern for the format. The system will format it and process it. After lexical analysis,

all keyword, numbers and identifiers are replaced by the symbol tokens as in Figure 2.5. Then the parser checks the grammar and, if correct, syntax checking is passed. If it is not correct, the error message is printed on the screen. If the source language is syntactically correct, the code generators are used to create the codes for their applications. The code generators and associated target language syntaxes are discussed in later chapters.



Surface-finish refers to part description.

Figure 2.3 Sample Part

```
part (name (part01)
          , class (3)
          , material (aisi_1038)
          , heat_treating (annealing)
          )
      (feature (name (face1)
                , type (thread_nf_front, 70.0)
                , location (175.0, 0.0, 0.0, 0.0, 90.0, -90.0)
                , surface_finish (2.5)
                )
           feature (name (bore2)
                , type (bore, 50.0, 25.0)
                , location (175.0, 0.0, 0.0, 0.0, 90.0, -90.0)
                , tolerance (diameter, -0.02, 0.01)
                , tolerance (concentricity, 0.04, bore3)
                , surface_finish (1.5)
                , relation (pfit, part07)
                )
           feature (name (bore3)
                , type (bore, 40.0, 150.0)
                , location (150.0, 0.0, 0.0, 0.0, 90.0, -90.0)
                , tolerance (diameter, -0.02, 0.01)
                , tolerance (cylindricity, 0.04)
                , surface_finish (1.5)
                , relation (pufit, part07)
                )
           feature (name (screw2)
                , type (thread_nf_side, 70.0, 20.0, 1.5)
                , location (175.0, 0.0, 0.0, 0.0, 90.0, -90.0)
                , surface_finish (2.5)
                , relation (screw_fix, part02)
                )
           feature (name (face3)
                , type (cylinder_front, 80.0)
                , location (155.0, 0.0, 0.0, 0.0, 90.0, -90.0)
                , tolerance (distance, -0.025, 0.025, face8)
                , surface_finish (1.8)
                , relation (contact_free, part01)
                )
           feature (name (cylinder4)
                , type (cylinder_side, 80.0, 25.0)
                , location (155.0, 0.0, 0.0, 0.0, 90.0, -90.0)
                , tolerance (diameter, -0.02, 0.01)
                , tolerance (concentricity, 0.04, bore3)
                , surface_finish (1.5)
                , relation (pfit, part02)
                )
```

Figure 2.4 Formatted Source Language for the Part Shown in Figure 2.4

```
feature (name (face5)
        , type (t_cone_front, 90.0)
        , location (130.0, 0.0, 0.0, 0.0, 90.0, -90.0)
        , tolerance (perpendicularity, 0.025, bore3)
        , surface_finish (1.8)
        , relation (contact_free, part02)
        )
feature (name (cone6)
        , type (t_cone_side, 110.0, 30.0, 10.0)
        , location (130.0, 0.0, 0.0, 0.0, 90.0, -90.0)
        , tolerance (angularity, 0.04, bore3)
        , surface_finish (1.5)
        , relation (tfit, part03)
        )
feature (name (cylinder7)
        , type (cylinder_side, 110.0, 10.0)
        , location (100.0, 0.0, 0.0, 0.0, 90.0, -90.0)
        , surface_finish (3.0)
        )
feature (name (face8)
        , type (cylinder_front, 140.0)
        , location (90.0, 0.0, 0.0, 0.0, 90.0, -90.0)
        , surface_finish (2.0)
        )
feature (name (cylinder9)
        , type (cylinder_side, 140.0, 30.0)
        , location (90.0, 0.0, 0.0, 0.0, 90.0, -90.0)
        , surface_finish (3.0)
        )
feature (name (face10)
        , type (cylinder_front, 140.0)
        , location (60.0, 0.0, 0.0, 180.0, -90.0, 90.0)
        , tolerance (perpendicularity, 0.05, bore3)
        , surface_finish (1.5)
        , relation (contact_free, part04)
        )
feature (name (cylinder11)
        , type (cylinder_side, 100.0, 40.0)
        , location (60.0, 0.0, 0.0, 0.0, 90.0, -90.0)
        , tolerance (diameter, -0.02, 0.01)
        , tolerance (concentricity, 0.04, bore3)
        , surface_finish (1.5)
        , relation (pfit, part04)
        )
```

Figure 2.4 Formatted Source Language for the Part Shown in Figure 2.4 (Cont.)

```
feature (name (face12)
        , type (cylinder_front, 100.0)
        , location (20.0, 0.0, 0.0, 180.0, -90.0, 90.0)
        , tolerance (distance, -0.025, 0.025, face10)
        , surface_finish (1.5)
        , relation (contact_free, part05)
        )
feature (name (screw13)
        , type (thread_nf_side, 70.0, 20.0, 1.5)
        , location (20.0, 0.0, 0.0, 0.0, 90.0, -90.0)
        , surface_finish (2.5)
        , relation (screw_fix, part05)
        )
feature (name (face14)
        , type (thread_nf_front, 70.0)
        , location (0.0, 0.0, 0.0, 180.0, -90.0, 90.0)
        , surface_finish (2.5)
        )
feature (name (keyseat15)
        , type (p_keyseat_side, 15.0, 10.0)
        , location (40.0, 40.0, 6.0, -90.0, 90.0, 180.0)
        , surface_finish (2.0)
        , relation (kfit, part06)
        )
feature (name (keyseat16)
        , type (p_keyseat_side, 15.0, 10.0)
        , location (40.0, 40.0, -6.0, 90.0, -90.0, 0.0)
        , surface_finish (2.0)
        , relation (kfit, part06)
        )
feature (name (keyseat17)
        , type (p_keyseat_corner, 6.0, 10.0)
        , location (37.0, 50.0, 0.0, -90.0, 0.0, 90.0)
        , surface_finish (2.0)
        , relation (kfit, part06)
        )
feature (name (keyseat18)
        , type (p_keyseat_corner, 6.0, 10.0)
        , location (52.0, 50.0, 0.0, -90.0, 0.0, 90.0)
        , surface_finish (2.0)
        , relation (kfit, part06)
        )
)
```

Figure 2.4 Formatted Source Language for the Part Shown in Figure 2.4 (Cont.)

```
P (K (I)
          , K (N)
          , K (K)
          , K (K)
          )
     (F (K (I)
               , K (K, N)
               , K (N, N, N, N, N, N)
               , K (N)
               )
          F (K (I)
               , K (K, N, N)
               , K (N, N, N, N, N, N)
               , K (K, N, N)
               , K (K, N, I)
               , K (N)
               , K (K, I)
               )
          F (K (I)
               , K (K, N, N)
               , K (N, N, N, N, N, N)
               , K (K, N, N)
               , K (K, N)
               , K (N)
               , K (K, I)
               )
          F (K (I)
               , K (K, N, N, N)
               , K (N, N, N, N, N, N)
               , K (N)
               , K (K, I)
               )
          F (K (I)
               , K (K, N)
               , K (N, N, N, N, N, N)
               , K (K, N, N, I)
               , K (N)
               , K (K, I)
               )
          F (K (I)
               , K (K, N, N)
               , K (N, N, N, N, N, N)
               , K (K, N, N)
               , K (K, N, I)
               , K (N)
               , K (K, I)
               )
```

Figure 2.5 Grammar Tokens of Source Language in Figure 2.4

```
F (K (I)
      , K (K, N)
      , K (N, N, N, N, N, N)
      , K (K, N, I)
      , K (N)
      , K (K, I)
      )
F (K (I)
      , K (K, N, N, N)
      , K (N, N, N, N, N, N)
      , K (K, N, I)
      , K (N)
      , K (K, I)
      )
F (K (I)
      , K (K, N, N)
      , K (N, N, N, N, N, N)
      , K (N)
      )
F (K (I)
      , K (K, N)
      , K (N, N, N, N, N, N)
      , K (N)
      )
F (K (I)
      , K (K, N, N)
      , K (N, N, N, N, N, N)
      , K (N)
      )
F (K (I)
      , K (K, N)
      , K (N, N, N, N, N, N)
      , K (K, N, I)
      , K (N)
      , K (K, I)
      )
F (K (I)
      , K (K, N, N)
      , K (N, N, N, N, N, N)
      , K (K, N, N)
      , K (K, N, I)
      , K (N)
      , K (K, I)
      )
```

Figure 2.5 Grammar Tokens of Source Language in Figure 2.4 (Cont.)

```
F (K (I)
        , K (K, N)
        , K (N, N, N, N, N, N)
        , K (K, N, N, I)
        , K (N)
        , K (K, I)
        )
F (K (I)
        , K (K, N, N, N)
        , K (N, N, N, N, N, N)
        , K (N)
        , K (K, I)
        )
F (K (I)
        , K (K, N)
        , K (N, N, N, N, N, N)
        , K (N)
        )
F (K (I)
        , K (K, N, N)
        , K (N, N, N, N, N, N)
        , K (N)
        , K (K, I)
        )
F (K (I)
        , K (K, N, N)
        , K (N, N, N, N, N, N)
        , K (N)
        , K (K, I)
        )
F (K (I)
        , K (K, N, N)
        , K (N, N, N, N, N, N)
        , K (N)
        , K (K, I)
        )
F (K (I)
        , K (K, N, N)
        , K (N, N, N, N, N, N)
        , K (N)
        , K (K, I)
        )
)
```

Figure 2.5 Grammar Tokens of Source Language in Figure 2.4 (Cont.)

2.4.5 Integration of Design and Manufacturing Tasks Planning

The feature-based modeling and manufacturing tasks planning system shown in Figure 2.2 has been designed and implemented. The main theme is the FDDL system. The source language of parts and products modeled by either the feature-based modeler or through the FDDL can be processed by the FDDL system and the target languages can be created for the manufacturing tasks planning systems. In this way, the design and manufacturing are integrated efficiently and effectively.

2.5 Discussion

A new language FDDL has been proposed and designed in association with the feature representation. As a language, its syntax, semantics and vocabulary are defined in consideration of the human user, the engineering compatibility, and the ease with which computer implementation can be achieved. Thus, the vocabulary is compatible to existing engineering terminology. The grammar is reasonably simple to learn. The FDDL system has been developed and consists of a number of lexical analyzers, parser and code generators. Once the products or parts design is processed by the FDDL system, inputs with syntax error free are created for manufacturing tasks planning systems, and so, this new language helps integrate the developed feature-based modeler prototype and manufacturing tasks planning modules.

CHAPTER 3

FEATURE-BASED MODELER

3.1 Introduction

This chapter presents the development of a prototype for a feature-based modeler. As mentioned earlier, the FDDL system can be input using either direct text file or through a feature-based modeler. This prototype models parts in an interactive fashion with a 3-D wireframe graphic representation, and the resultant data file is the source language in FDDL syntax. Compared with a geometric modeling system, a graphic representation of the designed object is obtained with a feature-based description being produced at the same time. Thus, the two representations are separated such that the feature-based representation plays the role of driving the application systems. A general scheme for the integration of the feature-based modeler and geometric modeler is proposed in Figure 3.1. Based on this scheme, successful development in such modelers must satisfy the following requirements:

1. The feature description must be complete from a design and manufacturing point of view, and the features can be recognized or reasoned at an abstract level;
2. The features can be represented by geometric entities used by geometric modelers.

The features defined in the feature-based modeler must be graphically represented by a geometric modeler. The geometric modeler has its own

data base to store the geometric information. The data base of the feature-based modeler contains an abstract description of the parts containing all the necessary geometric and technical parameters. This prototype is used to demonstrate these concepts.

3.2 Development of a Prototype of Feature-Based Modeler

This modeler has three main components shown in Figure 3.2 and Figure 3.3:

1. Feature Base
2. Interactive Modeling
3. Tolerancing Consultant Module for Fits

3.2.1 Feature Base

The Feature Base is composed of a number of form features which are describable in FDDL feature definitions. These form features are the definition-based analysis of some mechanical components, and more can be added if required. The features are listed in the form of menu, as shown in Figure 3.4, and are linked with GKS subroutine. These features are divided into different types, and the user chooses the features from the menu to design a component. When all the data for drawing the feature is completed, a 3-D wireframe model is displayed on the screen. The GKS is a very low level 2-D graphic package, which allows the drawing of a line between two points (x1,y1) and (x2,y2). The system is required to compute the projection and all data for the 3-D wireframe model.

The development of a complete geometric modeling system is a time consuming project and it is not the purpose of this research. Hence, it is sufficient to concentrate on a demonstration prototype with limited functions such that the ideas can be presented clearly and in a reasonable

amount of time. A number of features have been developed such as cylinder, cone, box, polygons, sectors, spline shaft, screw, keyseats, slots, chamfers, plane, various holes. Each of these features is represented symbolically in the FDDL language, explicit with correct syntax and lexical definitions (Appendix B). They can also be associated with possible tolerance specifications. These features must also be represented by 3-D model. Thus it is necessary to define them with variables which are assigned values during the modeling process. This means that each feature is defined as a shell with expected data. For example, a cylinder can be defined geometrically by a radius with two points (x1,y1,z1) and (x2,y2,z2) being the end points of its axis. Seven variables have to be assigned. When the data is input by the user, the system will compute all the necessary data for drawing the cylinder. Each feature constructs a shell unit, which requires different data such as tolerances, surface-finish, relations and other information supplied by the user, all of which are feature-dependent.

3.2.2 Interactive Modeling

Interactive Modeling is an environment in which questions are asked and the input data is expected. Due to the syntax and vocabulary of the FDDL, the questions are divided into two levels, part level and feature level. On the part level, the questions are concerned with the whole part properties, such as materials and their treatment. On the feature level, the questions concentrate on the dimensions, the coordinates for the feature in a world coordinate system, the location and orientation, the geometric and dimensional tolerances, the identifier of features, the relations with other parts and surface-finish. Therefore, all of the data requested are divided into three classes, geometric data, technical data,

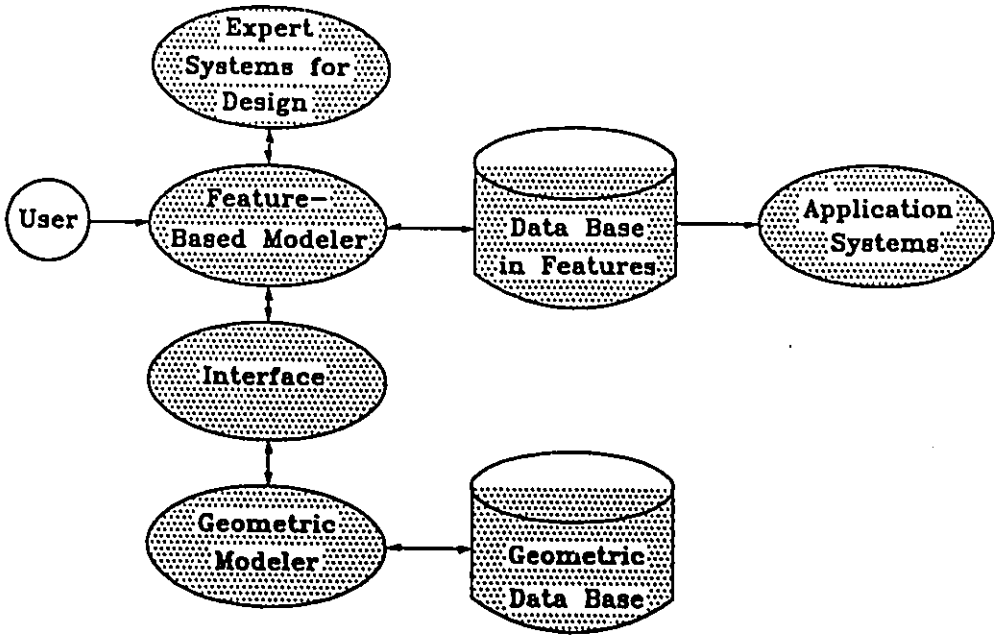


Figure 3.1 Integration Scheme by Feature Representation

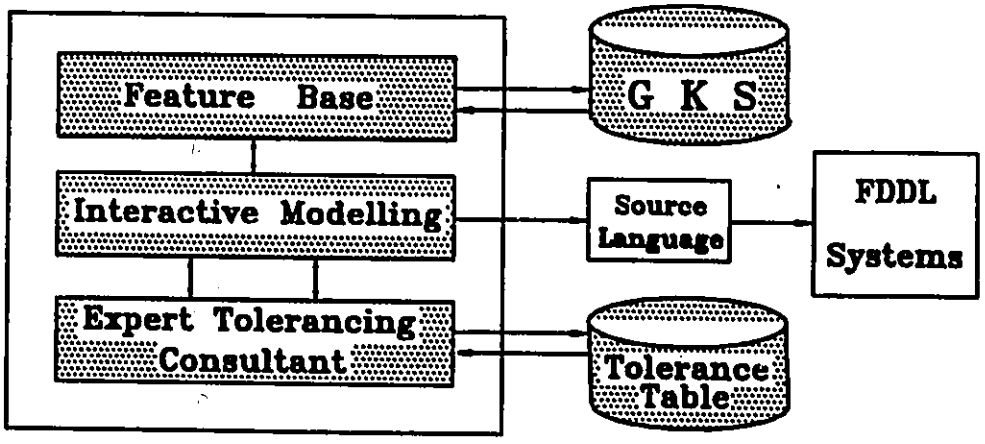


Figure 3.2 Structure of Feature-Based Modeler

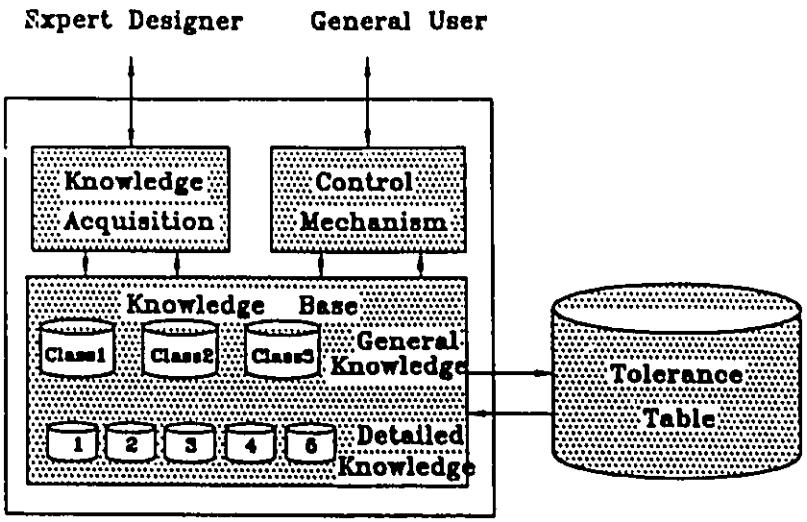


Figure 3.3 Expert Tolerancing Consultant Module

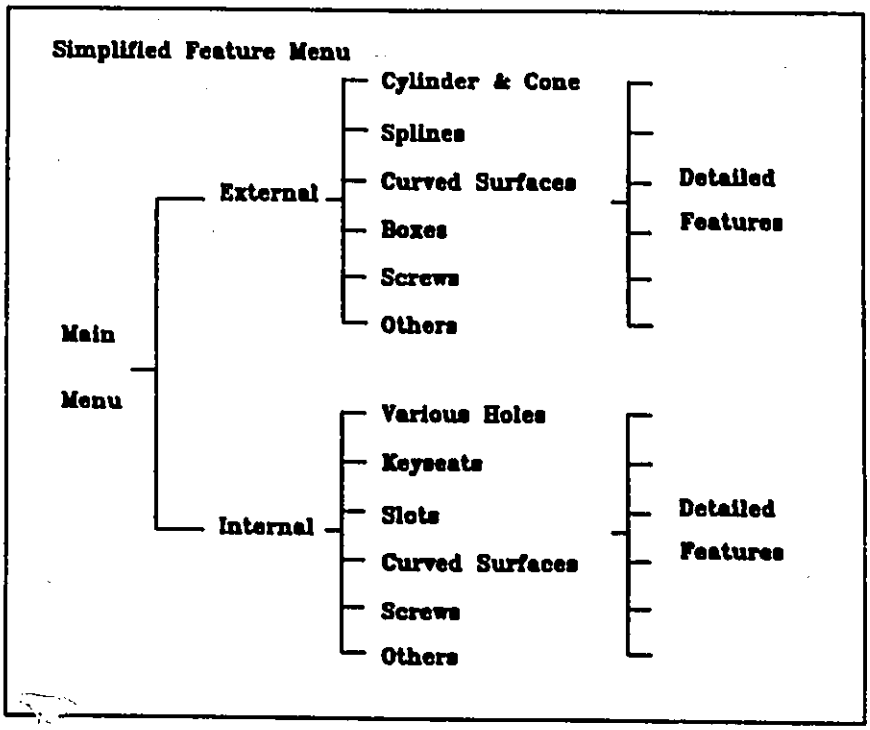


Figure 3.4 Feature Base Listed in Menu

and identifiers. These questions are compatible with the design and manufacturing terminologies. The feature name, tolerances type, datums, and so forth will all be determined interactively. For example, for the box in Figure 2.1, the process is as follows:

S(System): Menu for rotational/nonrotational part;

U(User) : Choose the nonrotational one;

S : Display a menu in which box, plate, control parts, etc. provided;

U : Choose the part class as box;

S : Provide all form features in a menu again, such as cylinder, cone, polygon, various holes, screw, box, etc;

U : Choose the box;

S : Show the user two choices, box with round or sharp corners;

U : Choose the sharp one;

S : Expect eight vertices coordinates, sequence is that keeping z coordinate unchanged, input four vertices in a sequential manner, then other four;

U : Input eight vertices coordinates;

S : Maximum dimension for whole box, which is normalized for display;

U : Give the value;

S : Ask the number of faces you want to describe;

U : Enter six in this case;

S : Display the numbers which represent the faces;

U : Choose a number to start;

S : Ask the user to give the identifier for the feature;

U : Give the identifier in character string;

S : Display the instruction for the user to input the three directional

parameters and waiting for inputs;

U : Enter three angles which are defined by the normal direction of feature and three axis of the system;

S : Dimensional tolerances?

U : Answer y for yes or n for no;

S : If yes, two values and one reference datum are expected;

U : Input following instructions on the screen;

S : Relations if any, y/n?

U : Yes if it has, then system displays all relations defined in this research, choose one for your case;

S : Asks the user to input the related part;

U : Give the part identifier;

S : Geometric tolerances y/n?

U : Yes for example;

S : Display all types of geometric tolerances in the form of a menu for user to choose;

U : Choose perpendicularity for example;

S : Asks the user to input value;

U : Give value;

S : Asks the user to input the number of datums for this case;

U : Enter one in this example;

S : Wait for user's input identifier of the datum feature;

U : Provide the feature identifier;

S : Asks the user again about whether there are other geometric tolerances because some features may have more than one type of geometric tolerances;

U : No is responded;

S : Display the instruction to wait for user's value for surface finish;

U : Input the value;

S : Wait for user's input number to indicate other face;

.

.

.

Other faces are treated exactly the same as above. After the user completely answers all questions, the system will print these descriptions in the syntax provided by the language. For simplicity, the question and answer process terminates by input of a surface-finish specification for the feature. The system displays the 3-D wireframe model after a form feature is completed.

3.2.3 Expert Tolerancing Consultant Module

Tolerances consist of both dimensional and geometric tolerances which, when combined with the modelling features, form the foundation for a part's function realization. The assignment of tolerances depends mainly on the expertise of the design engineers since very few laws and rules exist for this task. The Expert Tolerancing Consultant for fitting can provide recommendations about reasonable fits and tolerances (Figure 3.3).

The knowledge base for fit tolerances is designed as a tree structure shown in Figure 3.5. The user can be either a professional designer (expert) or general layman (novice). Since the knowledge acquisition is an important factor to maintain the system's usefulness, emphasis is placed on the learning and modification functions of the system knowledge. Therefore, the structure of the knowledge base has

been designed such that it can be easily extended by knowledge acquisition.

The relative motions of the mating parts are generally divided into two types: stationary and movable. For the cylinders, they may fit in bearings, gears, v-belt sheaves, etc. The stationary type can be apportioned into different classes according to parts function and fitting properties, such as gears and bearings. If the number of the classes is defined, it may limit the knowledge database scope and will finally result in a system which cannot be easily extended. Hence, the number of classes has not been determined a prior, and new knowledge can be added gradually. Therefore, whenever an expert decides that a new type of part should be added to the knowledge base, the system can automatically acquire the knowledge and store it in the database for future use. This provides flexibilities for the development of the system so that it can maintain its usefulness and evolve and expand as the need arises. For the movable type of parts, the exact same procedure may be used.

The general structure of the database is shown in the Figure 3.5. The index is stored in the key data file, where parts are classified into general form types, such as gears and bearings. Corresponding to each index item, a data file is defined which is used to store the descriptions of the sub-classes. For example, the bearings in the index data file correspond to a data file, in which ball bearings, cylindrical bearings, spherical bearings, thrust bearings are stored. For each item in the sub-classes data file a detailed knowledge is provided. In this data file, the rule number, the range of part diameters, the parts loading conditions, application examples and recommended fit types are fully defined (see

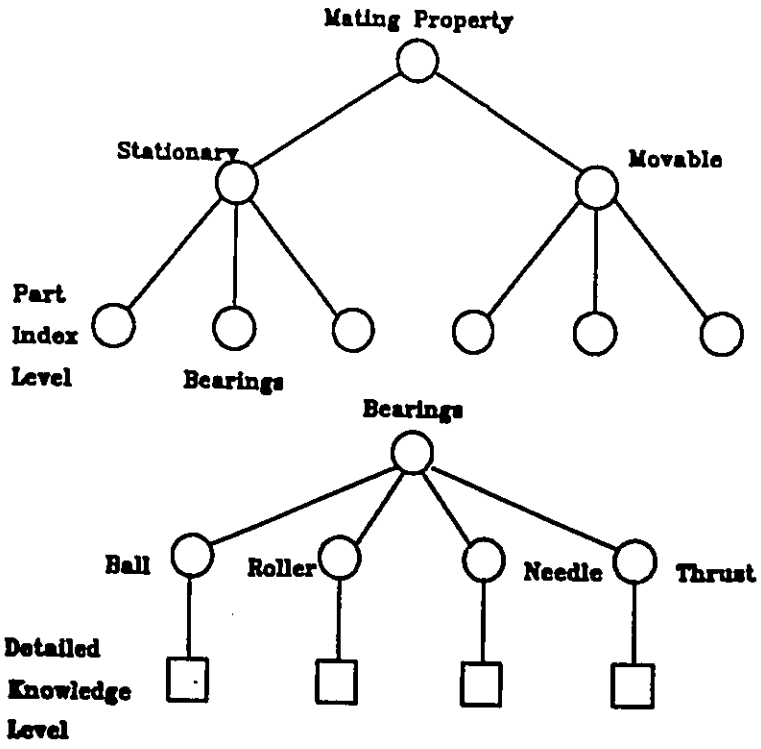


Figure 3.5 Knowledge Base Structure

14					
2	Diameter		Condition	Example	Fits
3	0.7	3.9	Light & Variable	Conveyors	j6
4	3.9	5.5	Light & Variable	Conveyors	k6
5	0.0	0.7	Normal & Heavy	Motor & Engine	j5
6	0.7	3.9	Normal & Heavy	Motor & Engine	k5(k6)
7	3.9	5.5	Normal & Heavy	Motor & Engine	m5(m6)
8	5.5	7.9	Normal & Heavy	Motor & Engine	m6
9	7.9	11.0	Normal & Heavy	Motor & Engine	n6
10	0.0	0.7	High Accuracy	Machine Tool	(h5)6
11	0.7	3.9	High Accuracy	Machine Tool	(j5)6
12	3.9	7.9	High Accuracy	Machine Tool	(k5)6
13	0.0	9.8	Axial Loads	All Kinds	j6
14	9.8	100.0	Axial Loads	All Kinds	js6

Figure 3.6 Detailed Knowledge for Ball Bearings

Figure 3.6). The extension and modification of the above three level data files can be freely made, which ensures that the system can maintain its correct functions and increase the validity of its consultations.

The tolerancing knowledge is represented by the fitted parts function, diameter range, loadings condition, and application as well as corresponding fits and tolerances. Those which are in the form of facts and rules are built in the system, and the standard fits and tolerances tables (Oberg et al., 1988) are also included in the system. The current module consists of about 86 rules including inference of the tolerance values. Moreover, whenever a user provides new knowledge, the system can put the knowledge in a certain position based on the rule number since the rules structure is defined as IF - THEN - ELSE, for example :

Rule:

IF a cylinder is mounted on a ball bearing and,
diameter is within 3.94 in. to 5.5 in. and,
loading condition is light, or variable
loads(p<=0.07) and, it is used in conveyors,
or lightly loaded gearbox

THEN recommended fit is k6 (ANSI Standard Transition
Location fit).

IF the diameter is greater than 3.94 in. and smaller
than or equal to 4.73 in.,

THEN the tolerance is between 0.001 and 0.0001 in.

IF the diameter is greater than 4.73 in. and smaller
than or equal to 5.5 in.,

THEN the tolerance is between 0.0011 and 0.0001 in.

Rule:

IF a cylinder is mounted on a spherical roller bearing and, diameter is within 5.5 in. to 11.02 in. and, loading condition is very heavy, or shock loads with difficult working conditions ($P > 0.15$ C) and, it is used in axleboxes for heavy railway vehicles, or traction motors, or rolling mills

THEN recommended fit is p6 (ANSI Standard Interference Location Fit).

IF the diameter is greater than or equal to 5.5 in. and smaller than or equal to 7.09 in.,

THEN the tolerance is between 0.0028 and 0.0018 in.

IF the diameter is greater than 7.09 in. and smaller than or equal to 9.85 in.,

THEN the tolerance is between 0.0032 and 0.002 in.

IF the diameter is greater than 9.85 in. and smaller than or equal to 11.05 in.,

THEN the tolerance is between 0.0034 and 0.0022 in.

The knowledge acquisition interface is actually designed for professional engineers who are truly expert in the field of tolerancing. The knowledge acquisition is an important feature which maintains the system's usefulness in the future. A limited knowledge acquisition facility has been implemented for tolerancing. Generally speaking, this process is done by knowledge engineers interviewing human experts. In this system the interview process is carried out in dialogue style between the computer

and the expert users. The implemented interface plays the role of knowledge engineers. Domain rules provided by human experts are put in the knowledge base automatically. When an expert provides a new class of parts which the system cannot find by searching the corresponding data files, the system arranges these into a file stack which stores the parts class index, while the corresponding pointer is increased by one. Then a sub-class is asked if any data file is opened for the sub-classes which is similar to the previous parts class index. Finally, if the sub-class is created in this interview, detailed knowledge is requested and stored in the new file. Otherwise, it can be added to an existing file. In this way, fit expertise can be transferred into the knowledge base. Accompanying this interface, the modification feature of the knowledge base was built in so as to change some expertise as technology develops. This process is also designed as a dialogue between the system and the human experts. Once the modification module is accessed, the detailed knowledge is displayed on the screen after a pointer is indicated. Then, modifications can be made. For example, if a shaft mounted with a gear is introduced but knowledge about the gears fitting on the shaft is not included in the system, the system then asks the user to input the appropriate knowledge:

1 S(system) : Input the part class, please

U(user) : Gears

S: There is not this type of knowledge, do you want to provide some (y/n)?

U: y

S: Input sub-class please

2 U: spur gears

S: Input rule number please

U: 1

S: Input diameter range

U: D1, D2

S: Input working condition(Heavy, normal, light, accurate)

U: normal

S: application(specific device, such as motor, conveyor)

U: machine tool

S: Input fit class please

U: n5

S: More knowledge(y/n)?

U: y go to 2, otherwise go to 3

3 S: Modification(y/n)?

U: y, go to 4 otherwise go to 5

4 S: Input rule number

U: Give the rule number

S: Input diameter range

U: D1, D2

S: Input working condition(Heavy, normal, light, accurate)

U: Give a condition

S: application(specific device, such as motor, conveyor)

U: Provide an application example

S: Input fit class please

U: such as n5

5 stop knowledge acquisition.

These production rules are built into the system, and tolerances can

be inferred by the system when the inference engine is activated. It is assumed that the user knows the basic requirements for the loading condition of the designed part and accurate requirements. During the consultation process, the user is asked to choose from the results recommended by the expert consultant. Since the general user understands what is being designed, the question from the computer can be easily answered. Therefore, whenever users input requests, the system can automatically infer consultant results. If one of the recommendations is confirmed by the user, the system can then provide the final tolerances. An example is provided later.

The search algorithm is breadth-first. The pattern matching process is a direct searching of rules and facts in the knowledge base to satisfy the goal. The control structure is forward chaining from the top level to a lower level until the pattern matching is completely successful and the goal is satisfied. The goal will not be achieved if required knowledge does not exist. First of all, the index is searched and if the class is matched, a group of consultant results are provided. If it cannot be satisfied, the goal is unsatisfied and the system will ask the user to provide additional knowledge or make some knowledge modification or both. After the first group of consultant results are inferred, the user will then need to make a decision. Once one of the consultations is chosen, more detailed consultant results are given. When the user decides which is the most suitable recommendation, the final tolerances are inferred. This consultant process is shown in Figure 3.7.

If the results are not satisfactory to the user, or some knowledge is not correct, the user can modify or provide more knowledge. The

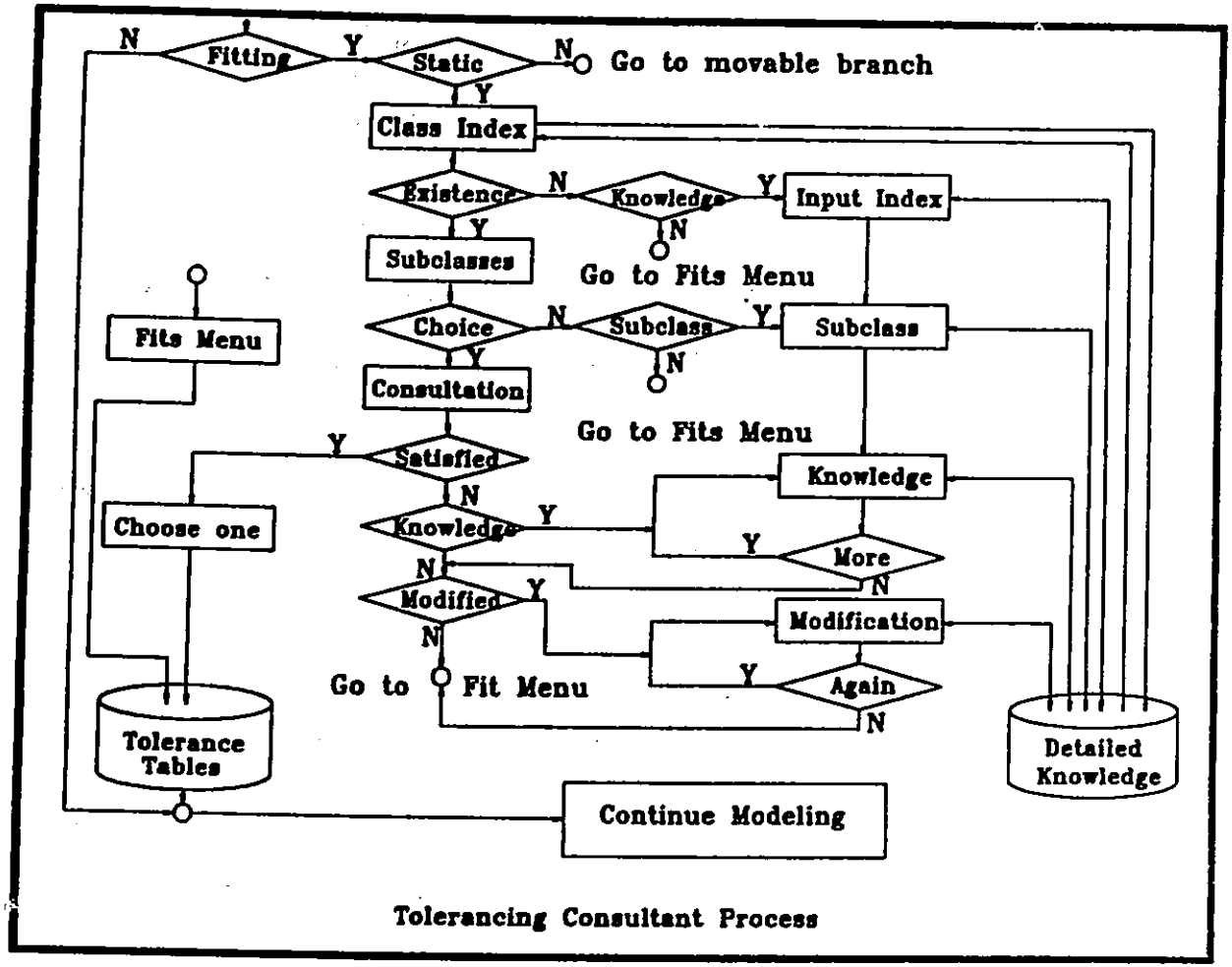


Figure 3.7 Tolerancing Consultant Process

system can capture this knowledge and store it in an appropriate place for future use as discussed before. It is possible in some cases that the user may want to assign the fits and tolerances directly. The expert system provides a facility whereby the user can choose the fits using a screen menu, and the system will then directly define the related tolerances.

This inference mechanism possesses a significant advantage in that it separates the search algorithm from the knowledge base. Thus, the knowledge base can be accessed when modifications are needed or some new knowledge becomes available.

Geometric tolerance assignments are more difficult than those of fitting and require more experienced experts because there are several ways to reach the individual accuracy criteria. Different tolerance assignments can result in different degrees of difficulty for machining, therefore, correct tolerance assignments are required to ensure that the manufacturing cost becomes a minimum, based on optimum combinations of various kinds of tolerances (ElMaraghy et al, 1988). Therefore, expert tolerancing functions play the key role for these assignments.

The geometric tolerances are more dependent on the parts function, accuracy, and mutual relations, this means that the correct relations of two parts are associated with geometric tolerances. The geometric tolerances are divided into 14 types based the ISO system (Foster, 1986):

FORM TOLERANCES	PROFILE TOLERANCES
flatness	profile of a line
straightness	profile of a surface
circularity	ORIENTATION TOLERANCES
cylindricity	perpendicularity
	angularity
LOCATION TOLERANCES	parallelism
position	RUNOUT TOLERANCES
concentricity	circular runout
symmetry	total runout

Since the 14 types of geometric tolerances are provided in the form of a menu, the user can choose them directly during the interactive modelling process. For example, when shaft concentricity is required to be assigned, the user simply chooses from the menu as discussed before, and the tolerance data are passed to the data file. A scheme which combines expert system techniques and optimization for geometric tolerancing has been discussed by Gu and ElMaraghy (1988).

3.3 Examples

The following is a discussion of two design examples, one a rotational and the other a nonrotational part. For the rotational part, the modeler is first run to indicate that the part is rotational. After that, the shaft is chosen as a general part class. Since the shaft consists of a number of features, each must be modeled. Once cylinder_front is chosen

as a modeling feature, the system asks for the radius, centre point coordinates, the angles between the normal vector of the feature and the three coordinate axes. Another question the system asks is if it has a dimensional tolerance. If the answer is yes, the two upper and lower limits and the one reference feature identifier are input by the user. Then, if any, geometric tolerances must be specified. If the feature does not have any geometric tolerances, the user must input surface-finish as micro inches to terminate the feature. There are several cylinders on the shaft. The cylinder is considered an important feature. It may have tolerances and relations to other parts. A similar procedure must be followed, such as inputting the radius, relations, geometric tolerances, and surface-finish. If the cylinder fits into another part, for example a bearing, the user should answer the relations question with 'yes'. Then, choosing the parts with a relation, the user consults with the Expert Tolerancing Consultant to determine the tolerances. This process is explained as follows: Since the cylinder is mounted on a bearing, the user should identify to the system the proper fitting. The user is then asked what part will be assembled on the cylinder after the bearing has been entered into the system. The system provides a menu of several types of bearings from which the user chooses. They are:

Ball Bearings

Cylinder & Taper Roller bearings

Spherical Roller Bearings

Thrust bearings

Radial with Taper bore & Sleeve

If ball bearing is the required type, the system searches the database to

find possible candidates based on the available information of bearing type and diameter of cylinder. The following recommendations will be made:

working condition	application example	fits
normal & heavy loads	motor & engine	n6
axial loads	all kinds	j6

By providing these consultant results to the user, the user can decide which situation is closest to her/his case. When the most suitable situation has been chosen, the system automatically searches for the final tolerances, and the corresponding grade fits and tolerances. It then displays the results onto the screen, in the following manner:

cylinder ball bearing

transition fits LT5

diameter tolerances

8.0 in. 2.60, 1.40 thou.

It then passes this to the output data file (see Figure 3.8). The whole inference process is exactly as previously described. The internal surface (i.e. round hole) can be treated in the same way. The only differences are the tolerances and fitting objects.

If the user wants to define this information himself, the user can choose the item of self-define in the menu. Following that, all the fit classes are provided in the menu again. Once the user picks one of them, the detailed grades are shown on the screen. If the user selects one as desireable, the tolerances are displayed on the screen and written in the output code.

For geometric tolerances, the user responds with 'yes' to the system's request, all types of geometric tolerances are displayed. If one is

confirmed, the system expects the user to input the values and datums as required. For example, if the cylinder has a concentricity tolerance, the "concentricity" option in the menu is chosen. Then, the system asks the value and reference feature which may be one, two even three datums. All questions must be answered completely and all essential data must be input. After all the data is input into the system, each feature image is displayed on the screen. The whole part is modeled when all the features are completed. The resulting shaft is shown in Figure 3.8 and the output file is in Figure 3.9. Another example, a prismatic part is shown in Figure 3.10, and its output file is shown in Figure 3.11. These two parts are processed by the FDDL system and the corresponding codes for assignment of parts to the machine cells and for inspection planning are complete and are discussed in Chapter 4 and 5.

3.4 Discussion

With feature representation as a medium for design and manufacturing, the most significant point is that the features are meaningful, both for geometric modeling and manufacturing processes. In this Chapter, the focus is on the relationship between the feature-based modeler and a general geometric modeler. A geometric modeler usually has its own data structure in its data base, and it may only allow a very restricted input format for modeling. The feature-based modeler links the geometric modeler by creating an interface which performs the function of translating the features in terms of the geometric entities definitions used by the geometric modeler. The feature-based modeler produces a data file for describing the modeled part or product. This data file contains all information for driving the manufacturing application systems. Herein, the

data file is designed in the FDDL syntax. This data file can be further processed by the FDDL system to create codes for expert systems of inspection and part assignments. Another important idea is that, in the feature-based modeler environment, expert system modules may be designed to assist the user in such matter as design for assembly, or design for manufacture. In this work, an Expert Tolerancing Module has been built in.

This modeler prototype is only used for a demonstration of the concepts. It is neither a complete geometric modeler nor is it perfect. Also, a great deal of effort is required to use GKS as the graphic presentation of the feature-based modeler. Thus, further development of the feature-based modeler should use a higher level and more advanced geometric modeler.

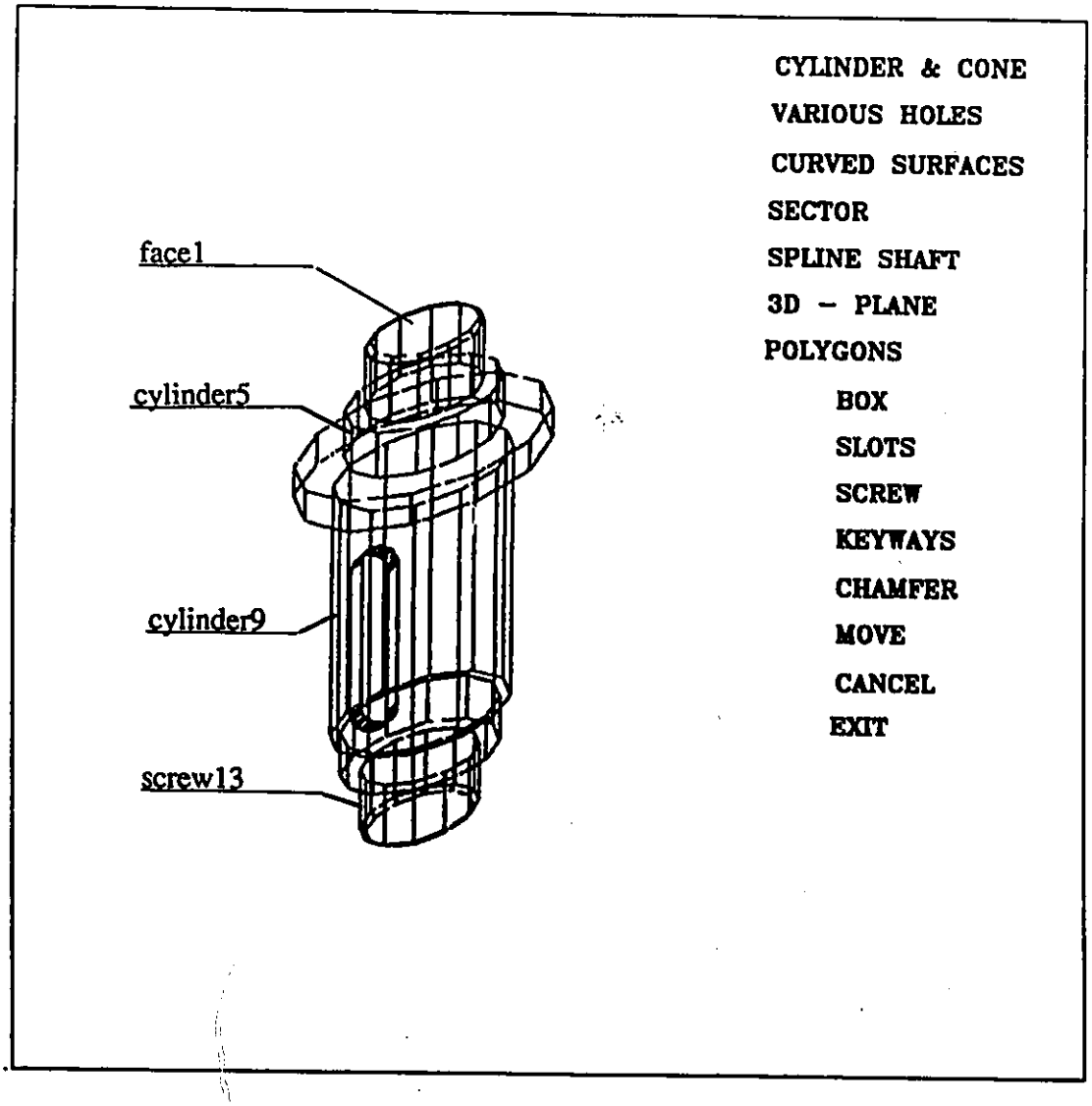


Figure 3.8 Sample Shaft Modeled by Feature-Based Modeler

```
part (name (shaft1)
          , class (10)
          , material (aisi_1330)
          , heat_treating (aging)
          )
     (feature (name (face1)
                    , type (cylinder_front, 5.40)
                    , location (20.0, 41.30, 20.0, 90.0, 0.0, -90.0)
                    , surface_finish (100.0)
                    )
              feature (name (chamfer2)
                    , type (chamfer, 5.40, 0.30, 6.0)
                    , location (20.0, 41.30, 20.0, 90.0, 0.0, -90.0)
                    , surface_finish (100.0)
                    )
              feature (name (screw3)
                    , type (thread_nc_side, 6.0, 4.0, 9.0)
                    , location (20.0, 41.0, 20.0, 90.0, 0.0, -90.0)
                    , relation (screw_fix, nut1)
                    , surface_finish (100.0)
                    )
              feature (name (face3)
                    , type (cylinder_front, 8.0)
                    , location (20.0, 37.0, 20.0, 90.0, 0.0, -90.0)
                    , relation (contact_fix, net1)
                    , tolerance (distance, -2.50, 2.0, face6)
                    , surface_finish (100.0)
                    )
              feature (name (cylinder5)
                    , type (cylinder_side, 8.0, 3.0)
                    , location (20.0, 37.0, 20.0, 90.0, 0.0, -90.0)
                    , relation (pfit, bearing2)
                    , tolerance (diameter, 2.60, 1.40)
                    , surface_finish (30.0)
                    )
              feature (name (face6)
                    , type (cylinder_front, 13.0)
                    , location (20.0, 34.0, 20.0, 90.0, 0.0, -90.0)
                    , relation (contact_free, bearing2)
                    , tolerance (perpendicularity, 0.80, cylinder5)
                    , surface_finish (50.0)
                    )
              feature (name (cylinder7)
                    , type (cylinder_side, 13.0, 2.0)
                    , location (20.0, 34.0, 20.0, 90.0, 0.0, -90.0)
                    , surface_finish (100.0)
                    )
```

Figure 3.9 Formatted Source Language for the Modeled Shaft Shown in Figure 3.8

```
feature (name (face8)
        , type (cylinder_front, 9.0)
        , location (20.0, 32.0, 20.0, -90.0, 180.0, 90.0)
        , relation (contact_free, gear3)
        , tolerance (distance, -1.50, 2.0, face6)
        , tolerance (parallelism, 1.0, face6)
        , surface_finish (50.0)
        )
feature (name (cylinder9)
        , type (cylinder_side, 9.0, 17.0)
        , location (20.0, 32.0, 20.0, 90.0, 0.0, -90.0)
        , relation (pufit, gear3)
        , tolerance (diameter, 0.60, -0.60)
        , tolerance (concentricity, 1.0, cylinder5)
        , surface_finish (30.0)
        )
feature (name (face10)
        , type (cylinder_front, 9.0)
        , location (20.0, 15.0, 20.0, 90.0, 0.0, -90.0)
        , relation (contact_free, bearing4)
        , tolerance (distance, -1.50, 1.0, face8)
        , tolerance (perpendicularity, 1.0, cylinder11)
        , surface_finish (50.0)
        )
feature (name (cylinder11)
        , type (cylinder_side, 8.0, 3.0)
        , location (20.0, 15.0, 20.0, 90.0, 0.0, -90.0)
        , relation (pfit, bearing4)
        , tolerance (diameter, 2.60, 1.40)
        , tolerance (concentricity, 0.80, cylinder5)
        , surface_finish (30.0)
        )
feature (name (face12er11)
        , type (cylinder_front, 8.0)
        , location (20.0, 12.0, 20.0, -90.0, 180.0, 90.0)
        , relation (contact_fix, net5)
        , tolerance (distance, -1.50, 1.0, face10)
        , surface_finish (100.0)
        )
feature (name (screw13)
        , type (thread_nc_side, 6.0, 4.0, 9.0)
        , location (20.0, 12.0, 20.0, 90.0, 0.0, -90.0)
        , relation (screw_fix, nut5)
        , surface_finish (100.0)
        )
```

Figure 3.9 Formatted Source Language for the Modeled Shaft Shown in Figure 3.8 (Cont.)

```
feature (name (chamfer14)
        , type (chamfer, 6.0, 0.30, 5.40)
        , location (20.0, 8.0, 20.0, 90.0, 0.0, -90.0)
        , surface_finish (100.0)
        )
feature (name (face15)
        , type (cylinder_front, 5.40)
        , location (20.0, 7.70, 20.0, -90.0, 180.0, 90.0)
        , surface_finish (100.0)
        )
feature (name (side16)
        , type (p_keyseat_side, 0.90, 10.0)
        , location (21.1250, 33.0, 15.950, 0.0, -90.0, 90.0)
        , relation (kfit, key6)
        , tolerance (distance, -0.50, 0.50, side17)
        , surface_finish (50.0)
        )
feature (name (side17)
        , type (p_keyseat_side, 0.90, 10.0)
        , location (18.8750, 33.0, 15.950, 180.0, 90.0, -90.0)
        , relation (kfit, key6)
        , surface_finish (50.0)
        )
feature (name (corner18)
        , type (p_keyseat_corner, 1.1250, 0.90)
        , location (20.0, 28.0, 15.50, 90.0, -90.0, 180.0)
        , surface_finish (50.0)
        )
feature (name (corner19)
        , type (p_keyseat_corner, 1.1250, 0.90)
        , location (20.0, 18.0, 15.50, 90.0, -90.0, 180.0)
        , surface_finish (50.0)
        )
)
```

Figure 3.9 Formatted Source Language for the Modeled Shaft Shown in Figure 3.8 (Cont.)

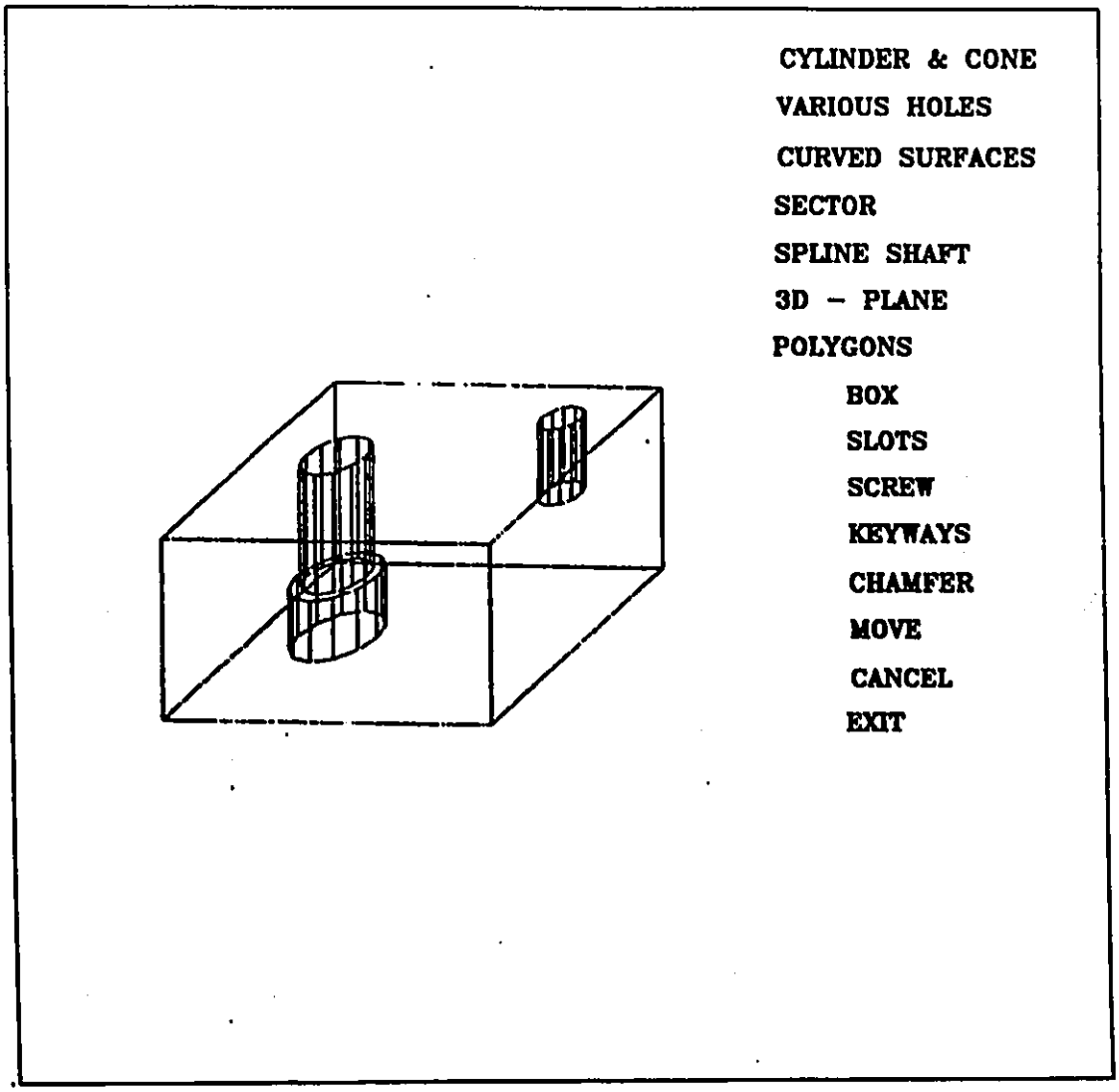


Figure 3.10 Sample Prismatic Part Modeled By Feature-Based Modeler

```
part (name (box2)
          , class (10)
          , material (aisi_1330)
          , heat_treating (aging)
          )
     (feature (name (hole1)
                , type (bore, 2.40, 5.0)
                , location (26.0, 30.0, 22.0, 90.0, 0.0, -90.0)
                , relation (pfit, shaft1)
                , tolerance (diameter, 1.80, 0.0)
                , tolerance (parallelism, 1.60, hole8)
                , tolerance (distance, 1.50, -1.50, face5)
                , tolerance (distance, 1.50, -1.50, face7)
                , surface_finish (40.0)
                )
              feature (name (face2)
                , type (box_face, 20.0, 15.0)
                , location (20.0, 30.0, 25.0, 90.0, 0.0, -90.0)
                , tolerance (flatness, 0.60)
                , surface_finish (30.0)
                )
              feature (name (face3)
                , type (box_face, 20.0, 12.0)
                , location (20.0, 30.0, 10.0, 90.0, -90.0, 180.0)
                , tolerance (distance, 1.20, -0.50, face4)
                , tolerance (parallelism, 1.0, face4)
                , surface_finish (50.0)
                )
              feature (name (face4)
                , type (box_face, 20.0, 12.0)
                , location (20.0, 30.0, 25.0, -90.0, 90.0, 0.0)
                , tolerance (perpendicularity, 0.80, face2)
                , surface_finish (50.0)
                )
              feature (name (face5)
                , type (box_face, 12.0, 15.0)
                , location (10.0, 18.0, 25.0, 180.0, 90.0, -90.0)
                , tolerance (perpendicularity, 1.50, face2)
                , surface_finish (100.0)
                )
              feature (name (face6)
                , type (box_face, 12.0, 15.0)
                , location (30.0, 18.0, 25.0, 0.0, -90.0, 90.0)
                , surface_finish (100.0)
                )
```

Figure 3.11 Formatted Source Language for the Modeled Prismatic Part Shown in Figure 3.10

```
feature (name (face7)
        , type (box_face, 20.0, 15.0)
        , location (20.0, 18.0, 25.0, -90.0, 180.0, 90.0)
        , tolerance (parallelism, 1.0, face2)
        , surface_finish (50.0)
        )
feature (name (hole8)
        , type (bore, 3.60, 8.0)
        , location (15.0, 30.0, 18.0, -90.0, 180.0, 90.0)
        , relation (mfit, shaft1)
        , tolerance (diameter, 1.40, 0.0)
        , tolerance (distance, 1.50, -1.50, face5)
        , tolerance (distance, 1.50, -1.50, face3)
        , surface_finish (40.0)
        )
feature (name (hole9)
        , type (bore, 4.80, 4.0)
        , location (15.0, 22.0, 18.0, -90.0, 180.0, 90.0)
        , relation (mfit, shaft1)
        , tolerance (diameter, 1.60, 0.0)
        , tolerance (distance, 2.50, 1.0, face4)
        , tolerance (concentricity, 1.0, hole8)
        , surface_finish (40.0)
        )
)
```

Figure 3.11 Formatted Source Language for the Modeled Prismatic Part Shown in Figure 3.10 (Count.)

CHAPTER 4

FEATURE-BASED CELLULAR MANUFACTURING PLANNING USING EXPERT SYSTEM AND PATTERN RECOGNITION TECHNIQUES

4.1 Introduction

Once component design is completed, it is usually planned for manufacturing. This chapter presents a feature-based cellular manufacturing planning system which consists of a cellular manufacturing design module and a feature-based assignment system. The integration of this assignment system with the feature-based design is also described.

Cellular manufacturing enables small batch size production to achieve high productivity, and low cost which is a feature normally associated with mass production. Manufacturing cells may be used by themselves or as modules in flexible manufacturing systems. Part classification and grouping into families with similar geometric and processing attributes are the basic concept leading to the formation of manufacturing cells and is also a prerequisite for the successful development of any flexible manufacturing system. As discussed before, detailed process planning for cutting has gained much attention research in the community. This project is concerned with a higher level of planning, i.e. the integration of CAD and machine cells. Two aspects of cellular manufacturing are considered: the design of machine cells and formation of part families, and the automated assignment of parts to the machine cells directly from a CAD data base.

4.2 Grouping Formation by Cluster-Seeking Approach

4.2.1 Introduction

The two main approaches in Group Technology generally applied to forming machine cells and grouping parts into families are coding systems and Production Flow Analysis (PFA). The first coding system was developed by Optiz (1970). A number of coding systems such as MICLASS, KK, DCLASS and COFORM (Rembold et al., 1985) have also been developed. Parts coding and classification systems are concerned with the description of the parts characteristics. The Production Flow Analysis, originally developed by Burbidge, deals with the sequence of processes in which parts are produced. PFA is an analytical technique which determines the groups and families by analyzing the information given in the components route cards.

Grouping formation of part families and machine cells using PFA are generally carried out by employing various machine-components grouping algorithms. A machine-component matrix is used as input for these algorithms. By manipulating this matrix, the block diagonal form is derived as a result of formation. During the formation process, it is not always possible that all components of a part family be manufactured within a single cell. Thus, some components may visit more than one cell in the process of their manufacturing. Such components are defined as "exceptional parts". The corresponding visited machines from other cells are defined as "bottleneck machines". It is preferable that no bottleneck machines exist for the final cells configuration. If bottleneck machines do exist, the employed algorithm should identify and minimize the number of

bottleneck machines.

A number of algorithms for parts grouping have been developed. Burbidge (1971) presented a manual technique used to form part families and machine groups which are particularly suitable for small size problems. The simple linkage cluster analysis approach of numerical taxonomy was applied by McAulley (1972). A similarity coefficient for any machine pair is computed and a tree diagram called a dendragram is constructed. The dendragram is simply a pictorial representation of the bonds of similarity between machines. This approach could be used manually for simple problems. For larger problems, a minimum spanning tree method was proposed by McAulley, based on the algorithms given by Ross (1969). McCormick et al (1972) proposed the bond energy approach which is defined as the product of the adjoining element in the machine-component matrix. This method requires long computing time and heuristic methods must be used for problems of realistic size, leading to approximate solutions.

Recently, some computerized algorithms have been developed. An iterative algorithm is implemented on computer by King (1980), called the Rank Order Clustering (ROC) algorithm, which is designed to generate diagonalized groupings of the machine-component matrix. The algorithm rearranges the rows and columns of the machine-component matrix in an iterative manner that eventually, in a finite number of steps, produces a matrix in which both rows and columns are arranged in order of decreasing value when read as binary words. The algorithm is simple and can easily be implemented on computers, but it cannot provide final mutually independent groupings.

Chan and Milner (1982) developed an approach called Direct Clustering Algorithm (DCA) which forms families and groups by using blocks and rods, and by changing the sequence in which components and machines are listed in the matrix. Based on the available description both the ROC and the DCA algorithms contain six solution steps and the same termination criterion. The solution obtained by the Direct Clustering Algorithm is identical to Burbidge's trial-and-error result. Based on the analysis of solutions to the same problem by Burbidge, Chan and Milner, and King, the one difference between the algorithms is that King's solution will produce a group which is divided into two by the others. A common feature among these algorithms is the iterative machine-component matrix manipulation. All solutions contain some exceptional components, and they provide only one configuration of machine cells and component families.

Han and Ham (1986) reported the multi-objective cluster analysis for part family formation using Goal Programming based on the concept of group technology and the parts coding system, where all parts are coded using the coding system and the method is then applied to form the part families. Wu et al (1986) applied a syntactical pattern recognition method for the design of a cellular manufacturing system. The results show that the task of grouping components to form families can be carried out by means of syntactic pattern recognition techniques.

This chapter presents the cluster-seeking algorithms and the optimization techniques used to group parts into families and form machines into cells. The K-means, revised K-means algorithms and the combination of the Isodata algorithm and the optimization strategies are

discussed.

4.2.2 Cluster-Seeking Algorithms

The group formation of cellular manufacturing can be viewed as unsupervised learning only if one set of component routes are available. The unsupervised learning problem is one of identifying classes in the given set of patterns, which are components in the context of this work, in order to group all given components into families, and form the machines into cells. The application of cluster-seeking algorithms is, in principle, straightforward. Suppose that a set of components $\{X_1, X_2, ..., X_N\}$ has known operation routes and that the component families and associated machine cells are unknown. The following algorithms may be used to identify representative cluster centers. The resulting cluster domain may then be interpreted as different component families and associated machine cells. The cluster centers are reference points in the pattern space and the number of the centers indicate the number of part families and machine cells (see Figure 4.1).

In statistical pattern recognition, a pattern (a component in this context) is usually expressed as a vector:

$$X = [x_1, x_2, ..., x_n]^T \tag{4.1}$$

Each element of the vector represents one dimensional attribute. In this problem, X is a component and x_i, i = 1, 2, ..., n are process routes which are associated with machines. If a machine is required for a process, $x_i = 1$ otherwise $x_i = 0$.

Since the components are clustered, the corresponding similarity between them should be determined. In the cluster-seeking algorithms, the measure of similarity is defined as the Euclidian distance between the two

pattern vectors X and Z (Tou and Gonzalez, 1974):

$$D = \|X - Z\| \tag{4.2}$$

Based on the measure of similarity between the patterns representing the process plans, a clustering criterion is required for partitioning the given data into components families and associated machine cells. The clustering criterion used may represent a heuristic scheme, or it may be based on the optimization of a certain performance index.

The heuristic scheme is usually guided by intuition and experience. A set of rules are used to group components into families and machines into cells, for example, the choice of first centers for the clustering domains. Since the Euclidian distance measure is a relative measure of similarity, it is necessary to set a threshold to define the degree of acceptances in the cluster-seeking process. Also, the threshold is determined by experience or through some experimental computations. One typical heuristic algorithm is the maximum distance method (Tou and Gonzalez, 1974). The heuristic approach is simple and easy to implement. However clustering results depend on that first chosen cluster centre, the order of components considered, and the threshold value. Therefore, it can be an ideal method to roughly estimate the properties of a given set of components.

The optimization of a performance-index approach is based on the chosen index. One of the commonly used indices is the sum of the squared errors:

$$J = \sum_{j=1}^{N_c} \sum_{x \in S_j} \| X - \mu_j \|^2 \tag{4.3}$$

Where N_c is the number of cluster domains, i.e. components families, S_j is

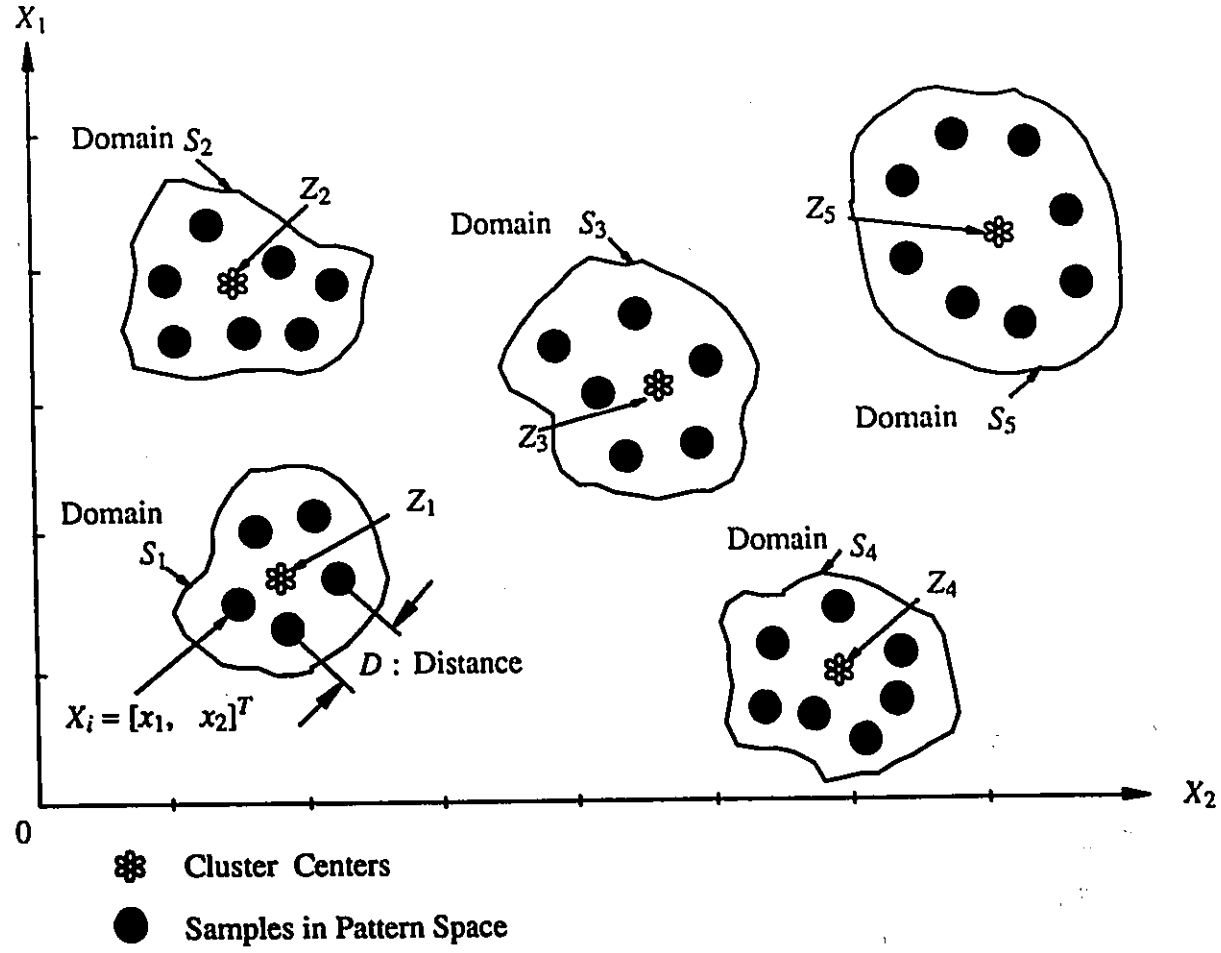


Figure 4.1 Two Dimensional Example of Clustering

the set of samples belonging to the jth domain, i.e. jth group of components in this research, and μj is the component mean vector of set S_j. N_j here represents the number of samples in S_j and X is a sample i.e. one component. μ_j is computed as follows:

$$\mu_j = \frac{1}{N_j} \sum_{x \in S_j} X \tag{4.4}$$

From this simple description, the performance-index approach appears to be better than the heuristic one since it involves optimization. One of the performance-index algorithms, the K-means, has problems similar to the heuristic scheme. The behaviour of the K-means approach depends upon the number of cluster domains (groups) desired, the choice of the initial cluster centers, and the order in which the samples are taken. Various values of K and initial cluster centers, therefore, should be tested in order to obtain satisfactory results.

4.2.3 Revised K-means Algorithm

To overcome these drawbacks, a revised K-means algorithm is proposed and implemented in this work. The basic idea is to combine the advantages of both the maximum distance and K-means algorithms. The K-means approach is influenced by the number and choice of cluster centers. Arbitrary choice of the initial cluster centers does not yield the best results. If the desired number of cluster domains (groups of machines or components families) and initial centers are chosen based on some criteria, the K-means algorithm could yield better solutions. From the clustering point of view, patterns should be classified into classes which result in significant differences between those centers. The maximum distance between patterns, as described earlier, is suitable for initially finding out

the number of domains and cluster centers. The K-means algorithm is discussed in (Tou and Gonzalez, 1982). The revised K-means algorithm is described below. Steps 1 to 3 are based on the use of the maximum distance criterion in order to set up the initial appropriate cluster centers. Steps 4 to 6 are similar to the original K-means algorithm:

1. To find all possible initial cluster centers; set a threshold T as a criterion for determining centers if distance exceeds T.
2. Randomly choose a first cluster centre for a given set of samples.
3. Compare all components with cluster centers. The cluster centers are incremented based on the following expression:

$$\sum_{j=1}^{m} | X_{ij} - Z_{pj} | > T \qquad \begin{matrix} i = 1, 2, ..., N, \\ p = 1, 2, ..., K \end{matrix} \qquad (4.5)$$

Where X_{ij} is jth component of ith sample in the sample set. Z_{pj} is jth component of pth cluster centre. N is the number of samples (components). K is the number of cluster centers. This process determines all initial cluster centers.

4. Start iteration, at the kth (lower case k) iteration, distribute the samples {X} (components set) among the K cluster domains (component families), using the relation:

$$X \in S_j(k) \quad \text{if} \| X - Z_j(k) \| < \| X - Z_i(k) \| \qquad (4.6)$$

for all $i = 1,2,...,K$, $i \neq j$, where $S_j(k)$ denotes the set of components whose cluster centre is $Z_j(k)$.

5. After new domains (groups) have been formed, update cluster centers such that the sum of the squared distances from all points in $S_j(k)$ to the new cluster centers is minimized. In other words, the new cluster centre $Z_j(k)$ is computed so that the performance

index is optimised:

$$J = \sum_{x \in S_j(k)} \| X - Z_j(k+1) \|^2 \qquad (4.7)$$

The $Z_j(k+1)$ which minimizes this performance index is simply the component mean of $S_j(k)$. The new cluster centre, therefore, is determined by:

$$Z_j(k+1) = \frac{1}{N_j} \sum_{x \in S_j} X \qquad (4.8)$$

where N_j is the number of samples in $S_j(k)$.

6. Check convergence of iteration using:

$$| Z_{ij}(k+1) - Z_{ij}(k) | \leq e \qquad i=1,2,..., K, \qquad j=1,2,...,n \qquad (4.9)$$

If it is satisfied for all i and j, then the iteration terminates, otherwise go to 4.

Application examples of this scheme are shown in Figures 4.3 and Figure 4.4. The results show that for the same number of cells, fewer machines are required by the Revised K-means than by the K-means. This will be discussed in more detail later.

4.2.4 Isodata Algorithm

The Isodata algorithm (Tou and Gonzalez, 1982) is similar in principle to the K-means algorithm in the sense that cluster centers are iteratively determined by sample means. However, the Isodata includes a fairly comprehensive set of additional heuristic procedures which have been incorporated into an interactive scheme such as following step 4 to 12.

Just as the K-means algorithm, the Isodata requires a set N_c of initial cluster centers, Z_1, Z_2, ..., Z_{Nc}. This set need not necessarily be equal in number to the desired cluster centers and can be formed by selecting samples from the given set of components. Other parameters

which should be specified before the execution of the iteration or at the first step of the iteration are:

K = number of cluster domains (groups) desired;

N_c = K at beginning of iteration;

Q_N = a parameter against which the number of samples in a cluster domain is compared;

Q_s = standard deviation parameter;

Q_c = lumping parameter;

L = maximum number of pairs of cluster centers which can be lumped;

I = maximum number of iterations allowed;

e = convergence criteria

The Isodata Algorithm is described below:

1. Distribute the N samples(components) among the present cluster centers, using the relation:

   $X \in S_j$ if $\|X - Z_j\| < \|X - Z_i\|$ $\quad i=1,2,\ldots, N_c, \quad i \neq j$ (4.10)

   for all X in the components set, S_j represents the subset of components assigned to cluster centre Z_j.

2. Discard sample subsets with fewer than Q_N members; that is, if for any j, $N_j < Q_N$, discard S_j and reduce N_c by 1.

3. Update each cluster centre Z_j, j=1, 2, ..., N_c by setting it equal to the sample mean of its corresponding set S_j;

$$Z_j = \frac{1}{N_j} \sum_{x \in S_j} X \qquad j = 1, 2, \ldots, N_c \tag{4.11}$$

   where N_j is the number of components in S_j.

4. Compute the average distance D_j of components in the cluster domain S_j from their corresponding cluster centre, using the

relation:

$$D_j = \frac{1}{N} \Sigma \|X - Z_j\|, \qquad j = 1, 2, ..., N_c \tag{4.12}$$

5. Compute the overall average distance of the components from their respective cluster centers, using the relation:

$$D = \frac{1}{N} \sum_{j=1}^{N_c} N_j D_j \tag{4.13}$$

6. If this is the last iteration, set $Q_c = 0$ and go to step 10. If this is an even-numbered iteration, or if $N_c \geq 2K$, go to 10 also; otherwise, continue.

7. Find the standard deviation vector $\sigma_j = (\sigma_{1j}, \sigma_{1j}, ..., \sigma_{nj})^T$ for each component subset, using the relation

$$\sigma_{ij} = \sqrt{\frac{1}{N_j} \sum_{x \in S_j} (x_{ik} - z_{ij})^2} \qquad \begin{matrix} i = 1, 2, ..., n, \\ j = 1, 2, ..., N_c \end{matrix} \tag{4.14}$$

where n is the sample dimensionality, x_{ik} is the ith component of the kth sample in S_j, z_{ij} is the ith component of Z_j and N_j is the number of components in S_j. Each component of σ_j represents the standard deviation of the samples in S_j along a principal coordinate axis.

8. Find the maximum component of each σ_j, $j = 1, 2, ..., N_c$ and denote it by σ_{max}.

9. If for any σ_{jmax}, $j = 1, 2, ..., N_c$, we have $\sigma_{jmax} > Q_s$ and

(a) $D_j > D$ and $N_j > 2(Q_N + 1)$ (4.15)

or

(b) $N_c \leq K/2$ (4.16)

then split Z_j into two new cluster centers Z_j^+ and Z_j^-, and

increase N_c by 1. Cluster centers Z_j^+ and Z_j^- can be determined by adding a given quantity β_j to the component of Z_j which corresponds to the maximum component of σ_j; Z_j^- is formed by subtracting β_j from the same component of Z_j. One way of specifying β_j is to let it be equal to some fraction of σ_{jmam}, that is, $\beta_j = k^*\sigma_{jmax}$, where $0 < k \leq 1$. The basic requirement in choosing β_i is that it should be sufficient to provide a detectable difference in the distance from an arbitrary sample to the two new cluster centers, but not so large as to change the overall cluster domain arrangement appreciably. If cluster splitting took place in this step, go to 1; otherwise continue.

10. Compute the pair-wise distance D_{ij} between all cluster centers:

$$D_{ij} = \| Z_i - Z_j \| \qquad i = 1, 2, ..., N_c-1, \quad j = i+1, 2, ..., N_c \tag{4.17}$$

11. Compare the distance D_{ij} against the parameter Q_c. Arrange the L smallest distances which are less than Q_c in ascending order:

$$[\, D_{i1j1}, D_{i2j2}, ..., D_{iLjL} \,]$$

where $D_{i1j1} < D_{i2j2} < ... < D_{iLjL}$ and L is the maximum number of pairs of cluster centers which can be lumped together.

12. With each distance D_{iljl} there is an associated pair of cluster centers Z_{il} and Z_{jl}. Starting with the smallest of these distances, perform a pair-wise lumping operation according to the following rule:

For l = 1, 2, ..., L if neither Z_{il} nor Z_{jl} has been used in lumping in this iteration, merge these two cluster centers using the following relation:

$$Z_l^* = \frac{1}{N_{il}+N_{jl}} [N_{il}*(Z_{il}) + N_{jl}*(Z_{jl})] \qquad (4.18)$$

Delete Z_{il} and Z_{jl} and reduce N_c by 1.

It should be noted that only pairwise lumping is allowed and that a lumped cluster centre is obtained by weighing each old cluster centre by the number of components in its domain. Experimental evidence indicates that more complex lumping can produce unsatisfactory results (Tou and Gonzalez, 1982). The above procedure makes the lumped cluster centers representative of the true average point of the combined subsets. It is also important to note that, since a cluster centre can be lumped only once, this step will not always result in L lumped centers.

13. Check convergence of iteration using criterion e set by the user

$$| Z_{ij}(k+1) - Z_{ij}(k) | < e \qquad i = 1, 2, ..., N_c, \quad j=1,2,...,n \qquad (4.19)$$

If the above expression for all i and j is satisfied, the iteration is terminated and all derived solutions are printed out. If it is not satisfied and this is the last iteration, the algorithm fails and an adjustment of parameters such as K, Q_n, Q_m, Q_c, L, or other should be made. Otherwise, go to 1.

This algorithm allows the user to change initial parameters during iterations. The parameters are defined before step 1. Also, the algorithm includes some important steps for splitting and lumping the cluster domains. These dynamic adjustments of the cluster domains make the final results more satisfactory. This algorithm also requires the user to specify

the several initial parameters. Only the optimum selection of these parameters can result in the best solution. The following case study illustrates the Isodata algorithm and the selection of these parameters by optimization techniques.

An example is chosen from Chan and Milner (1982) and is used initially to test the algorithms, and the original machine-component matrix is shown in Figure 4.2 (a). The solution derived by the K-means, the Revised K-means and the Isodata algorithm are given in Figure 4.2 (b). Examples of using the revised K-means algorithm are shown in a case study in the following section. From this simple example, the results obtained using the two algorithms are identical. However, Isodata with optimization techniques becomes superior as the problem becomes more complicated, as is shown in Figure 4.6. This will be illustrated further in a case study.

4.2.5 Combination of Optimization and Isodata Algorithm

Although the Isodata algorithm seems more comprehensive and can produce better results compared to the K-means and Revised K-means, intensive experimental computations are still required to determine the optimum combinations of the parameters (Gu and ElMaraghy, 1989). Obviously, this is very time consuming. The optimum parameters may be determined by some optimization strategies for the formation of part families and machine cells. A combination scheme of optimization and ISODATA algorithm is proposed and implemented to determine the optimum solutions. The idea is to minimize the number of bottleneck machines necessary for the final cells configuration. This is equivalent to minimizing the total number of machines required to form the cells. Therefore, the

Comp. No.	M/C No. 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		x								x	x	x			
2			x		x			x					x		x
3	x					x			x					x	
4	x			x					x					x	
5			x		x			x					x		x
6	x			x		x			x					x	
7		x					x			x	x	x			
8			x		x			x					x		x
9				x		x			x					x	
10		x					x			x	x	x			

Figure 4.2(a) Example of Machine-Component Matrix

Comp. No.	M/C No. 2	7	10	11	12	3	5	8	13	15	1	4	6	9	14
1	x		x	x	x										
7	x	x	x	x	x										
10	x	x	x	x	x										
2						x	x	x	x	x					
5						x	x	x	x	x					
8						x	x	x	x	x					
3											x		x	x	x
4											x	x		x	x
6											x	x	x	x	x
9												x	x	x	x

Figure 4.2(b) Solution by K-Means and ISODATA

problem formulation is given as follows:

$$\text{Objective Function:} \qquad U = \sum_{i=1}^{N_c} C_i = \text{minimum} \tag{4.20}$$

$$\begin{aligned}
\text{Constraint Functions:} \qquad & \phi_1 = Q_{nmax} - Q_n \geq 0 \\
& \phi_2 = Q_n - Q_{nmin} \geq 0 \\
& \phi_3 = N_{cmax} - N_c \geq 0 \\
& \phi_4 = N_c - Nc_{min} \geq 0 \\
& \phi_5 = Qc_{max} - Q_c \geq 0 \\
& \phi_6 = Q_c - Qc_{min} \geq 0 \\
& \phi_7 = Qs_{max} - Q_s \geq 0 \\
& \phi_8 = Qs - Qs_{min} \geq 0
\end{aligned} \tag{4.21}$$

Where U is the total machines required to form all cells. C_i is machines number of ith cell. Q_n is a parameter against which the number of samples in a cluster domain is compared. N_c is the number of the cluster domains desired. Q_c is the lumping parameter and Q_s is the standard deviation parameter. The symbols with max. or min. refer to the upper and lower bounds.

Instead of specifying concrete parameters of Q_n, N_c, Q_c and Q_s, the user just determines regions in which the above four parameters are valid. The algorithm automatically determines the optimum solution for the given problem. During the optimization computation, C_i is determined by passing a set of values of Q_n, N_c, Q_c and Q_s through Isodata algorithm routine. In this way optimum solutions can be derived by changing the constraints of the upper and lower bounds.

The software has been developed which combines the Isodata and

optimization methods. The software requires only that the user specifies the specific values of these bounds. The system automatically finds out the optimum solution, the cells configuration and the part families as well as the bottleneck machines and their numbers.

4.2.6 Utilization of Machines in Formed Clusters

It has been shown that mutually independent components families and machine cells can be formed using the revised K-means and Isodata algorithms. However, the utilization of the formed cells should be examined before the solutions are implemented on the shop floor. If a machine utilization in a formed cell is low, the two following strategies are suggested to improve it: 1) re-clustering; and 2) process redesign or part redesign.

1. Re-clustering means that a utilization function (total value) should be determined to better guide the clustering process. This requires algorithms incorporated with the utilization evaluation in order to obtain different configurations.
2. Part redesign or process redesign means that the manufacturing operation which corresponds to the under-utilized machine should be re-designed by changing the process or part to allow the removal of that particular machine from the cell. This requires practical knowledge of the components geometric features and the manufacturing processes required to produce them, and is often a feasible alternative.

A Utilization Factor is defined in equation (4.22) as an indication of the utilization of a machine in a given cell.

$$f_i = \Sigma (T_i * P_i) \qquad (4.22)$$

Where f_i is the utilization factor of ith machine in a cell, T_i is the occupying time per part on the ith machine and P_i is the part number on ith machine. Once the thresholds are set for the machines in the cells, for example-85%, the machines whose utilizations are lower than this percent are recognized and corresponding policies may be applied.

4.2.7 A Case Study

The problem chosen for this case study is shown in Figure 4.3 and consisted of 43 components and 16 machine tools. It was originally provided by Burbidge (1971), who used a manual method to obtain a solution which consisted of five component groups and three exceptional components (Figure 4.7). This example was consequently used by King with his Rank Order Clustering algorithm, and by Chan et al with their Direct Clustering algorithm. The solution derived by King has four component groups with two exceptional components, (Figure 4.8). The result obtained by Chan has five component groups with three exceptional components (Figure 4.9), which is identical to the solution discovered by Burbidge. In order to compare the cluster-seeking algorithms developed in this work with the above algorithms, the same problem is used in this case study.

It is clear, by examining the above mentioned solution, that the final results are not totally independent. These algorithms cannot give different configurations of the cells with different numbers of components and memberships. All of these algorithms are based upon the manipulation of the machine-component matrix. The cluster-seeking algorithms presented in this Chapter can provide different configurations of the machine cells and the groupings of parts, with desired combinations. This feature makes this approach easily adaptable to a real production environment.

M/C	Component Number																																									
No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1																																					x					x	
2		x								x																		x				x					x	x		x		x	
3							x										x																	x	x	x							
4					x				x					x					x		x		x						x														
5					x			x	x					x	x	x			x		x		x						x				x										
6	x	x				x	x	x				x	x	x			x		x				x									x	x	x			x		x	x		x	x
7	x												x												x																		
8	x	x	x					x	x		x	x			x				x	x	x		x	x			x	x			x						x	x			x		x
9		x		x						x								x										x				x					x	x		x		x	
10	x											x	x												x	x					x								x				
11			x						x											x				x			x			x													
12											x											x		x			x			x													
13			x																					x																			
14		x				x											x																		x								
15						x								x					x		x												x								x		x
16		x					x			x								x														x					x	x				x	

Figure 4.3 The Machine-Components Matrix Presented by Burbidge (1971)

4.2.8 Results and Discussions

1. Comparison of Different Cluster-Seeking Strategies

The results in Figure 4.4 were derived by specifying the number of desired groups to be four and associated initial cluster centers arbitrarily as the first K components in the component set of 43. The criterion for stopping the iteration is when two sequential iteration results are the same. The results show that for a different K (K = 3, 4, and 5) and the same stopping criteria of two sequential iterations for cluster centers, e = 0.0001, independent cell configurations and component families are formed. It is clear that if these components are completely manufactured in these cells, extra machines of similar types such as machine No. 3, 4, 5, 6, 8, 14, 15 and 16 in Figure 4.4 (a) are required to form the above cells. It is possible to form better configurations by adjusting the initial cluster centers and K.

In the revised K-means algorithm, the final number of cells or component families depend on the desired threshold which is used as a criterion to determine the groupings. Therefore, K is controlled by this threshold, T (it is the maximum distance between a sample and cluster centre for the same group). If T is increased, K is decreased. The maximum distance between patterns is used to form the initial cluster centers. Thus, the clustering results are influenced by the first initial centre. In the original K-means algorithm, if the initial cluster centers defined by user are different, the K-means can produce the same number of groups with different configurations. For the revised K-means algorithm, the initial cluster centers are determined by applying the maximum

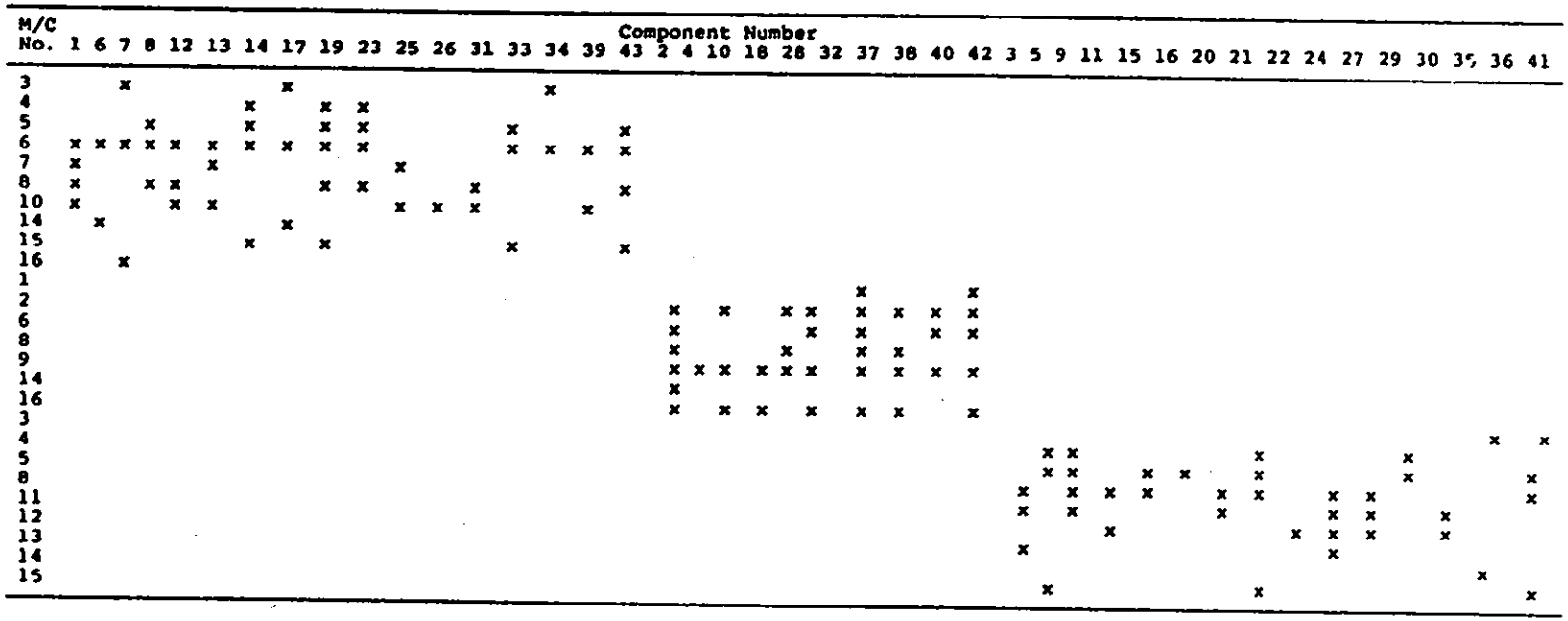


Figure 4.4(a) Solution by K-Means with K = 3

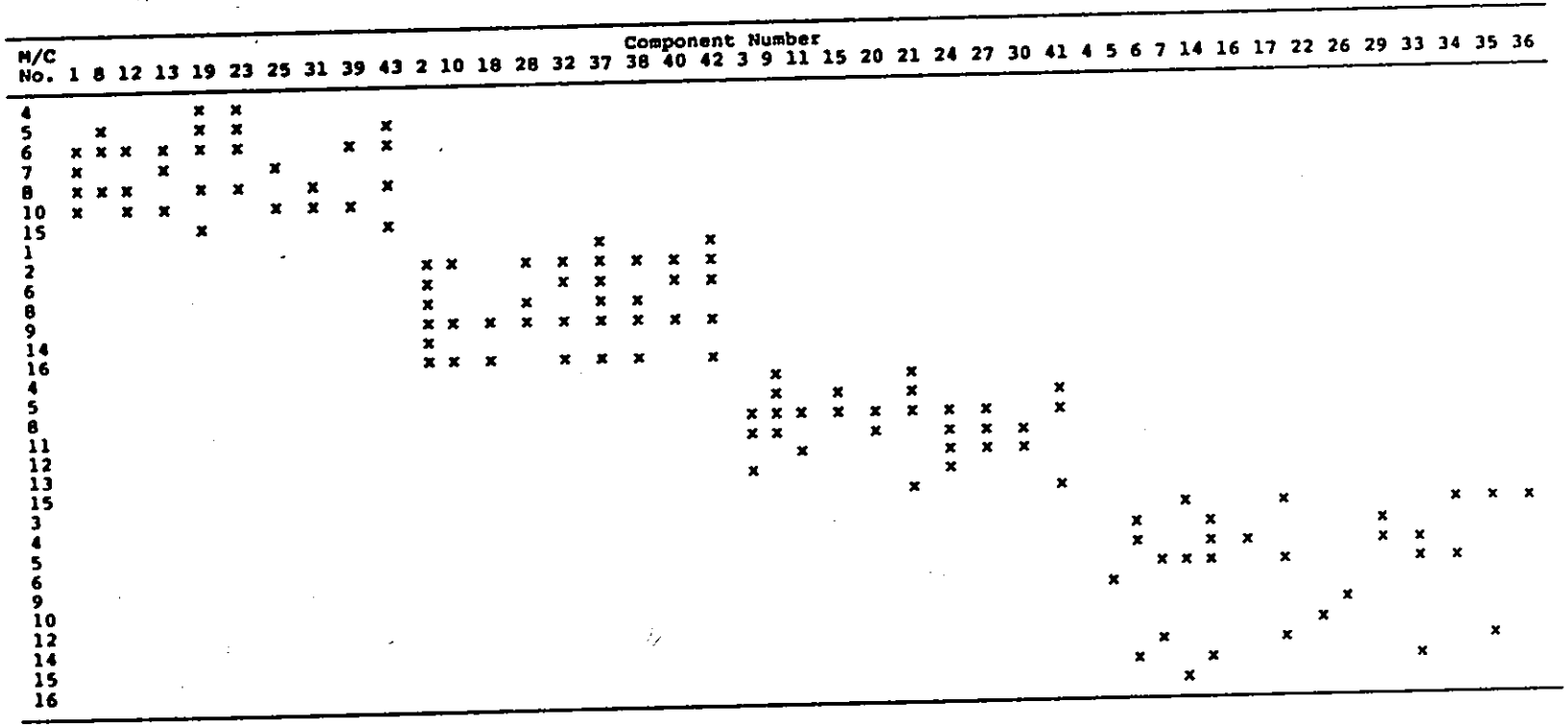


Figure 4.4(b) Solution by K-Means with K = 4

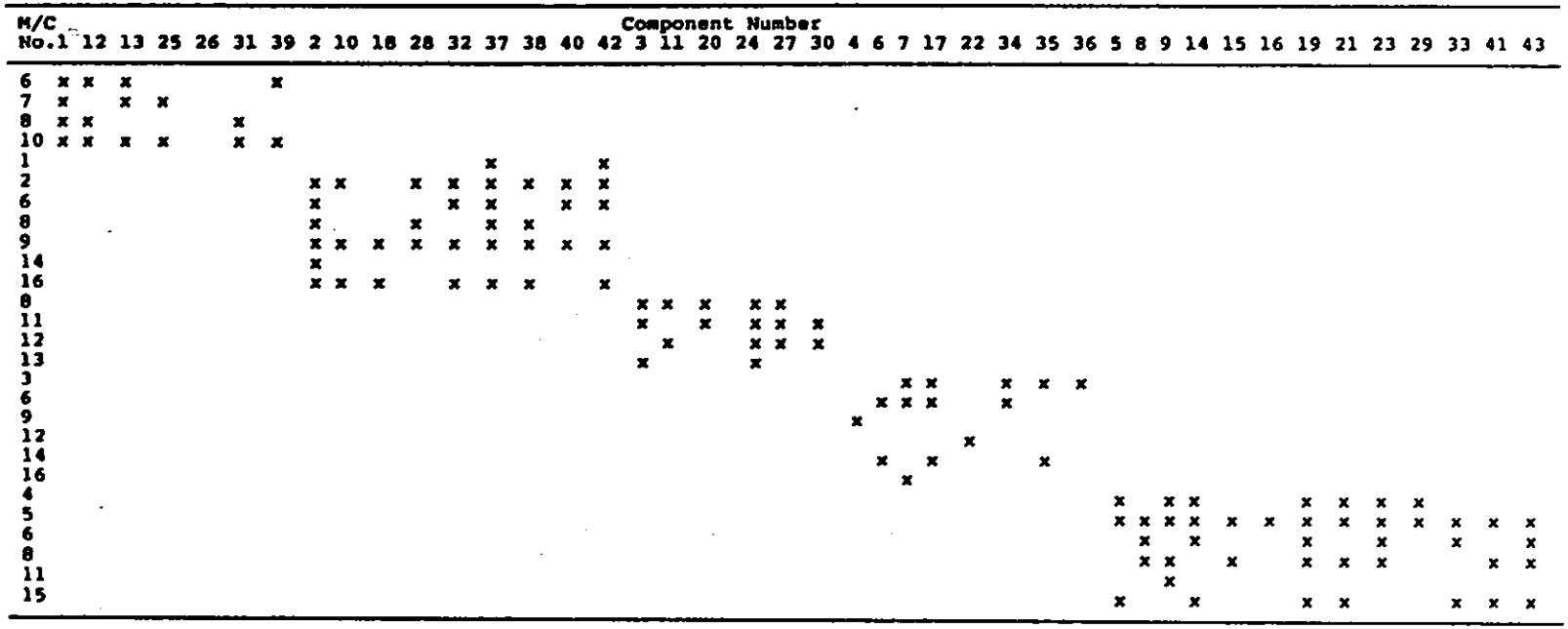


Figure 4.4(c) Solution by K-Means with K = 5

distance algorithm with the pre-determined threshold T. Since subsequent cluster centers are determined by the selection of the initial cluster centre, therefore, the first cluster centre is chosen from the given 43 components randomly. Then, the Revised K-means algorithm applies the maximum distance measure in association with the provided threshold. The results derived for the different thresholds are shown in Figure 4.5. In comparing Figure 4.4 (a) with Figure 4.5 (a), it is found that for the same number of groupings and the extra machines required are less than those proposed by the original K-means solution - 23 machines for Revised K-Means and 26 machines for K-means. Therefore, the machine utilization in these cells should be higher. Both the K-means and the revised K-means algorithms can be used manually.

As discussed before, the Isodata algorithm is manipulated by input parameters which include desired groups, minimum number of members in a group, standard deviation parameter, lumping parameter used as basis for merging two cluster centers during each iteration, and the convergence criteria. These parameters may be chosen by experimental runs if no constraints on the formation of component families exist. Otherwise, the desired number of groups is defined by system configuration and shop floor space constraints. The minimum members in a cell are defined considering the load balancing in the system. Other parameters are also defined based on the actual production situations. It is possible that the results obtained may not be satisfactory. In such cases, adjustment to the above production related parameters can give satisfactory solution. The adjustments are done by using optimization techniques. The model of this problem has been developed (see 4.20 and 4.21 in last section). For the same example, the

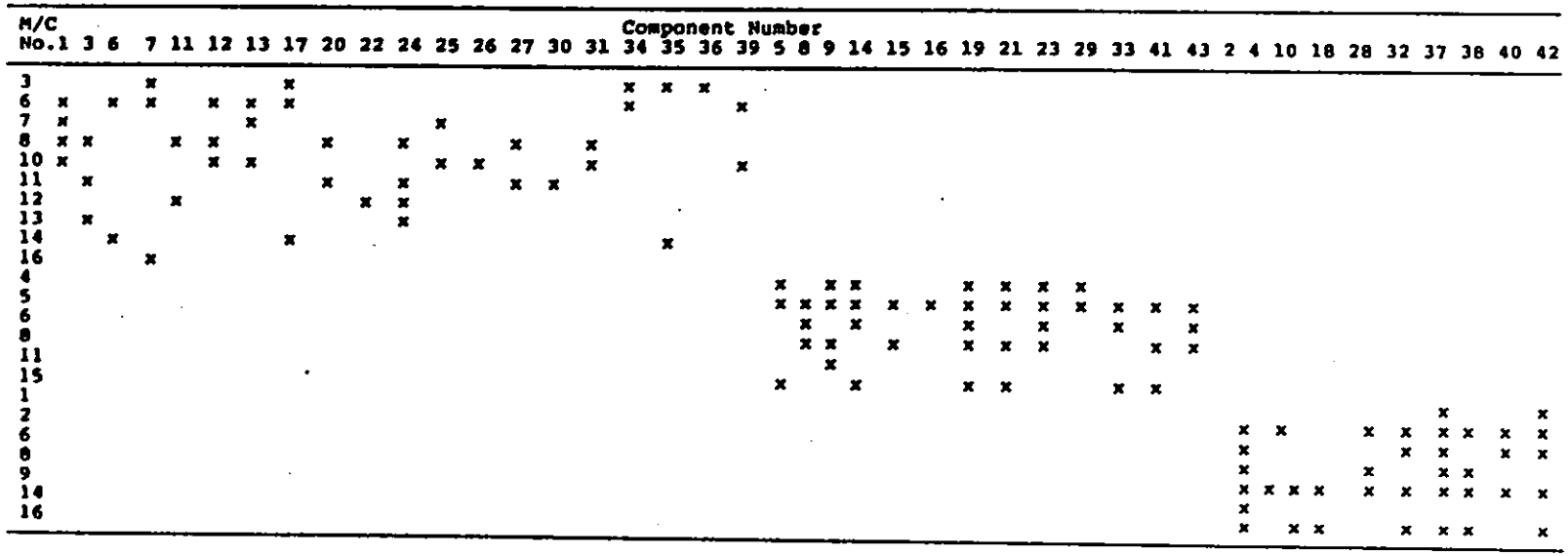


Figure 4.5(a) Solution by Revised K-Means with Threshold T = 7

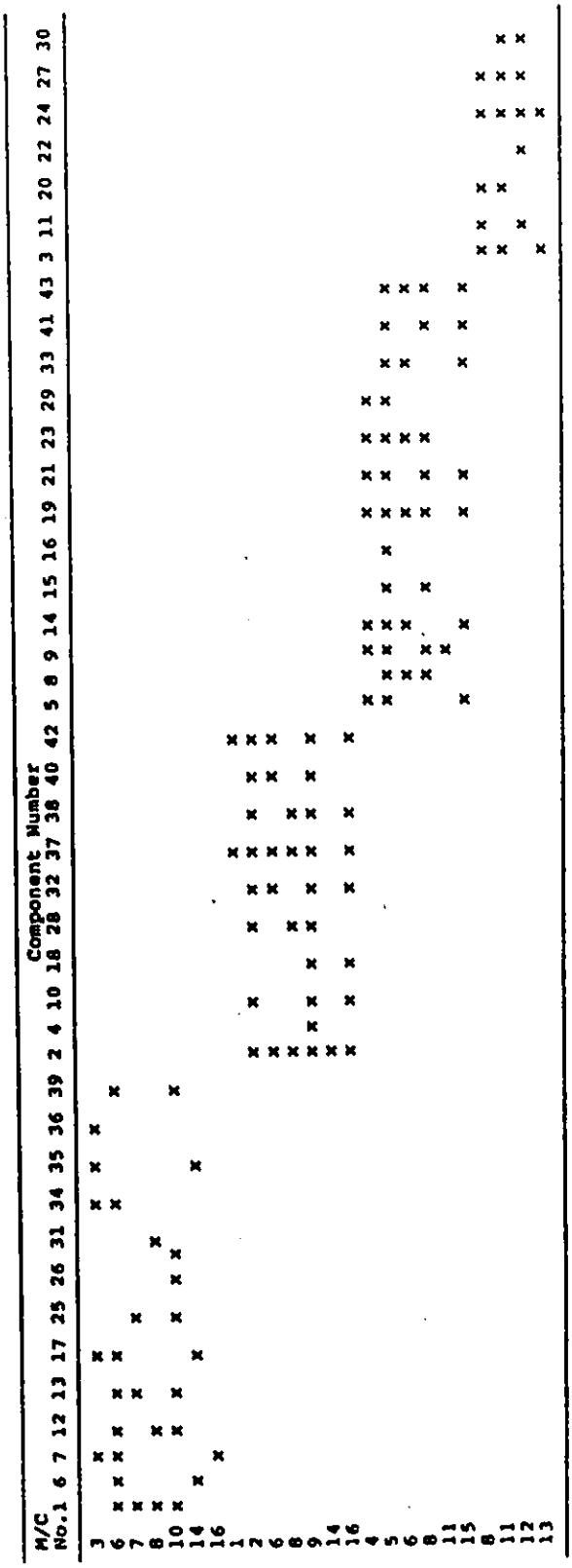


Figure 4.5(b) Solution by Revised K-Means with Threshold T = 6

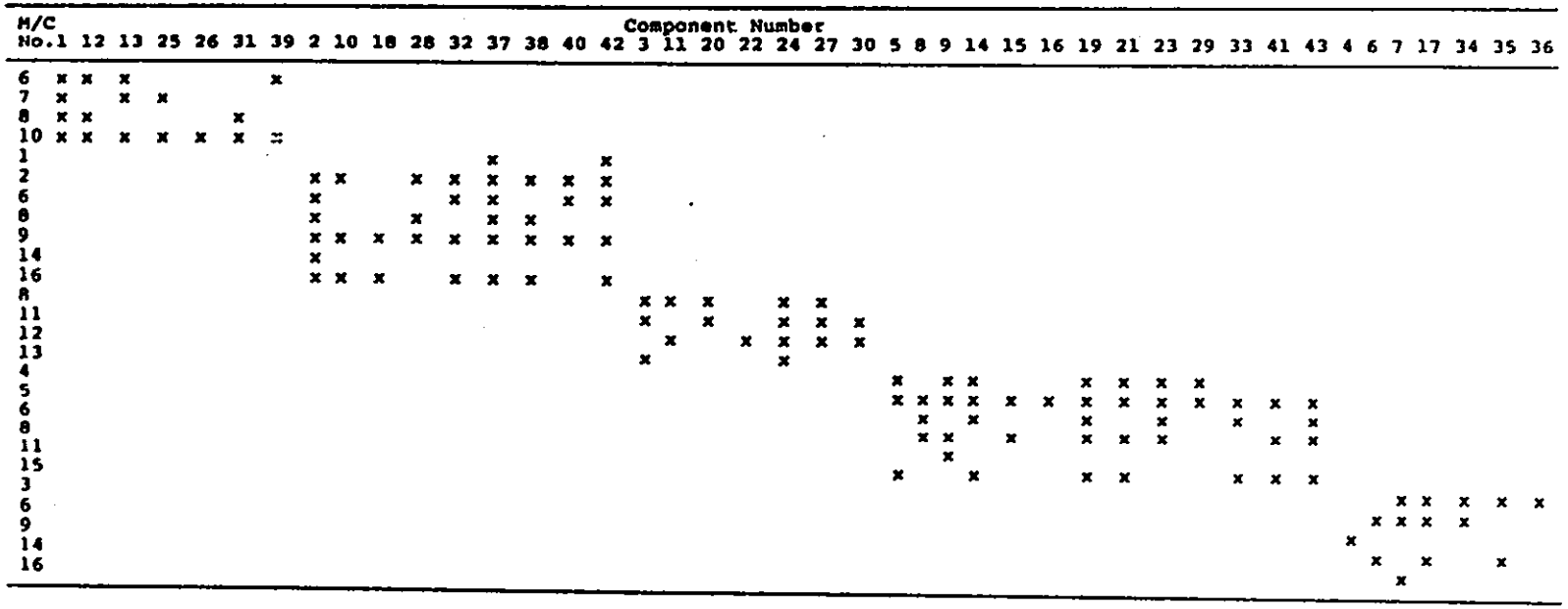


Figure 4.5(c) Solution by Revised K-Means with Threshold T = 5

following formulation has been used:

$$\text{Objective Function:} \qquad U = \sum_{i=1}^{N_c} C_i = \text{minimum} \tag{4.23}$$

$$\begin{aligned}
\text{Constraint Functions:} \qquad & \phi_1 = 6.0 - X(1) \geq 0 \\
& \phi_2 = X(1) - 3.0 \geq 0 \\
& \phi_3 = 8.0 - X(2) \geq 0 \\
& \phi_4 = X(2) - 6.0 \geq 0 \\
& \phi_5 = 4.0 - X(3) \geq 0 \\
& \phi_6 = X(3) - 0.01 \geq 0 \\
& \phi_7 = 3.5 - X(4) \geq 0 \\
& \phi_8 = X(4) - 0.02 \geq 0
\end{aligned} \tag{4.24}$$

Where U is the total machines required to form all cells. C_i is the machines number of ith cell. X(1) is a parameter against which the number of samples in a cluster domain is compared. X(2) is the number of cluster domains desired. X(3) is the standard deviation parameter and X(4) is the lumping parameter. By using the Hooke and Jeeves method (Siddall, 1982), the following solution has been derived:

$$U = 0.250000000E+02$$

$$X(1) = 0.50000000E+01$$

$$X(2) = 0.79400000E+01$$

$$X(3) = 0.50000000E+00$$

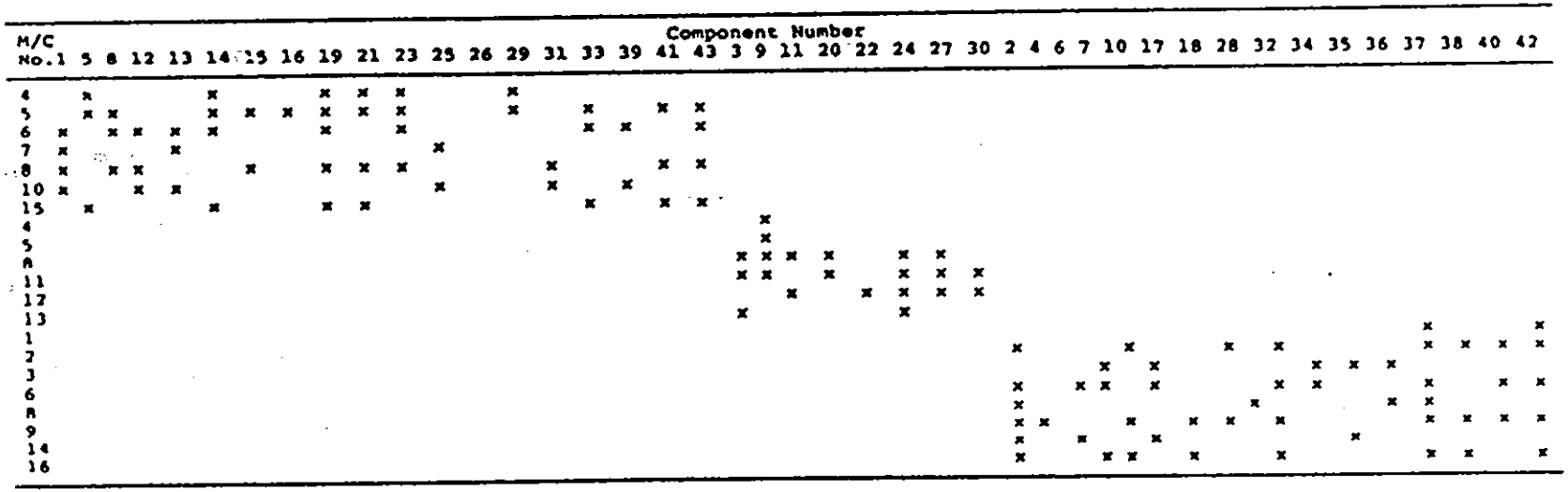
$$X(4) = 0.15000000E+01$$

Bottleneck Machines No.	Extra Machines Required
6	3
8	3
11	1
14	1
16	1

The other solutions produced by the Isodata algorithm with optimization are shown in Figure 4.6.

In this case, the minimum number of components for each group do not strongly control the final grouping results. This range can be specified by looking at the production or by experience. The desired number of cluster centres is a strong constraint and can be specified by multiplying the minimum number of components for each group by the desired number of domains. Also, the range can be determined based on production organization. The standard deviation and lumping parameter can be chosen by experiment runs. In order to control the iteration, the maximum allowed iteration number should be specified. For example, 10 is usually enough for this particular problem.

By comparing these results with the solutions obtained using K-means and revised K-means algorithms, it is found that the Isodata with the optimization algorithm gives better results, since the bottleneck machines required for the same number of cells is less than those required by the other two algorithms. For example, for the same number of cells, Figure 4.4 (a) derived by K-means consists of 26 machines, Figure 4.5 (a) 23 machines and Figure 4.6 (a) 21 machines. The reason is that, during each iteration, the cluster centre is not only updated, but the groups may



Objective Function: $U = \sum_{i=1}^{N_c} C_i = \text{minimum}$

Constraint Functions:

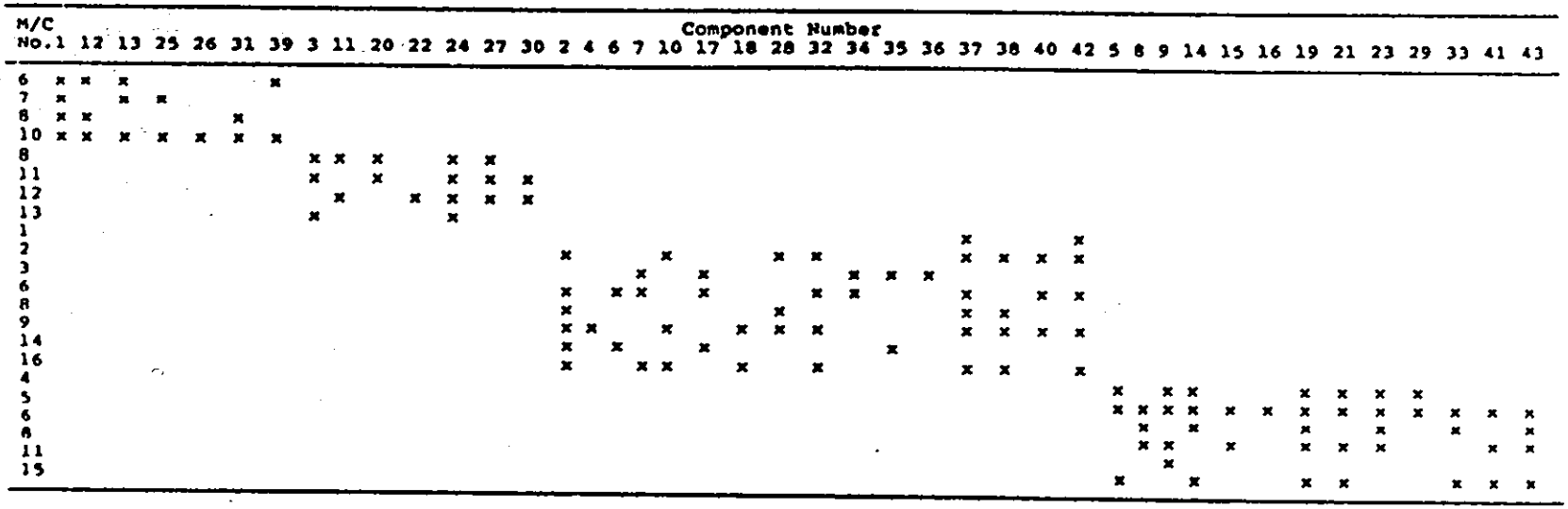
$\phi_1 = (7.0 - X(1)) * 10.0 \geq 0$
$\phi_2 = X(1) - 5.0 \geq 0$
$\phi_3 = 5.0 - X(2) \geq 0$
$\phi_4 = X(2) - 3.0 \geq 0$
$\phi_5 = 4.0 - X(3) \geq 0$
$\phi_6 = X(3) - 0.01 \geq 0$
$\phi_7 = 3.5 - X(4) \geq 0$
$\phi_8 = X(4) - 0.02 \geq 0$

Optimum Solution: U = 0.21000000E+02

X(1) = 0.69699998E+01
X(2) = 0.40000000E+01
X(3) = 0.50000000E+00
X(4) = 0.15000000E+01

Bottleneck Machines No.	Extra Machines Required
8	2
5	1
6	1
4	1

Figure 4.6(a) Solution by ISODATA with Optimization Techniques



Objective Function:

$$U = \sum_{i=1}^{N_c} C_i = \text{minimum}$$

Constraint Functions:

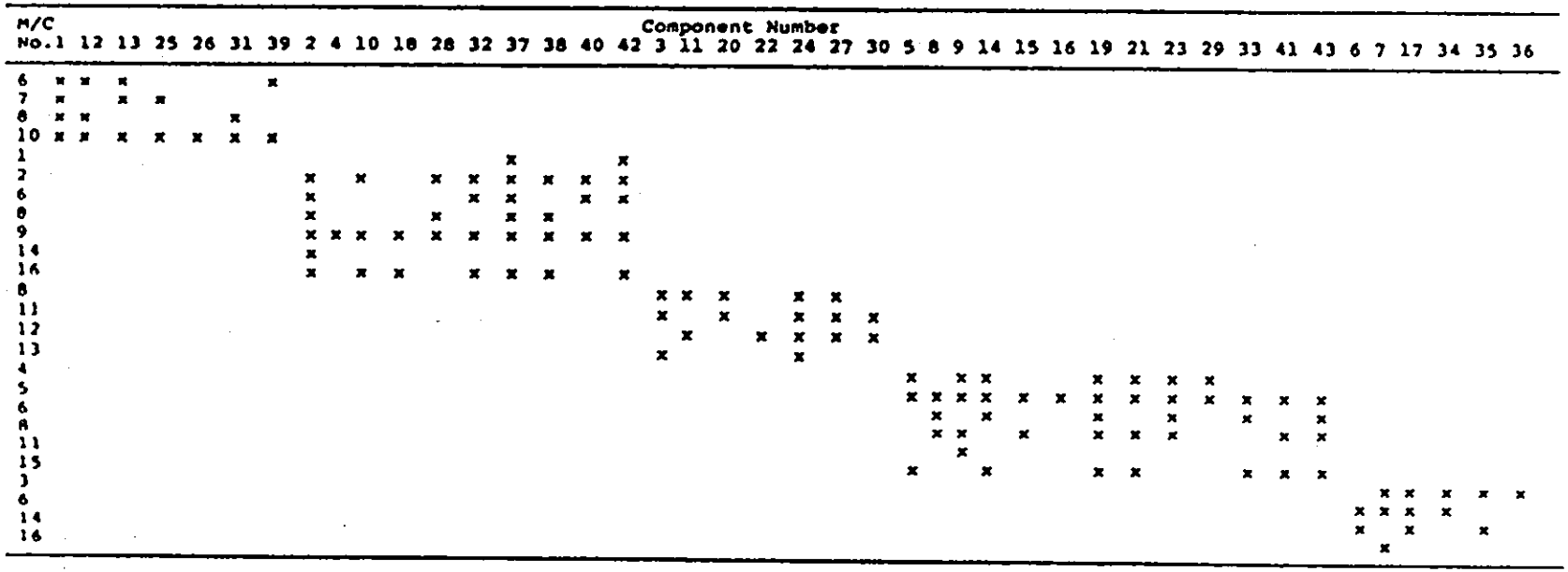
$\phi_1 = 6.0 - X(1) \geq 0$
$\phi_2 = X(1) - 5.0 \geq 0$
$\phi_3 = 6.0 - X(2) \geq 0$
$\phi_4 = X(2) - 5.0 \geq 0$
$\phi_5 = 1.5 - X(3) \geq 0$
$\phi_6 = X(3) - 0.01 \geq 0$
$\phi_7 = 1.5 - X(4) \geq 0$
$\phi_8 = X(4) - 0.02 \geq 0$

Optimum Solution: U = 0.22000000E+02

X(1) = 0.60000000E+01
X(2) = 0.50000000E+01
X(3) = 0.10000000E+01
X(4) = 0.10000000E+01

Bottleneck Machines No.	Extra Machines Required
8	3
6	2
11	1

Figure 4.6(b) Solution by ISODATA with Optimization Techniques



Objective Function:

$$U = \sum_{i=1}^{N_c} C_i = \text{minimum}$$

Constraint Functions:

$\phi_1 = 6.0 - X(1) \geq 0$
$\phi_2 = X(1) - 3.0 \geq 0$
$\phi_3 = 8.0 - X(2) \geq 0$
$\phi_4 = X(2) - 6.0 \geq 0$
$\phi_5 = 4.0 - X(3) \geq 0$
$\phi_6 = X(3) - 0.01 \geq 0$
$\phi_7 = 3.5 - X(4) \geq 0$
$\phi_8 = X(4) - 0.02 \geq 0$

Optimum Solution: U = 0.250000000E+02

X(1) = 0.50000000E+01
X(2) = 0.79400000E+01
X(3) = 0.50000000E+00
X(4) = 0.15000000E+01

Bottleneck Machines No.	Extra Machines Required
6	3
8	3
11	1
14	1
16	1

Figure 4.6(c) Solution by ISODATA with Optimization Techniques

also be discarded, split or merged. This process continues and the number of cluster domains are adjusted at each iteration. The algorithm adjusts the number of groups and members in each group so that the set criterion is satisfied. Also, the optimum combination of these parameters can be determined by including this algorithm within the optimization routines.

2. Comparison of Cluster-Seeking with Other Algorithms

Shown in Figure 4.6 (a) is the solution obtained by the Isodata with optimization algorithm. When compared with the solution obtained by King (1980), Figure 4.9, it is found that if component No. 9 in King's solution is moved to the lower corner group and machine No. 11 is added, the two solutions are identical. Unfortunately, King's algorithm cannot solve the exceptional element problem and, therefore, mutually independent groups cannot be obtained. If the solution in Figure 4.6 (b) is compared with solutions by Burbidge (1971), Figure 4.7, and Chan and Milner (1982), Figure 4.9, it is found that if all three exceptional components are moved to corresponding groups shown in Figure 4.6 (b), and the additional corresponding three machines are added into their cells, the solutions become identical. That is, machine No. 16 is added into the cell with machines No. 3,6 and 14 and machine No. 11 is placed in the cell with machines No. 4,5,6,8 and 15 and machine No. 14 is added into the cell with machines No. 1,2,6,8,9 and 16. These observations are quite interesting. The additional potential advantages of analysis of machine utility in the cells and part assignments to the cells will be discussed later.

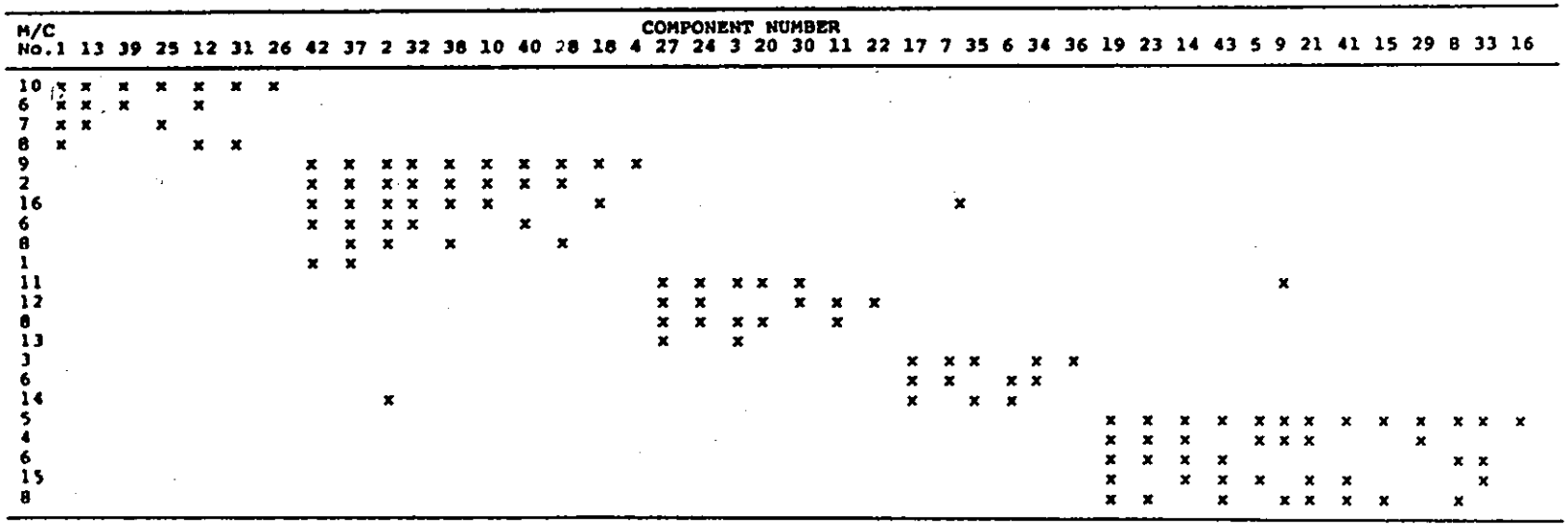


Figure 4.7 Solution by Burbidge (1971)

M/C	COMPONENT NUMBER																																										
No.	1	12	13	39	31	25	26	2	37	42	32	28	38	10	40	18	4	7	6	34	17	35	36	24	3	27	9	20	11	30	22	19	14	21	5	23	29	33	41	43	8	15	16
10	x	x	x	x	x	x	x																																				
6	x	x	x	x																																							
8	x	x			x																																						
7	x		x			x																																					
9								x	x	x	x	x	x	x	x	x	x																										
2								x	x	x	x	x	x	x	x																												
6								x	x	x	x	x						x	x	x																							
16								x	x	x	x		x	x		x		x																									
8								x	x			x	x																														
14								x											x		x	x																					
1									x	x																																	
3																		x		x	x	x	x																				
8																								x	x	x	x	x	x														
11																								x	x	x	x	x		x													
13																								x	x																		
12																								x		x			x	x	x												
5																											x					x	x	x	x	x	x	x	x	x	x	x	x
4																											x					x	x	x	x	x	x						
15																																x	x	x	x			x	x	x			
6																																x	x			x		x			x		
8																																x		x		x			x	x	x	x	

Figure 4.8 Solution by King (1980)

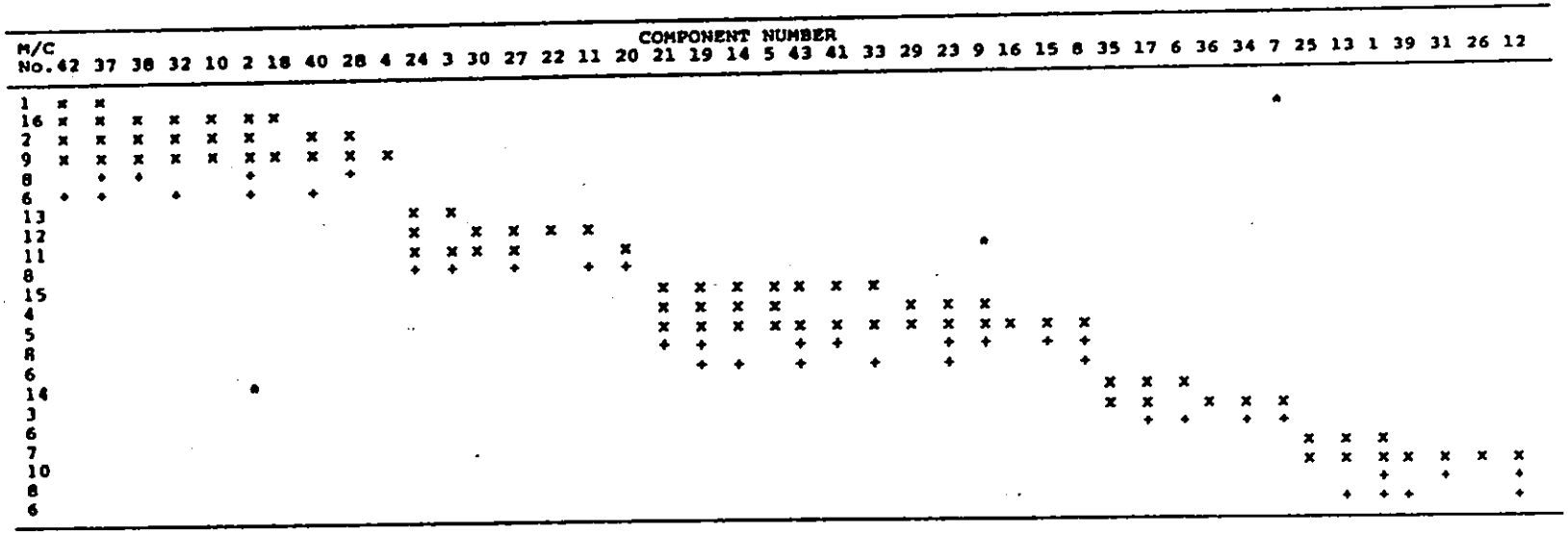


Figure 4.9 Solution by Chan and Milner (1982)

4.3 Feature-Based Expert Assignment of Parts to Machine Cells

4.3.1 Introduction

Manufacturing cells and part families can be designed using the cluster-seeking approaches as, discussed previously. Once manufacturing cells have been formed, it is necessary to develop methods and algorithms to support the on-going decision-making process regarding production planning and parts assignment to the existing cells. It is best to establish a link to integrating the design and manufacturing cells and directly assign new parts to the appropriate existing manufacturing cell (ElMaraghy and Gu, 1988). A feature-based assignment system is developed as the linkage for the feature-based design and the formed cells. The component design can be done through either the FDDL or the feature-based modeler. A full description of the designed component is included in the source language. The assignment system needs only certain parts of the source language in a specific format, defined as the target language.

4.3.2 Code Generator for Parts Assignment

Before the syntax for the target language is defined, the required information should be decided upon for the assignment system. As discussed in Chapter 2, the information for part assignment includes material condition, feature type, the main dimensions of the feature geometry, tolerances and surface finish. The syntax of the target language has been defined for the part assignment system and is included in Appendix A.

The assignment is performed for each individual part, that is one part at a time. Thus, the code generator will create a file containing the

target language for every part in the product. The name of each file is derived from the part identifier. This is further explained as follows:

The code generator first expects an occurrence of "part". This tells it that a part is being defined. For each occurrence of "part", it finds the identifier by searching for the "name" keyword within the parentheses. The identifier used is found within parentheses following "name". Then a file is created for this part.

For each occurrence of "feature", a keyword is searched. These keywords are classified as follows:

"name"

"type"

"location"

"tolerance"

"surface_finish"

"relation"

If one of the above classes is recognized, the expected information is found and verified.

name: Determine an identifier to be placed in the symbol table. Check that this identifier is unique. If the particular identifier has been used before, then an error message is printed out.

type: A type keyword is expected. Classify this keyword and expect associated parameters. Since the features have different geometric shapes and characteristics, for example, following "cylinder_front" one diameter is expected; for "cylinder_side", two parameters are anticipated - diameter and length.

location: Six parameters are expected. These are the reference point

coordinates and three direction angles or parameters, e1,e2,e3.

tolerance: A tolerance keyword is expected as a class of all types of defined tolerances. If one of these tolerances is matched, the relevant parameters and possible datums are searched.

surface-finish: A value is expected,

relation: Search a keyword as one of the defined relations, then a expected part identifier is sought.

The code generator processes one feature and prints it until all features are processed. Some information, such as location and orientation, are not printed. Only the checking is performed in these instances.

4.3.3 System Structure

The feature-based assignment system structure is shown in Figure 4.10. The system consists of an analyzer, a synthesizer, and a recognizer. This system is integrated with the FDDL system and the feature-based modeler and has been developed in Micro Vax II and in Fortran language.

4.3.4 Analyzer

After a part is modeled, the corresponding source language file is generated. Using the code generator, the target language is created for the use of the assignment system. The target language contains the features, main dimensions, tolerances and surface-finish. These form the basis for deciding the required machining operations and sequence. As mentioned before, these features with their attributes are related to specific machining operations. The analyzer expert module recognizes these relationships using built in production rules. For example:

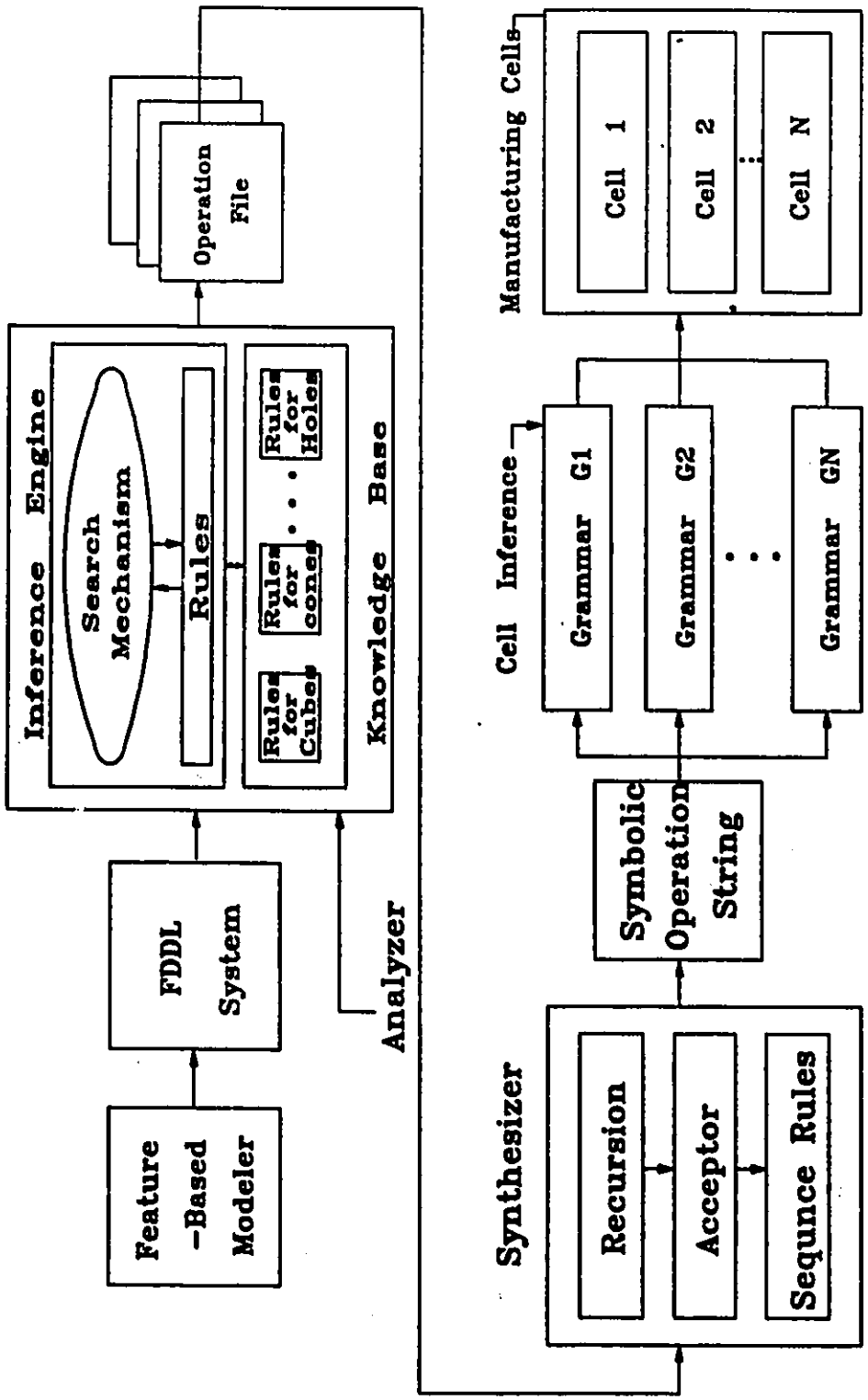


Figure 4.10 Feature-Based Assignment System

IF the feature is a cylinder and;
its diameter is larger than or
equal to D1 and smaller than D2 and;
its total tolerance is smaller than or
equal to T1 and; blank material is bar;
THEN Turning + Grinding or Turning + Diamond
Turning are recommended.

IF the feature is a hole and;
its diameter is larger than or
equal to D1 and smaller than D2 and;
its total tolerance is smaller
than TH1 and; the part
is a rotational part;
THEN Drilling + Boring + Grinding
or Drilling + Boring + Diamond Boring
or Drilling + Boring + Broaching are
recommended.

IF the feature is a spline hole and
through a whole bore;
THEN Drilling + Boring + Broaching are recommended.

IF the feature is a cylinder and;
its surface-finish height is
lower than 16.0 micro inches
and the part is a rotational one;
THEN Turning + Cylindrical Grinding

are recommended.

All features and their possible machining operations are defined in a similar manner and, therefore, all necessary manufacturing operations for the design features can be found by using this module.

4.3.5 Synthesizer

From the nature of the feature-based design of components and the part geometric structure, a problem arises in the duplication of operations inference. For example, Figure 3.8 shows two cylinders to be fitted into the same kind of bearing. They require exactly the same machining operations and sequences. In such cases, the expert module deletes all duplications by using a set operation UNION. If all operations are treated in this way, all duplicate operations can be deleted and only one of each type will remain. Another potential problem is the individual operations duplications. This problem is handled by following the Finite-State Automata technique.

The output from this UNION operation is a string which represents the operations for the part. However, a feature may be produced by different operation combinations, which means that the part may be manufactured in different ways. These alternatives are provided essentially for production planning and scheduling purposes to balance the load of machines or cells. Difficulties arise from the requirements of the recursion operations since it is impossible to predict the number of operation combinations. Therefore, a sophisticated technique and data structuring are truly required to deal with this problem.

Input into this process is a string of characters which represent the machining operations. To determine all of the machining combinations

the string must be decomposed into a number of new strings, each of them representing a sequence of operations required to produce the modeled part. This decomposition process continues until all possibilities are derived. This is similar to creating a tree structure, starting with the original string as level one, and the first choice of the first feature machining operations as level two. In this way, options are continuously decomposed into leaves. An example is shown in Figure 4.11.

All strings created by this process represent possible procedures for producing the part, but it is possible for an individual operation to be duplicated, as previously mentioned. In order to examine these strings, a Finite-State Acceptor is designed to accept the strings from the above process and create corresponding new strings by filtering out all duplicate operations.

The Finite-State Automata is based on the finite-state automaton and is usually represented as Fu (1982) and Hopkin and Moss (1976):

$$A = (\Sigma, Q, \delta, q, F) \tag{4.25}$$

where $\Sigma = \{s1, s2,..., sn\}$ is a finite set of input symbols, $Q = \{q1, q2,..., qm\}$ is a finite set of states, $\delta(qi, si)$ is a mapping $Q \times \Sigma$ into Q, $q \in Q$ is the initial state, and $F \in Q$ is the set of final states.

The Finite-State Acceptor is designed to test if an input string belongs to one of the final states, which are considered as possible strings or combinations of the machining operations. This acceptor works by examining the leftmost symbol of a string in Σ. Assuming that the current state is q, and reading the input symbol, the automaton A transit to state of q', which can be expressed as :

$$\delta(q, x) = q' \qquad q, q' \in Q \; x \in \Sigma \tag{4.26}$$

This mapping is usually represented by a state transition diagram, shown in Figure 4.12. The algorithm is exactly the same as that described in this figure. Thus, all strings are examined by the acceptor and the corresponding new strings are created for each machining sequence arrangement and cells assignment used.

The new strings from the acceptor are tested again to see whether their sequences are logically correct from a manufacturing point of view. If a string contains symbols in the wrong order, they are again rearranged. These logical rules are set up based on the knowledge of the manufacturing process. Knowledge rules have been built in to handle various parts which have different geometric and technical characteristics. Fundamentally, the rough machining should be followed by the semi-finishing machining which is then followed by the finish machining. Some operations are actually more flexible if they belong to the same group of operations from the rough, semi-finishing and finishing operations' point of view. For example, the order of the rough turning of a cylinder or the drilling of a hole may be interchanged without causing problems. More elaborate rules are required for the concerns required for the purpose of process planning and machine load balancing. In this system, these problems are solved in a general way whereby only the machining logic is taken into consideration. This part of the module can order operations by considering the existing machine tools in the cells. Therefore, the output of this module is the strings of operations for cells inference, i.e. assignment of a part to a cell.

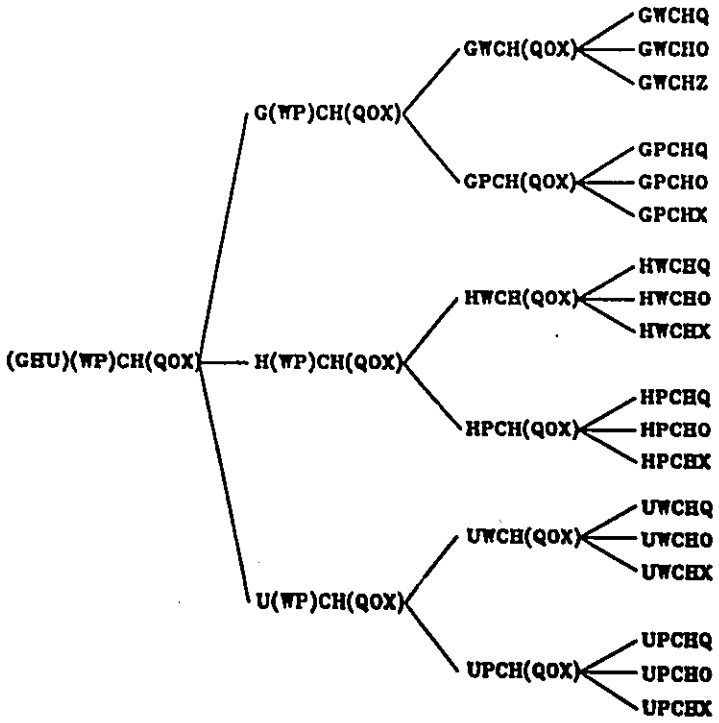


Figure 4.11 Decomposition Process of Machining Combinations

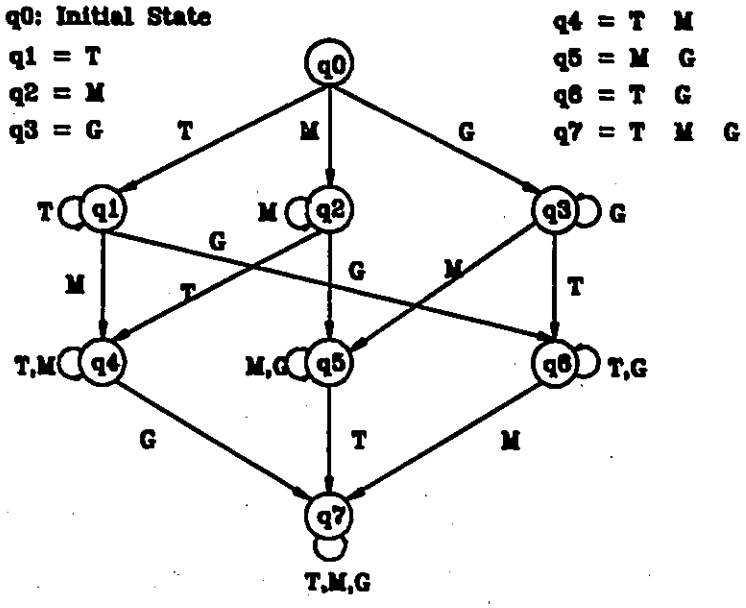


Figure 4.12 Finite-State Acceptor

4.3.6 Part Assignment to Manufacturing Cells

The manufacturing cells have been already formed using the Isodata approach combining with optimization techniques. The chosen configuration is shown in Figure 4.6(c). Each cell contains several machine tools and each machine, in turn, can be represented as a symbol associated with operations. The problem now is how to assign the parts to the existing cells. The operations required to machine the modelled parts are derived using the previous modules. Matching the part with the existing cells can be done by using the syntactic pattern recognition technique (Fu, 1982).

The syntactic pattern recognition approach is analogous to natural languages. A pattern is represented as a collection of primitives just as a sentence is composed of a group of words. Usually, a 4-tuple is defined as a grammar G :

$$G = (Vt, Vn, P, S) \tag{4.27}$$

where Vt is a set of terminal symbols, Vn is a set of non-terminal symbols, P is a set of production rules or rewrite rules, and S is the start symbol. Since the cells have been formed, pattern recognition (i.e. the assignment of part to cells) can be done. However, the production rules are not only based on the cells configuration, but also on the material flow pattern in the cells. The material flow pattern is defined based on real time status of the machines in the flexible manufacturing cells. Flexible manufacturing, in general, allows random parts routing. Therefore, the parts may access the machine tools in any order, to maximize cell efficiency and production scheduling. If finite-state grammar

is used, the production rules would have the following form:

G = (Vt, Vn, P, S)

Vt = {drilling, boring, broaching}

Vn = {S, A, B} (4.28)

Production P :

S--> drillingA, A --> boringB, B --> broaching

By using these production rules for each individual cell, a new part can be assigned to its related cells by matching the parts machining requirements and the cell production capabilities. A problem may arise in which a part cannot be produced totally in one cell. Error transformations can be used here to find the closest cell which would satisfy the machining requirements.

Error transformations consist of three types (Fu, 1982):

XaY = T(XY)	deletion	
XcY = T(XaY)	substitution	(4.29)
XY = T(XdY)	insertion	

Since the material flow is assumed to be random, the transformations of computation can be realized as follows: 1) by matching the identical characters from the input part to each cell, and 2) by substituting the cell's machines by parts' operations. If the operations for the part are more than those in the cell, insertion is required (additional operations). These operations for producing the part can have different weights according to the importance of the machine tools in the cell. For real situations, other treatments may be required for the optimum assignments of parts. For example, if an operation required for a part does not exist in a cell and a suitable substitute cannot be found, a high penalty can be set.

If replacement machines can be found in the cell, then a smaller penalty is used.

4.3.7 A Case Study

Examples of machine tools and components for forming the cells were provided by Burbidge. The cells are formed and assigned as different machines, the machines and the formed cells are shown in Table 4.1 and 4.2. Three parts are used for the case study. One shaft and one box are modeled using the feature-based modeler. Another part is input directly by the user following the syntax of the FDDL.

The shaft model is shown in Figure 3.8 and the associated detailed descriptions of the part are shown in Figure 3.9. After the code generator processes the output (the source language), the corresponding target language created is shown in Figure 4.13. The output from the analyzer of the assignment system is given in Table 4.3. It contains operations for the production of each feature included in the part model. The letters in the brackets represent a group of machining operations which will be converted by the synthesizer. The number of letters in these brackets correspond to the number of operation combinations. It is clear that duplications exist (see A, (WY) in Table 3). After the UNION operation, the data file contains only the essential operations required to produce the part features as shown in Table 4.4. The operation combinations for machining the parts are derived by the recursion module. The results indicate that there are four alternate paths available to produce this part. After the combinations are entered, the acceptor creates the corresponding new strings without duplicating any of the individual operations. These strings are examined by the manufacturing logic module (see Table 4.5).

Finally, these four alternatives are inferred with the cells configuration represented in the strings. Since flexible material flow is assumed, the cell inference is actually carried out by distance computations without parsing. Unit weight, which means that all machines in the cells are equally important, is used in this example for cell inference and the results are shown in Table 4.6. For different weights assigned to the same part, the expert assignment and cell inference results are shown in Table 4.7. Therefore, the part should be assigned to cell No. 4 if it is produced with operations AFH, or cell No. 1 if it is machined with AHM.

The same procedures are applied to the part shown in Figure 3.10. The output (the source language) from the FBM is given in Figure 3.11. The target language created by the code generator refers to Figure 4.14. The outputs from the analyzer, UNION operation and synthesizer are shown in Table 4.8, Table 4.9 and Table 4.10 respectively. The final assignments can be determined based on the results in Tables 4.11 and 4.12. Thus, if the part is produced in the DEFGI, it should be sent to cell No. 4.

The last example is shown in Figure 2.3. Its input language description is given in Figure 2.4. The code after being processed by the code generator is in Figure 4.15. The outputs from the analyzer, UNION operation and synthesizer are shown in Table 4.13, Table 4.14 and Table 4.15 respectively. The inference results are shown in the Table 4.16 and Table 4.17. Clearly, if this part is produced using operations ADEFGH, it should be machined in cell No. 4.

Table 4.1 Machine Tools

Symbol	Machine Tools
A	Lathe
B	Shaper1
C	Shaper2
D	Drilling Machine
E	Boring Machine
F	Vertical Mill
G	Internal Grinding
H	Cylindrical Grind.
I	Surface Grinding
J	Diamond Boring
K	Broaching
L	Diamond Lathe
M	Universal Mill
N	Hobbing
O	Lapping
P	Tool Grinding

Table 4.2 Formed Cells

Cell No.	Machine Tools
1	AFHM
2	ACDIJP
3	DJKO
4	ADEFGHN
5	ABDL

```
cylinder_front
5.40 0.0
surface_finish
100.0
chamfer
5.40 0.30
surface_finish
100.0
thread_nc_side
6.0 4.0
surface_finish
100.0
cylinder_front
8.0 0.0
distance
-2.50 2.0
surface_finish
100.0
cylinder_side
8.0 3.0
diameter
2.60 1.40
surface_finish
30.0
cylinder_front
13.0 0.0
perpendicularity
0.80
surface_finish
50.0
cylinder_side
13.0 2.0
surface_finish
100.0
cylinder_front
9.0 0.0
distance
-1.50 2.0
parallelism
1.0
surface_finish
50.0
cylinder_side
9.0 17.0
diameter
0.60 -0.60
concentricity
1.0
surface_finish
30.0
cylinder_front
9.0 0.0
distance
-1.50 1.0
perpendicularity
1.0
surface_finish
50.0
cylinder_side
8.0 3.0
diameter
2.60 1.40
concentricity
0.80
surface_finish
30.0
cylinder_front
8.0 0.0
distance
-1.50 1.0
surface_finish
100.0
thread_nc_side
6.0 4.0
surface_finish
100.0
chamfer
6.0 0.30
surface_finish
100.0
cylinder_front
5.40 0.0
surface_finish
100.0
p_keyseat_side
0.90 10.0
distance
-0.50 0.50
surface_finish
50.0
p_keyseat_side
0.90 10.0
surface_finish
50.0
p_keyseat_corner
1.1250 0.90
surface_finish
50.0
p_keyseat_corner
1.1250 0.90
surface_finish
50.0
stop
```

Figure 4.13 Target Language of Modeled Shaft in Figure 3.8 for the Assignment System

Table 4.3 Output from analyzer module

No.	Feature	Operation
1	cylinder_front	A
2	chamfer	A
3	thread_nc_side	A
4	cylinder_front	A
5	cylinder_side	(WY)
6	cylinder_front	A
7	cylinder_side	A
8	cylinder_front	A
9	cylinder_side	(WY)
10	cylinder_front	A
11	cylinder_side	(WY)
12	cylinder_front	A
13	thread_nc_side	A
14	chamfer	A
15	cylinder_front	A
16	p_keyseat_side	(FM)
17	p_keyseat_side	(FM)
18	p_keyseat_corner	(FM)
19	p_keyseat_corner	(FM)

W=AH, Y=AL

Table 4.4 Operations after UNION

Length of String	String
1	A
4	(WY)
4	(FM)

Table 4.5 Output from the Synthesizer

Alternative	String
1	AFH
2	AHM
3	AFL
4	ALM

Table 4.6 Cells Inference Results

Operations for the Part :			AFH
Cell :	AEHM	Distance :	1
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	3
Cell :	ADEFGHN	Distance :	0
Cell :	ABDL	Distance :	2
Operations for the Part :			AHM
Cell :	AEHM	Distance :	0
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	3
Cell :	ADEFGHN	Distance :	1
Cell :	ABDL	Distance :	2
Operations for the Part :			AFL
Cell :	AEHM	Distance :	2
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	3
Cell :	ADEFGHN	Distance :	1
Cell :	ABDL	Distance :	1
Operations for the Part :			ALM
Cell :	AEHM	Distance :	1
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	3
Cell :	ADEFGHN	Distance :	2
Cell :	ABDL	Distance :	1

Weight = 1

Table 4.7 Cells Inference Results

Operations for the Part :			AFH
Cell :	AEHM	Distance :	4
Cell :	ACDIJP	Distance :	8
Cell :	DJKO	Distance :	10
Cell :	ADEFGHN	Distance :	0
Cell :	ABDL	Distance :	8
Operations for the Part :			AHM
Cell :	AEHM	Distance :	0
Cell :	ACDIJP	Distance :	5
Cell :	DJKO	Distance :	7
Cell :	ADEFGHN	Distance :	1
Cell :	ABDL	Distance :	5
Operations for the Part :			AFL
Cell :	AEHM	Distance :	5
Cell :	ACDIJP	Distance :	5
Cell :	DJKO	Distance :	7
Cell :	ADEFGHN	Distance :	1
Cell :	ABDL	Distance :	4
Operations for the Part :			ALM
Cell :	AEHM	Distance :	1
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	4
Cell :	ADEFGHN	Distance :	2
Cell :	ABDL	Distance :	1

Weight: A-D W=2, E-K W=4, L-P W=1

```
bore
2.40 5.0
diameter
1.80 0.0
parallelism
1.60
distance
1.50 -1.50
distance
1.50 -1.50
surface_finish
40.0
box_face
20.0 15.0
flatness
0.60
surface_finish
30.0
box_face
20.0 12.0
distance
1.20 -0.50
parallelism
1.0
surface_finish
50.0
box_face
20.0 12.0
perpendicularity
0.80
surface_finish
50.0
box_face
12.0 15.0
perpendicularity
1.50
surface_finish
100.0
box_face
12.0 15.0
surface_finish
100.0
box_face
20.0 15.0
parallelism
1.0
surface_finish
50.0
bore
3.60 8.0
diameter
1.40 0.0
distance
1.50 -1.50
distance
1.50 -1.50
surface_finish
40.0
bore
4.80 4.0
diameter
1.60 0.0
distance
2.50 1.0
concentricity
1.0
surface_finish
40.0
stop
```

Figure 4.14 Target Language of Modeled Part in Figure 3.10 for the Assignment System

Table 4.8 Output from analyzer module

No.	Feature	Operation
1	bore	(ZR)
2	box_face	I
3	box_face	I
4	box_face	I
5	box_face	I
6	box_face	(BF)
7	box_face	I
8	bore	(ZR)
9	bore	(ZR)

Z=DEJ(for nonrotational part)
Z=DAJ(for rotational part)
R=DEG(for nonrotational part)
R=DAG(for rotational part)

Table 4.9 Operations after UNION

Length of String	String
4	(ZR)
1	I
4	(BF)

Table 4.10 Output from the Synthesizer

Alternative	String
1	BDEIJ
2	DEFIJ
3	BDEGI
4	DEFGI

Table 4.11 Cells Inference Results

Operations for the Part :			BDEIJ
Cell :	AEHM	Distance :	4
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	3
Cell :	ADEFGHN	Distance :	3
Cell :	ABDL	Distance :	3
Operations for the Part :			DEFIJ
Cell :	AEHM	Distance :	4
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	3
Cell :	ADEFGHN	Distance :	2
Cell :	ABDL	Distance :	4
Operations for the Part :			BDEGI
Cell :	AEHM	Distance :	4
Cell :	ACDIJP	Distance :	3
Cell :	DJKO	Distance :	4
Cell :	ADEFGHN	Distance :	2
Cell :	ABDL	Distance :	3
Operations for the Part :			DEFGI
Cell :	AEHM	Distance :	4
Cell :	ACDIJP	Distance :	3
Cell :	DJKO	Distance :	4
Cell :	ADEFGHN	Distance :	1
Cell :	ABDL	Distance :	4

Weight = 1

Table 4.12 Cells Inference Results

Operations for the Part :			BDEIJ
Cell :	AEHM	Distance :	12
Cell :	ACDIJP	Distance :	6
Cell :	DJKO	Distance :	10
Cell :	ADEFGHN	Distance :	10
Cell :	ABDL	Distance :	12
Operations for the Part :			DEFIJ
Cell :	AEHM	Distance :	14
Cell :	ACDIJP	Distance :	8
Cell :	DJKO	Distance :	12
Cell :	ADEFGHN	Distance :	8
Cell :	ABDL	Distance :	16
Operations for the Part :			BDEGI
Cell :	AEHM	Distance :	12
Cell :	ACDIJP	Distance :	10
Cell :	DJKO	Distance :	14
Cell :	ADEFGHN	Distance :	6
Cell :	ABDL	Distance :	12
Operations for the Part :			DEFGI
Cell :	AEHM	Distance :	14
Cell :	ACDIJP	Distance :	12
Cell :	DJKO	Distance :	16
Cell :	ADEFGHN	Distance :	4
Cell :	ABDL	Distance :	16

Weight: A-D W=2, E-K W=4, L-P W=1

```
thread_nf_front
70.0 0.0
surface_finish
2.5
bore
50.0 25.0
diameter
-0.02 0.01
concentricity
0.04
surface_finish
1.5
bore
40.0 150.0
diameter
-0.02 0.01
cylindricity
0.04
surface_finish
1.5
thread_nf_side
70.0 20.0
surface_finish
2.5
cylinder_front
80.0 0.0
distance
-0.025 0.025
surface_finish
1.8
cylinder_side
80.0 25.0
diameter
-0.02 0.01
concentricity
0.04
surface_finish
1.5
t_cone_front
90.0 0.0
perpendicularity
0.025
surface_finish
1.8
t_cone_side
110.0 30.0
angularity
0.04
surface_finish
1.5
cylinder_side
110.0 10.0
surface_finish
3.0
cylinder_front
140.0 0.0
surface_finish
2.0
cylinder_side
140.0 30.0
surface_finish
3.0
cylinder_front
140.0 0.0
perpendicularity
0.05
surface_finish
1.5
cylinder_side
100.0 40.0
diameter
-0.02 0.01
concentricity
0.04
surface_finish
1.5
cylinder_front
100.0 0.0
distance
-0.025 0.025
surface_finish
1.5
thread_nf_side
70.0 20.0
surface_finish
2.5
thread_nf_front
70.0 0.0
surface_finish
2.5
p_keyseat_side
15.0 10.0
surface_finish
2.0
p_keyseat_side
15.0 10.0
surface_finish
2.0
p_keyseat_corner
6.0 10.0
surface_finish
2.0
p_keyseat_corner
6.0 10.0
surface_finish
2.0
stop
```

Figure 4.15 Target Language of Modeled Part in Figure 2.3 for the Assignment System

Table 4.13 Output from analyzer module

No.	Feature	Operation
1	thread_nc_front	A
2	bore	(ZR)
3	bore	(ZR)
4	thread_nc_side	A
5	cylinder_front	A
6	cylinder_side	(WY)
7	t_cone_front	A
8	t_cone_side	A
9	cylinder_side	A
10	cylinder_front	A
11	cylinder_side	A
12	cylinder_front	A
13	cylinder_side	(WY)
14	cylinder_front	A
15	thread_nc_side	A
16	thread_nc_front	A
17	p_keyseat_side	(FM)
18	p_keyseat_side	(FM)
19	p_keyseat_corner	(FM)
20	p_keyseat_corner	(FM)

Table 4.14 Operations after UNION

Length of String	String
1	A
4	(ZR)
4	(WY)
4	(FM)

Table 4.15 Output from the Synthesizer

Alternative	String
1	ADFHJ
2	ADHJM
3	ADFJL
4	ADJLM
5	ADFGH
6	ADGHM
7	ADFGL
8	ADGLM

Table 4.16 Cells Inference Results

Operations for the Part :			ADFHJ
Cell :	AEHM	Distance :	3
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	3
Cell :	ADEFGHN	Distance :	1
Cell :	ABDL	Distance :	3
Operations for the Part :			ADHJM
Cell :	AEHM	Distance :	2
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	3
Cell :	ADEFGHN	Distance :	2
Cell :	ABDL	Distance :	3
Operations for the Part :			ADFJL
Cell :	AEHM	Distance :	4
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	3
Cell :	ADEFGHN	Distance :	2
Cell :	ABDL	Distance :	2
Operations for the Part :			ADJLM
Cell :	AEHM	Distance :	3
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	3
Cell :	ADEFGHN	Distance :	3
Cell :	ABDL	Distance :	2
Operations for the Part :			ADFGH
Cell :	AEHM	Distance :	3
Cell :	ACDIJP	Distance :	3
Cell :	DJKO	Distance :	4
Cell :	ADEFGHN	Distance :	0
Cell :	ABDL	Distance :	3
Operations for the Part :			ADGHM
Cell :	AEHM	Distance :	2
Cell :	ACDIJP	Distance :	3
Cell :	DJKO	Distance :	4
Cell :	ADEFGHN	Distance :	1
Cell :	ABDL	Distance :	3
Operations for the Part :			ADFGL

Cell :	AEHM	Distance :	4
Cell :	ACDIJP	Distance :	3
Cell :	DJKO	Distance :	4
Cell :	ADEFGHN	Distance :	1
Cell :	ABDL	Distance :	2

Operations for the Part : ADGLM

Cell :	AEHM	Distance :	3
Cell :	ACDIJP	Distance :	3
Cell :	DJKO	Distance :	4
Cell :	ADEFGHN	Distance :	2
Cell :	ABDL	Distance :	2

Weight = 1

Table 4.17 Cells Inference Results

Operations for the Part : ADFHJ

Cell :	AEHM	Distance :	10
Cell :	ACDIJP	Distance :	8
Cell :	DJKO	Distance :	10
Cell :	ADEFGHN	Distance :	4
Cell :	ABDL	Distance :	12

Operations for the Part : ADHJM

Cell :	AEHM	Distance :	6
Cell :	ACDIJP	Distance :	5
Cell :	DJKO	Distance :	7
Cell :	ADEFGHN	Distance :	5
Cell :	ABDL	Distance :	9

Operations for the Part : ADFJL

Cell :	AEHM	Distance :	11
Cell :	ACDIJP	Distance :	5
Cell :	DJKO	Distance :	7
Cell :	ADEFGHN	Distance :	5
Cell :	ABDL	Distance :	8

Operations for the Part : ADJLM

Cell :	AEHM	Distance :	7
Cell :	ACDIJP	Distance :	2
Cell :	DJKO	Distance :	4
Cell :	ADEFGHN	Distance :	6
Cell :	ABDL	Distance :	5

Operations for the Part : ADFGH

Cell :	AEHM	Distance :	10
Cell :	ACDIJP	Distance :	12
Cell :	DJKO	Distance :	14
Cell :	ADEFGHN	Distance :	0
Cell :	AB' L	Distance :	12

Operations for the Part : ADGHM

Cell :	AEHM	Distance :	6
Cell :	ACDIJP	Distance :	9
Cell :	DJKO	Distance :	11
Cell :	ADEFGHN	Distance :	1
Cell :	ABDL	Distance :	9

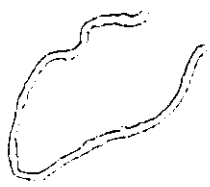
Operations for the Part : ADFGL

Cell :	AEHM	Distance :	11
Cell :	ACDIJP	Distance :	9
Cell :	DJKO	Distance :	11
Cell :	ADEFGHN	Distance :	1
Cell :	ABDL	Distance :	8

Operations for the Part : ADGLM

Cell :	AEHM	Distance :	7
Cell :	ACDIJP	Distance :	6
Cell :	DJKO	Distance :	8
Cell :	ADEFGHN	Distance :	2
Cell :	ABDL	Distance :	5

Weight: A-D W=2, E-K W=4, L-P W=1



4.4 Discussion

This chapter presents a new methodology for forming the machine cells and part families, and the integration of feature-based design with the machine cells. The cluster-seeking approach can provide different configurations for a given production and, therefore, is flexible to the extent that it can be used in a real production system. Once the manufacturing cells are formed, the automatic and dynamic assignment of parts from the design database to the cells can be done directly by the use of the proposed techniques. The automaton and pattern recognition in combination with manufacturing knowledge are applied to the feature-based parts assignment system. The approaches presented in this chapter can be applied generally to existing manufacturing cells and systems.

CHAPTER 5

FEATURE-BASED INSPECTION PLANNER

5.1 Introduction

Automated inspection is of predominant importance in computer-integrated manufacturing. Coordinate Measuring Machines (CMMs) are being used for inspection as an important means in quality control. These machines make checking the 3 dimensional geometric and dimensional tolerances possible. However, as one of the key components in flexible or computer-integrated manufacturing systems, they are not well studied in terms of operating efficiency, automated inspection task planning, and integration with other elements such as CAD. In this research, a feature-based inspection planner for a CMM has been developed and this planner has also been integrated with the feature-based design. This chapter deals with the feature-based inspection task planning.

Automated inspection planning involves the description of the real world, the representation of inspection actions and their effects on the real world, reasoning about the effects of sequences of such actions, reasoning about the interaction of actions that are taking place concurrently, and controlling the search as well as symbolic data manipulation (ElMaraghy and Gu, 1987). It is necessary to go through the whole inspection process so that an expert system can carry out the human planners' and operators' tasks. The analysis of traditional inspection processes reveals the following generic steps:

1. Understanding the part and its inspection criteria as specified in the engineering drawings;
2. Decision-making regarding the inspection procedure given the available inspection facilities;
3. Executing the inspection according to plan.

The understanding of a part and its inspection criteria is a matter of interpreting the engineering drawings and specifications. Once this is done, the inspection task may be planned for available inspection facilities and tools, using expert knowledge. Then, the inspection plans can be executed using the appropriate inspection facilities. It is clear that the most crucial steps are the interpretation of engineering drawings and the inspection planning.

The first element consists of describing a component that is compatible with the engineering drawings and describable for computer representation. Obviously, lines and numbers on the drawings cannot serve the role anymore and the FDDL has been proposed and developed as design description language. By using the FDDL, parts are described including all necessary geometric and technological constraints, such as shape, dimensions, tolerances, surface-finish. The interpretation of the description of components is another key issue for a planning system to determine the parts geometric and technical constraints. Then the system generates a sequence of inspection actions based on the interpretation of the inspection criteria using the built-in inspection strategies.

It is difficult to imagine how a human inspector determines exactly how to inspect a part since this will change from person to person, not to mention from part to part. However, the fundamental knowledge and

principles should be developed and implemented into the planner to simulate the human inspectors or inspection experts. Undoubtedly, implicit rules and laws do exist for traditional geometric and dimensional tolerances checking, and the acquisition of new knowledge requires exploration since some methods are different from traditional manual inspection to CMM inspection. This becomes an invaluable development in inspection automation.

A CMM measures a part and checks the part tolerances based on a chosen coordinate system, with either the machine or the part coordinate system using the geometric building elements. A method must be found to link the CMM representations for the real world and the description of parts in the FDDL. In order to develop the inspection strategies, a systematic analysis of CMM is essential.

5.2 Analysis of CMM Characteristics

First of all, a part is set up on the machine table or a fixture which keeps the part secure. After the part is set up on the machine, the machine requires alignment of its coordinate system, that is, a calibration. The probe's natural orientation is perpendicular to the machine table. Thus the part's accessible features in this orientation have already been determined. Each time the inspector selects a feature, such as a bore or a cylinder of the part and decides which element is to be used, he then chooses the menu to input the feature ID and to conduct the measurement. Following that, the inspector changes the menu to determine the tolerancing. At times, the above procedure may have to be repeated. For example, when a feature has a tolerance related to a datum but this datum feature has not been inspected. This process continues until all

features are done. After changes in the probe or its orientation, the inspector has to qualify the probe again by choosing an item in the menu and then measuring a calibration ball set on the machine table. If the part orientation is changed, re-calibration on the machine is required.

Inefficient inspection is usually the result of an incorrect choice in the inspection order and geometric elements. Frequently changing the probe and/or its orientations are other sources. The individual inspection of each feature is the third possibility. In order to make the whole process automated and efficient, new inspection knowledge and strategies must be systematically developed and will allow the machines to be used optimally and therefore efficient inspection procedures will be derived.

5.3 Problem Analysis

1. A part, or some features, even tolerances may not be suitable for inspection by a CMM, since some tolerances or features cannot be checked by CMMs, or are not economical. For some parts, it is more appropriate to use other inspection methods for reasons of productivity and economy of operation. This frees the CMMs to measure the more accurate parts. Thus, features and corresponding tolerances should be identified and planned for other inspection devices when appropriate.

2. The inspection tasks carried out on CMM are different from traditional manual inspection. For example, the Brown & Sharpe CMMs use several types of geometric elements such as point, line, circle, cylinder, cone, sphere and plane as a foundation for geometric inspection. All features defined in the FDDL, therefore, should be related to these elements in the final plan. The relations between geometric features defined from the design point of view and geometric elements from the

machine's point of view should be determined.

3. In order to reduce the time used in changing the menu, the features should be grouped together based on their measuring and tolerancing properties, although the geometry of their features may be completely different. From the inspection point of view, they may be considered as similar features; for example, a cylinder and a bore may both be measured as cylinders.

4. The inspection probes require qualification after they or their orientation are changed. Thus if possible, all features accessible by the current orientation of the probe should be inspected. The human inspectors can access different features by adjusting their tools and orientation. The inspection sequence often appears to be arbitrary but becomes increasingly important since the machines are expensive and should be operated efficiently. Therefore, such knowledge should be organized carefully before the implementation of an expert inspection planner.

5. The CMMs are not common machines. Their operational characteristics are not well investigated and essentially, a set of inspection strategies is required to optimally use the machines.

All of these problems and the associated inspection knowledge should be represented and coded in the expert planner. Therefore, the system, the data structure, and the corresponding symbolic data manipulating techniques should be well designed.

5.4 Code Generator for Inspection Planning

Before the syntax for the target language is defined, the information required and the necessary data type structure should be determined by the planner. These also relate to the knowledge

representation and the whole system design. As discussed in Chapter 2, the information for inspection planning includes the feature identifier, the feature type, the main dimensions of the feature geometry, the tolerances and related datums, and the location and orientation of the feature. Based on the data structure and the manipulating techniques, the input format of the target language has been defined to the inspection planning system. The designed syntax is included in Appendix A. An example of the feature description in the target language is:

ID Feature Type Main Dimensions Tolerances Location List

"ID" is an integer considered as an identification number by the CMM. "Feature Type" provides information on the geometric properties which may be related to point, line, circle, plane, cylinder, cone and sphere, the 7 elements used by the CMM. The features are also used to determine whether they are suited to CMM inspection or manual inspection. "Main dimensions" are parameters which define the feature geometry. For example, a cylinder has its diameter and length as the main dimensions. The tolerances type and their datums provide important information for the planning of the inspection. "Location List" contains position and orientation. The orientation information may be used to determine feature accessibility, probe orientations and to determine the planning inspection sequences. For example, two faces of the cylinder (shown in Figure 2.1) have different orientations. It follows that one face may be touched by a probe and the other cannot be touched if the part orientation is not changed.

The code generator creates an output according to the specified syntax provided in the source code. As mentioned previously, the language

is not only used to model an individual part, but also to model a product which may contain a number of parts. The inspection is performed for individual parts. Therefore, the code generator will create a file containing the code for every part in the product. The name of each file is derived from the part identifier. How the code generator works is described in the previous chapter. Some particular treatments for inspection are discussed below:

As same as before, for each occurrence of "feature", a keyword is sought. These keywords are classified as follows:

"name"

"type"

"location"

"tolerance"

"surface_finish"

"relation"

If one of the above classes is recognized, the expected information is sought out and verified. Here only tolerances are discussed, the others can be found in Chapter 4.

tolerance: A tolerance keyword is expected as each one of the tolerances is defined. If one of these tolerances is matched, the following parameters and possible datums are searched. An important point is that the tolerances in one feature may contain more than one type, but only one list is created. The language grammar is context-free and this list is kept until the whole feature is completed such that it can be appended when new tolerances occur. Another reason for creating the

list is that one tolerance may or may not have datum, or may have more than one datum. These possibilities require that the code generator has some intelligence.

Since the CMM recognizes the feature by identifying them as integers, the corresponding feature identifiers must be replaced by integers which are consistent with the whole part. This means that the features' ID should uniquely appear and be referred to by other features as tolerances datum.

For the inspection planning, the system processes one feature and prints it in the syntax defined before part input. The processing and printing are conducted until all features are done and the 'stop input' is given as a terminal mark in the input file. The inspection planning does not concern itself with the surface-finish and the relations and thus, they are not printed - only the checking is performed.

5.5 Development of Inspection Knowledge

By examining the characteristics of CMMs, the inspection process, the parts geometric features, the feature tolerances and tolerancing theory, the following inspection strategies have been developed based on the Brown & Sharpe Machine shown in Figure 5.1. This procedure is generally applicable to other machines as well. The inspection process has two major portions, measurement and tolerancing.

1. First of all, the input part description is examined to determine which features should be inspected using a non-CMM device regarding geometric tolerances inspection since some, such as total runout, cannot be measured on the CMMs. Then, the corresponding planning is done for these

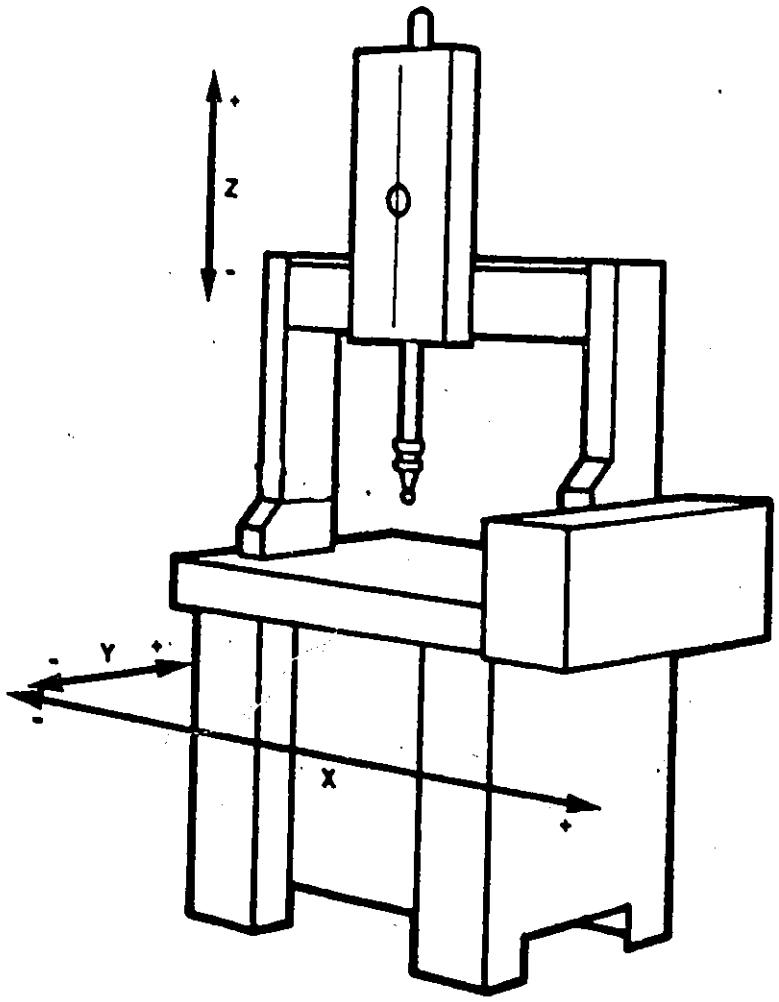


Figure 5.1 Brown & Sharpe Coordinate Measuring Machine

measurements.

2. As long as the part is fixed on the table, its orientation is determined. All features should then be examined for accessibility.
3. When the accessibility of all the features is determined, the feature datum should be searched, and recognized by identification of its position in the list data structure. Next, the measurement planning and corresponding tolerance checking can be conducted. Since the CMM checks the features geometric elements instead of recognizing the exact shape of the features, then all defined features must be decomposed and represented by these geometric elements.
4. When the first item is measured, an appropriate probe should be chosen. All accessible features which can be considered as the same geometric element should be inspected at one time, this being based upon efficient considerations of the CMM operation. For example, all points, and all circle measurement should be grouped together and carried out at the same time.
5. This step determines which features belong to other geometric elements from the inspection point of view, and which can be inspected by the current probe. Then, step 4 is repeated. This process is repeated until all accessible items, which can be inspected by the present probe, have been checked.
6. A feature is chosen and a suitable probe is selected. Then step 4 and 5 are repeated. In this way, all accessible features are measured and geometric tolerance checking can begin.

7. The datum features' tolerances are checked.
8. The same tolerance types are sought out for all the accessible features. Otherwise, the next tolerance item begins.
9. If the first chosen tolerance requires a datum, then all the measured features tolerances based on this datum, are found and compared with designed tolerances.
10. The above process is repeated until all the tolerances are completed.
11. Then, the probe orientation is changed and the above measurement and tolerancing process is repeated until all the features accessible to the changing probe's orientations are planned.
12. The part orientation is changed, and steps 1 to 10 are repeated until part inspection is completed.

5.6 Knowledge Representation

The inspection knowledge is represented in an integration scheme of frames and production rules where a number of features are described with dimensions and various tolerances. If they are represented in the form of production rules, then the same features can have different orientations and tolerances as well as different relations with other features. This may result in a very complicated structuring of the system. In fact, if all features are implemented in production rules, the knowledge base will be extremely large. On the other hand, frames are suitable for describing objects because they have slots for objects' properties. However, they lack the flexibilities once they are used to represent reasonably complicated relations and strategies. Thus, a combination scheme is

developed to completely represent the inspection knowledge. A frame contains a frame identification number as the frame name, one slot for the object description, two slots for the main dimensions, and an undetermined number of slots for the tolerances. In this way, all input part descriptions can be represented. Unlike object-oriented programming, PROLOG does not have this facility of inheritance messages and, therefore, the relations between the objects and the inspection knowledge must be explicitly expressed, particularly the meta-knowledge in the production rules. This is straightforward. One feature occupies a frame and one frame is represented in the system in a list data structure. Each element corresponds to a slot value. Therefore, a whole part has a super list in which each feature is included shown below:

```
[
    [ID, feature name, dimension1, dimension2,
        list of tolerances, [orientation list]],
    [ID, feature name, dimension1, dimension2,
        list of tolerances,[orientation list]],
    .....,
]
```

5.7 Implementation of Inspection Planning

The inspection planning system is implemented in MPROLOG (Logicware, 1986). The search algorithm provided by PROLOG is the depth-first, left to right search. The order of appearance of the facts and rules in the database guides this control process. The inference engine consists of two phases, pattern matching and unification. The pattern matching is a PROLOG search strategy to match predicates with the same number of arguments. Unification is simply argument matching.

The feature-based inspection planner, shown in Figure 5.2, consists of one main module and several sub-modules. Each of them has its role in

inspection planning. Modular structure allows the system to be easily extended and modified.

5.8 Planner Control Strategies

The control strategies are actually inspection strategies which hold consideration for the chosen data structure, the representation schemes of knowledge, and also the language to be used. Since PROLOG is used to code the system, backward chaining is employed by the control structure. An outline of the control mechanism is explained as follows:

1. ask user to input the part class, either rotational or nonrotational.
2. if the input is either of them, then continue, otherwise, print an error message.
3. select a fixture.
4. ask the user to input the three direction parameters which indicate that features with such orientations are parallel to the machine table orientation.
5. if correct, continue, otherwise print an error message and return to 1.
6. read a feature description and include it in a list if 'stop input' is met go to 7, otherwise continue.
7. call a non-CMM module to plan those features unsuitable for the CMM. In the mean time, cancel all planned features' tolerance items and keep the features such that they can be used in the determination of feature accessibilities.
8. process the features accessibilities based on input three directional parameters for nonrotational part. For the rotational part, extra checking must be performed for all axial features based on their

geometric shapes and dimensions.

9. search datums for all ready measurable features. If one is found, then a moving process is conducted. The moving process first checks whether the moved datum feature requires another datum. If the answer is yes, then move this datum feature behind the related feature. If not, then move it in front of the whole list. This process continues until all features are examined and the list is ordered.
10. start planning measurement. When a feature is planned, the feature list is deleted from the whole list data structure. This process continues until all accessible features are deleted in the current probe orientation.
11. a parallel whole list is used to plan the tolerancing; the difference being that after each tolerance is planned, the feature is examined to determine if it has other tolerances. If yes, the feature list is kept in the original position. If no, the feature is deleted from the whole list. This process terminates if all accessible features at the current probe orientation are deleted.
12. change the probe orientation if some features can be accessed, then go to 9.
13. if, by changing the probe orientation, all accessible features are planned, then change the part orientation and go to 9.
14. if all features are planned, then stop the process.

These are the general control strategies. More detail will be given with a production rule example later.

5.9 Knowledge Base

The knowledge base consists of more then 150 rules and 72 facts. The rules include the control structure and the entire process works in the same manner as the described control strategies. A more detailed description with the rules for the major steps is provided later. The examples with PROLOG code are referenced in ElMaraghy and Gu (1987).

5.9.1 Feature Base

The feature base is composed of the feature facts. A feature can be expressed as a sequence of feature number, feature name, main dimensions and tolerance items. This type of description takes into consideration both the geometric and technical descriptions of features, the outputs of language system, as well as the characteristics of PROLOG. In order to represent the features in terms of 7 geometric elements, the relations between the features and the elements must, in some way, be determined.

Some 3D features can be considered as 2D from a geometric tolerance checking point of view. This can save measurement time and data processing time because 2D tolerances require less measuring points and simpler fitting algorithms. An example is shown in Figure 5.3. All features and their dimensional possibilities regarding geometric tolerances, were examined and a related knowledge base was built up to deal with these relations. All features defined in the feature base are divided into geometric element classes. A feature which is defined as 3 dimensional is examined as a 3D feature. Its order is decreased if it can be thought of as 2D. For example, let us examine a cylinder feature composed of three

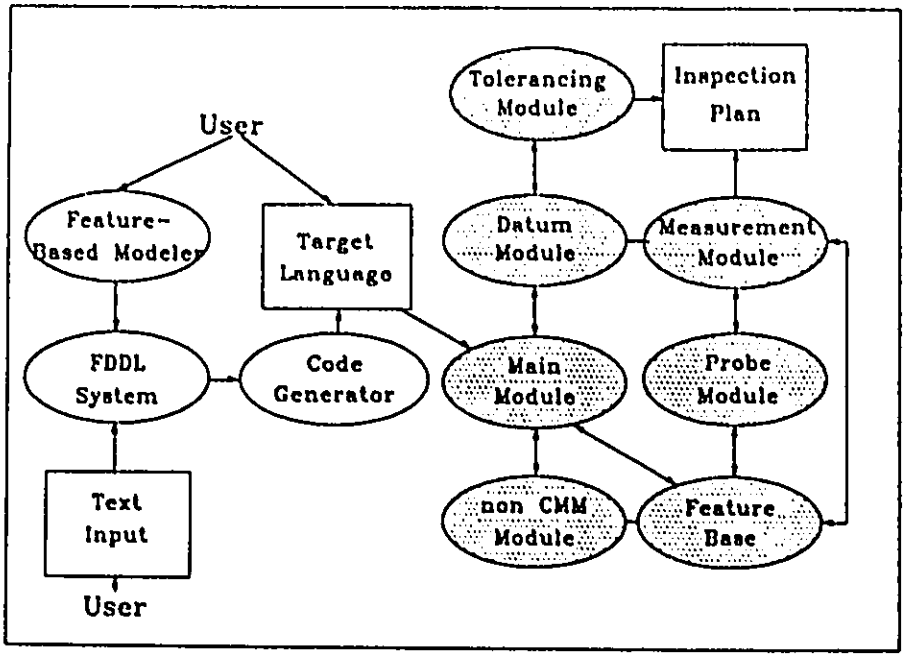


Figure 5.2 Structure of Feature-Based Inspection Planner

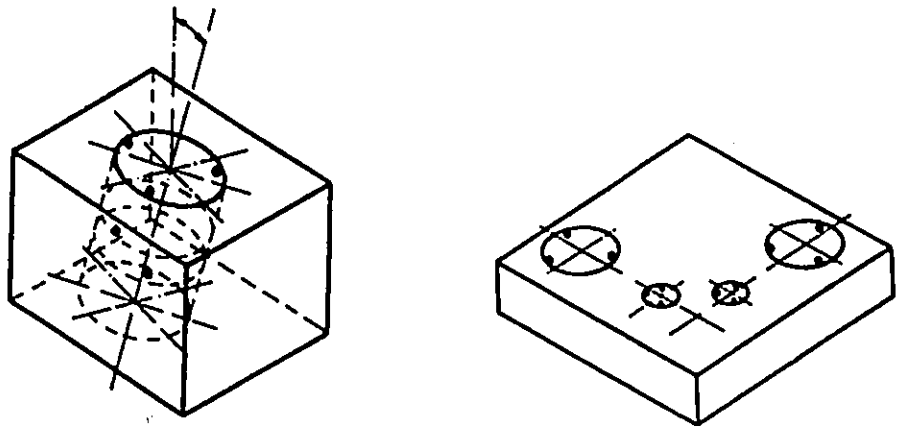


Figure 5.3 Examples of Cylinders and Circles Measurements

elements, one surface and two planes. The surface is considered as a 3D feature. If only 2D tolerance checking is required, its rank is decreased to 2D and it can be measured as a circle. This process is carefully conducted by the PROLOG search strategies. The other two faces are also considered as plane features if they have some 3D tolerance requirements, otherwise they are considered as point features. In the pattern matching search, they are first checked as planes, then as points. Some features are examined as a plane, a line and then a point. The advantage of this procedure is that no possible 3D tolerance checking can be missed. All available features, with their possible measurement element representation in CMM, geometric entity nature (solid, empty) as well as the preferable inspection method are included in the feature base Appendix E. For example;

```
feature(cylinder_side, cylinder, solid, machine);
feature(cylinder_side, circle, solid, machine);
feature(cylinder_front, plane, solid, machine);
feature(cylinder_front, point, solid, machine);
.
.
.
```

From this sample, it is evident that all possible relations between the features and the 7 building elements are defined. Based on the features' tolerances, the final representation element can be determined. Since the pattern matching process follows the fact appearing sequence, the former is a 3D element.

5.9.2 Selection of Inspection Facilities

The selection of inspection facilities is simplified whereby

assignment is made for non-CMM inspection and CMM inspection. All features and tolerances items that are suitable for non-CMM inspections are identified. After planning non-CMM inspection tasks, the remaining features can be identified and planned for CMM.

Some general rules about non-CMM inspection are shown as follows:

IF the feature is a gear tooth and;
its gear tooth profile has some specified tolerances
and; it has circular or total runout geometric
tolerances;
THEN the gear tooth should not be inspected on the CMM.

IF the feature is a screw and;
its pitch requires inspection and;
its tolerances specification is not too tight,
THEN it can be inspected with template or gauges.

After all features are examined, those suitable to non-CMM are planned, and these features are kept in the part list. However, their tolerance items are removed and they lose their inspection criterion, but they keep their geometric characteristics for feature accessibilities checking.

5.9.3 Features Accessibilities

Once a part is fixed on the machine table, some features can be touched directly by the probe in its original natural orientation. Some features may be accessed by adjusting the probe's orientation. Others can only be reached by changing the part's orientation. Thus features accessibilities must be determined and arranged so that the changes of the part and/or the probe orientation can be identified and planned. This

results in an investigation of all the features' geometric properties and orientations in this sense. For example, in Figure 2.2, the cylinder with diameter 100 is considered inaccessible if the part orientation is such that the cylinder with diameter 140 is closer to the probe. Of course, the feature accessibility checking is more complex than that. It includes identification of all accessible features, grouping them based their orientations and locations, and sorting them in the sequence of planning.

All features are first grouped with the same orientations, then the inspection sequence of groups is grouped. First of all, the whole part is examined to form a list which contain a few groups based on the orientations. That is to say, the features with the same orientation and the features without tolerances specification features become a group. For example:

IF feature(i) and feature(i+1) have the same orientation;

THEN put feature(i+1) in the group with feature(i).

ELSE leave the feature(i+1) in the original place.

IF all features are compared with feature(i);

THEN feature(i) group is formed.

IF original list is empty;

THEN terminal grouping

ELSE pick first feature as new feature(i), continue above.

After the processing, a new list is formed. It contains the same number of elements as before. The only difference is that the feature order may be changed. Then, the inspection sequence is determined for groups.

IF the orientation of the group(i) is same as the ORIEN

which is the orientation to the probe;

THEN the group should be inspected first.

IF the orientation of group(i) is opposite to ORIEN;

THEN the group should be inspected after the part's orientation is changed.

IF the orientation of the group is neither same as ORIEN nor opposite to ORIEN;

THEN the group should be inspected after the probe's orientation is changed.

A feature's dimensions may affect the feature accessibility, for example:

IF the feature(i) is a cylinder and;
the feature(i+1) is a cone and ;
the cone max. diameter is smaller than
the cylinder diameter;

THEN the cone is inaccessible in this orientation.

Features which are accessible without changing the probe orientation are stored in certain positions in a list data structure, the features are reached when probe orientation is adjusted are kept in other lists. The others are left in positions where they can be accessed when the part orientation is changed.

5.9.4 Datum Features Search and Arrangement

Important features, from both the parts function and their geometric accuracy point of view, are inspected first. Datums can be thought of as important features. The primary datum is checked first since others may relate to it. Figure 2.3 shows an example of a bore with diameter 40 as a datum. Thus it should be inspected first since other features relate to it. Only this datum is qualified; the others can be

measured relative to it. The search for datum becomes a starting point of an inspection sequence planning. This search process is carried out by examining the geometric tolerances of every feature to find the tolerances and datums. Then the datum features are re-arranged in the list. If this search fails, it means that no datum exists. The important feature selection is in front-end order. This order is determined after features accessibility. There are a few problems arising regarding the datums. First of all, a feature with a tolerance requires a datum. Then the system searches the datum feature and move it to the front of the list. In fact, it is common or possible that a feature may have several tolerances required datums. Their sequence should be determined. Also, if one feature is the datum of several features, the primary datum must be determined. Then, the other datums are arranged following the similar idea. This requires tracking of datum feature situations. During the processing, a datum list is formed as a reference. An example of searching for a datum is as follows:

IF all features of one group are examined regarding
the datums,
THEN terminate the process,
ELSE continue.
IF the feature contains a geometric tolerance and;
it is related to a datum; and if the datum
is not in the reference list of datums, and
it does not require a datum;
THEN append the datum feature identifier in the
reference list, then match the datum and place
it in front of the list,

ELSE if it requires a datum; put the feature behind
the datum, otherwise move to next feature.

5.9.5 Measurement and Tolerancing Planning

Once all the features in one orientation group are examined in terms of datum features, then the planner moves to the planning measurement phase. For the first feature as a datum, an appropriate probe is chosen to carry out this measurement. The probe selection is based upon the feature's geometric properties, the location of the part and the corresponding tolerances. Since a feature is represented in a list, four elements of the list show the feature number and name as well as the main dimensions. The later three characteristics together with tolerances are used to select the probe according to pre-determined rules. After that, all accessible features are planned, based on measuring points. Here is an example for determining the measuring points:

IF the feature can be considered as
a cylindrical one and its geometric
tolerances contain one 3D form,
orientation, location and position;
THEN it is measured as cylinder feature
and the minimum measuring points are 5.

An example for inspecting all accessible features of this item is:

IF the end mark is met; and group becomes empty,
THEN move to the tolerancing phase.
IF the end mark is met; the group is not empty;
THEN plan another feature and determine the
measurement point.

If the chosen feature is cylindrical, such as a hole, cylinder, or other cylindrical feature, then subsequent feature checking should be done as follows:

IF a feature belongs to cylindrical feature
it has one of 3D form, orientation,
location and position tolerances and ;
it matches the current geometric element
THEN measure this feature with a minimum of 5 points.

Once the geometric measurements have been finished, the tolerance checking process is activated. This process starts by checking whether the datum feature (or first chosen feature if no datum is found) exceeds the tolerance. Then, the same tolerance type for all accessible features is checked. This grouping of tasks saves inspection time. Example:

IF all tolerances are planned,
THEN the process terminates
ELSE continue
IF a tolerance item is chosen,
THEN search the same tolerance type and plan it
IF all features are examined for the current
tolerance item and this tolerance is planned,
THEN choose a new tolerance item.

Upon completion of tolerance checking, the part or probe orientation is changed as necessary and probe or part alignment is performed, then the above process is repeated until the whole part is inspected.

5.10 A Case Study

This section considers three application examples, a rotational and two prismatic parts. These are shown in Figure 2.3, 3.10 and 5.8. For this rotational part, the part is set such that all features on the left side of the biggest diameter cylinder (D140) can be accessed by a probe and the others are checked after the part is turned over. The source language is shown in the Figure 2.3 and the code generated by the code generator is given in Figure 5.4. The two screw features, two cylinders and a keyseat do not require inspection by a CMM, hence, they do not appear in the final inspection plan. However, they must be input to the system because these features may play a function in determination of features accessibilities. This system function becomes extremely important when the system is linked directly with a CAD database. From the plan, it is clear that the datum feature is bore3 and is, therefore, measured first based on the inspection knowledge. The other two features, bore2 and cylinder4 require the same number of measuring points, hence, they are measured together. Following this, the point measurement same tolerance checking is grouped. As mentioned before, this results in time saved by not changing software menu. After changing the part orientation, there are two cylinder_front features, face10 and face12. Because they require different tolerance specifications, the distance of face12 from face10 is inspected. This can be treated as a point feature. The feature face12 requires a check on its perpendicularity. It is, thus, a plane feature, and the minimum number of measurement points is three (see Figure 5.5).

Another example shown in Figure 3.10 was modeled by the Feature-Based Modeler. After the output shown in Figure 3.11 was processed by

```
1 thread_nf_front 70.0e0 0 [175.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
2 bore 50.0e0 25.0e0 diameter concentricity 3 [175.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
3 bore 40.0e0 150.0e0 diameter cylindricity [150.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
4 thread_nf_side 70.0e0 20.0e0 [175.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
5 cylinder_front 80.0e0 0 distance 6 [155.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
7 cylinder_side 80.0e0 25.0e0 diameter concentricity 3
[155.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
8 t_cone_front 90.0e0 0 perpendicularity 3 [130.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
9 t_cone_side 110.0e0 30.0e0 angularity 3 [130.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
10 cylinder_side 110.0e0 10.0e0 [100.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
6 cylinder_front 140.0e0 0 [90.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
11 cylinder_side 140.0e0 30.0e0 [90.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
12 cylinder_front 140.0e0 0 perpendicularity 3 [60.0e0,0.0e0,0.0e0,180.0e0,–90.0e0,90.0e0]
13 cylinder_side 100.0e0 40.0e0 diameter concentricity 3
[60.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
14 cylinder_front 100.0e0 0 distance 12 [20.0e0,0.0e0,0.0e0,180.0e0,–90.0e0,90.0e0]
15 thread_nf_side 70.0e0 20.0e0 [20.0e0,0.0e0,0.0e0,0.0e0,90.0e0,–90.0e0]
16 thread_nf_front 70.0e0 0 [0.0e0,0.0e0,0.0e0,180.0e0,–90.0e0,90.0e0]
17 p_keyseat_side 15.0e0 10.0e0 [40.0e0,40.0e0,6.0e0,–90.0e0,90.0e0,180.0e0]
18 p_keyseat_side 15.0e0 10.0e0 [40.0e0,40.0e0,–6.0e0,90.0e0,–90.0e0,0.0e0]
19 p_keyseat_corner 6.0e0 10.0e0 [37.0e0,50.0e0,0.0e0,–90.0e0,0.0e0,90.0e0]
20 p_keyseat_corner 6.0e0 10.0e0 [52.0e0,50.0e0,0.0e0,–90.0e0,0.0e0,90.0e0]
stop input
```

Figure 5.4 Target Language of the Part Shown in Figure 2.3 for Inspection Planner

the FDDL system, the target code for inspection planning is given in the Figure 5.6. The orientation of the part on the machine table is shown in Figure 3.10. The one hole with ID 1 and one face with ID 5 are accessible before the part is turned over, based on the orientations. The results are given in Figure 5.7.

Another demonstration example is shown in Figure 5.8, where the target language is directly input by the user to the expert inspection planner following the target language syntax, which is shown in Figure 5.9. The prismatic part is more difficult from a traditional inspection point of view, and is, therefore, more suitable to a CMM. The 37 features located in different orientations require inspection. This means that the probe's orientation should be changed after all features in one plane are checked. The part orientation also requires changing after all the other features are checked. Since the part is reasonably complicated, more than sixty thousands function calls were required and sixty hundred backtracks were included. The results are given in Figure 5.10. The feature number 18 is related by the number of features, and is checked first. Since there are no features with similarity (as a plane of 3 points), the cylinder_side (feature no. 2) is chosen and following the same property features, number 1 bore is measured. Then, the circle features are planned for feature no. 3, 4 and 5. Since the probe requires requalification after its orientation is changed, all features are planned to be measured before the probe orientation is changed. From this result, it is clear that all the same types of tolerances are grouped together.

PARTS INSPECTION PLANNING

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
bore	3	40	150	cylinder	5	iad7084
bore	2	50	25	cylinder	5	iad7084
cylinder_side	7	80	25	cylinder	5	iad7084
cylinder_front	5	80	0	point	1	iad7084
t_cone_front	8	90	0	plane	3	iad7084
t_cone_side	9	110	30	cone	6	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
bore	3	40	150	diameter	
bore	2	50	25	diameter	
cylinder_side	7	80	25	diameter	
bore	3	40	150	cylindricity	
bore	2	50	25	concentricity	3
cylinder_side	7	80	25	concentricity	3
t_cone_front	8	90	0	perpendicularity	3
t_cone_side	9	110	30	angularity	3
cylinder_front	5	80	0	distance	6

Change Part Orientation

Figure 5.5 Inspection Plan for the Part Shown in Figure 2.3

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
cylinder_front	12	140	0	plane	3	iad7084
cylinder_front	14	100	0	point	1	iad7084
cylinder_side	13	100	40	cylinder	5	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
cylinder_front	12	140	0	perpendicularity	3
cylinder_side	13	100	40	concentricity	3
cylinder_side	13	100	40	diameter	
cylinder_front	14	100	0	distance	12

Planning is finished

Figure 5.5 Inspection Plan for the Part Shown in Figure 2.3 (Cont.)

```
1 bore 2.4e0 5.0e0 diameter parallelism 2 distance 3 distance 4
[26.0e0,30.0e0,22.0e0,90.0e0,0.0e0,–90.0e0]
5 box_face 20.0e0 15.0e0 flatness [20.0e0,30.0e0,25.0e0,90.0e0,0.0e0,–90.0e0]
6 box_face 20.0e0 12.0e0 distance 7 parallelism 7
[20.0e0,30.0e0,10.0e0,90.0e0,–90.0e0,180.0e0]
7 box_face 20.0e0 12.0e0 perpendicularity 5 [20.0e0,30.0e0,25.0e0,–90.0e0,90.0e0,0.0e0]
3 box_face 12.0e0 15.0e0 perpendicularity 5 [10.0e0,18.0e0,25.0e0,180.0e0,90.0e0,–90.0e0]
8 box_face 12.0e0 15.0e0 [30.0e0,18.0e0,25.0e0,0.0e0,–90.0e0,90.0e0]
4 box_face 20.0e0 15.0e0 parallelism 5 [20.0e0,18.0e0,25.0e0,–90.0e0,180.0e0,90.0e0]
2 bore 3.6e0 8.0e0 diameter distance 3 distance 6
[15.0e0,30.0e0,18.0e0,–90.0e0,180.0e0,90.0e0]
9 bore 4.80e0 4.0e0 diameter distance 7 concentricity 2
[15.0e0,22.0e0,18.0e0,–90.0e0,180.0e0,90.0e0]
stop input
```

Figure 5.6 Target Language for the Prismatic Part Shown in Figure 3.10

PARTS INSPECTION PLANNING

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
bore	1	2.4E0	5	cylinder	5	iad7084
box_face	5	20	15	plane	3	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
bore	1	2.4E0	5	diameter	
bore	1	2.4E0	5	parallelism	2
box_face	5	20	15	flatness	
bore	1	2.4E0	5	distance	3
bore	1	2.4E0	5	distance	4

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
box_face	6	20	12	plane	3	iad7084

****
** Geometric Tolerancing **
****

Figure 5.7 Inspection Plan for the Part Shown in Figure 3.10

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
box_face	6	20	12	distance	7
box_face	6	20	12	parallelism	7

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
box_face	7	20	12	plane	3	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
box_face	7	20	12	perpendicularity	5

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
box_face	3	12	15	plane	3	iad7084

****
** Geometric Tolerancing **
****

Figure 5.7 Inspection Plan for the Part Shown in Figure 3.10 (Cont.)

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
box_face	3	12	15	perpendicularity	5

Change Part Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
bore	2	3.6E0	8	circle	3	iad7084
box_face	4	20	15	plane	3	iad7084
bore	9	4.8E0	4	cylinder	5	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
bore	2	3.6E0	8	diameter	
bore	9	4.8E0	4	diameter	
box_face	4	20	15	parallelism	5
bore	9	4.8E0	4	concentricity	2
bore	2	3.6E0	8	distance	3
bore	9	4.8E0	4	distance	7
bore	2	3.6E0	8	distance	6

Planning is finished

Figure 5.7 Inspection Plan for the Part Shown in Figure 3.10 (Cont.)

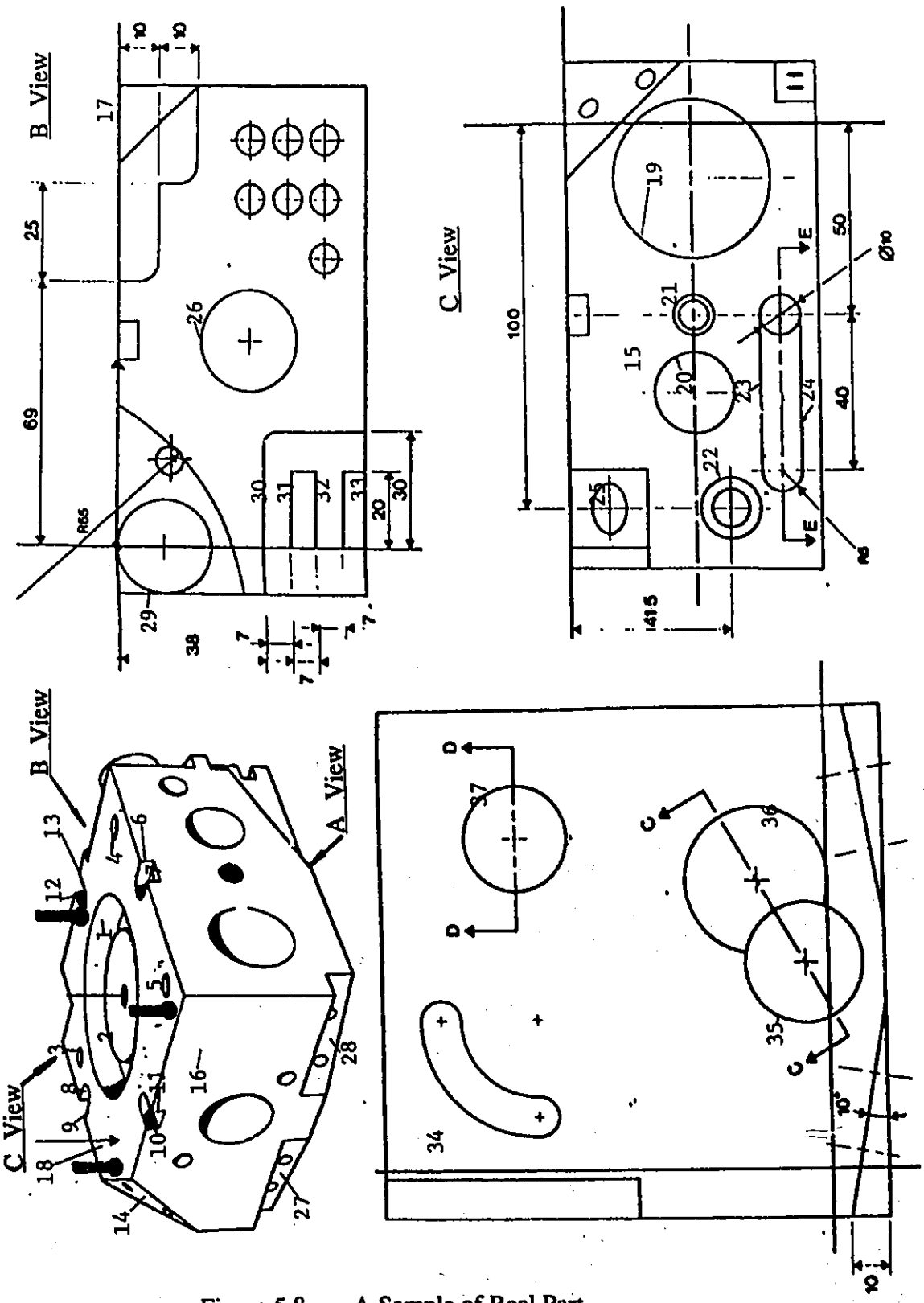


Figure 5.8 A Sample of Real Part

```
1 bore 80 10 diameter concentricity 2 [65.0e0,65.0e0,34.0e0,90.0e0,–90.0e0,0.0e0]
2 cylinder_side 80 5  diameter perpendicularity 18 [65.0e0,65.0e0,34.0e0,90.0e0,–90.0e0,0.0e0]
3 bore 13 15 diameter position 0 [25.0e0,40.0e0,34.0e0,90.0e0,–90.0e0,0.0e0]
4 bore 13 15 diameter position 0 [25.0e0,25.0e0,34.0e0,90.0e0,–90.0e0,0.0e0]
5 bore 13 15 diameter position 0 [40.0e0,40.0e0,34.0e0,90.0e0,–90.0e0,0.0e0]
6 p_keyseat_side 5 5 distance 16 [62.5e0,62.5e0,34.0e0,180.0e0,–90.0e0,90.0e0]
7 p_keyseat_side 5 5 distance 6 [62.5e0,62.5e0,34.0e0,0.0e0,90.0e0,–90.0e0]
8 p_keyseat_side 5 5 distance 16 [62.5e0,62.5e0,34.0e0,180.0e0,–90.0e0,90.0e0]
9 p_keyseat_side 5 5 distance 8 [62.5e0,62.5e0,34.0e0,0.0e0,90.0e0,–90.0e0]
10 p_keyseat_side 5 5 distance 19 [62.5e0,62.5e0,34.0e0,180.0e0,–90.0e0,90.0e0]
11 p_keyseat_side 5 5 distance 10 [62.5e0,62.5e0,34.0e0,0.0e0,90.0e0,–90.0e0]
12 p_keyseat_side 5 5 distance 19 [62.5e0,62.5e0,34.0e0,180.0e0,–90.0e0,90.0e0]
13 p_keyseat_side 5 5 distance 12 [62.5e0,62.5e0,34.0e0,0.0e0,90.0e0,–90.0e0]
18 box_face 130 130 flatness [35.0e0,34.0e0,23.0e0,90.0e0,–90.0e0,0.0e0]
14 plane 30 20 angularity 18 [30.0e0,30.0e0,30.0e0,60.0e0,50.0e0,30.0e0]
17 plane 30 42 angularity 18 [10.0e0,6.0e0,30.0e0,85.0e0,–45.0e0,70.0e0]
15 box_face 68 130 flatness perpendicularity 18
[ –35.0e0,34.0e0,23.0e0,180.0e0,–90.0e0,90.0e0]
19 bore 40 130 diameter cylindricity position 0 [–34.0e0,0.0e0,34.0e0,180.0e0,–90.0e0,90.0e0]
20 bore 20 130 diameter position 0 [–34.0e0,0.0e0,34.0e0,180.0e0,–90.0e0,90.0e0]
21 bore 10 130 diameter position 0 [–34.0e0,0.0e0,34.0e0,180.0e0,–90.0e0,90.0e0]
22 bore 15 20 diameter position 0 [–34.0e0,0.0e0,34.0e0,0.0e0,90.0e0,–90.0e0]
23 p_keyseat_side 5 40 distance 18 [62.5e0,62.5e0,34.0e0,90.0e0,–90.0e0,0.0e0]
24 p_keyseat_side 5 40 distance 23 [62.5e0,62.5e0,34.0e0,–90.0e0,90.0e0,180.0e0]
16 box_face 68 130 flatness perpendicularity 18 [–35.0e0,34.0e0,23.0e0,0.0e0,90.0e0,–90.0e0]
25 bore 12 10 diameter angularity 18 [–34.0e0,0.0e0,34.0e0,20.0e0,40.0e0,70.0e0]
26 bore 25 130 diameter cylindricity position 0 [–34.0e0,0.0e0,34.0e0,–90.0e0,0.0e0,90.0e0]
27 plane 10 34 angularity 16 [10.0e0,6.0e0,30.0e0,5.0e0,45.0e0,70.0e0]
28 plane 10 34 angularity 16 [10.0e0,6.0e0,30.0e0,9.0e0,15.0e0,40.0e0]
29 cylinder_side 21 5 diameter position 0 [–34.0e0,0.0e0,34.0e0,–90.0e0,0.0e0,90.0e0]
30 p_keyseat_side 7 20 distance 34 [62.5e0,62.5e0,34.0e0,90.0e0,–90.0e0,0.0e0]
31 p_keyseat_side 7 20 distance 30 [62.5e0,62.5e0,34.0e0,–90.0e0,90.0e0,180.0e0]
32 p_keyseat_side 7 20 distance 34 [62.5e0,62.5e0,34.0e0,90.0e0,–90.0e0,0.0e0]
33 p_keyseat_side 7 20 distance 32 [62.5e0,62.5e0,34.0e0,–90.0e0,90.0e0,180.0e0]
34 box_face 130 130 flatness parallelism 18 [–35.0e0,34.0e0,23.0e0,–90.0e0,90.0e0,180.0e0]
35 bore 30 20 diameter position 0 [0.0e0,34.0e0,–34.0e0,–90.0e0,90.0e0,180.0e0]
36 bore 36 20 diameter distance 35 position 0 [20.0e0,24.0e0,–34.0e0,–90.0e0,90.0e0,180.0e0]
37 bore 25 40 diameter position 0 [80.0e0,44.0e0,–34.0e0,–90.0e0,90.0e0,180.0e0]
stop input
```

Figure 5.9 Input Target Language of Part Shown in Figure 5.8

PARTS INSPECTION PLANNING

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
box_face	18	130	130	plane	3	iad7084
cylinder_side	2	80	5	cylinder	5	iad7084
bore	1	80	10	cylinder	5	iad7084
bore	3	13	15	circle	3	iad7084
bore	4	13	15	circle	3	iad7084
bore	5	13	15	circle	3	iad7084
p_keyseat_side	6	5	5	point	1	iad7084
p_keyseat_side	7	5	5	point	1	iad7084
p_keyseat_side	8	5	5	point	1	iad7084
p_keyseat_side	9	5	5	point	1	iad7084
p_keyseat_side	10	5	5	point	1	iad7084
p_keyseat_side	11	5	5	point	1	iad7084
p_keyseat_side	12	5	5	point	1	iad7084
p_keyseat_side	13	5	5	point	1	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
box_face	18	130	130	flatness	
cylinder_side	2	80	5	diameter	
bore	1	80	10	diameter	
bore	3	13	15	diameter	
bore	4	13	15	diameter	
bore	5	13	15	diameter	
cylinder_side	2	80	5	perpendicularity	18
bore	1	80	10	concentricity	2

Figure 5.10 Inspection Plan for the Part Shown in Figure 5.8

bore	3	13	15	position	0
bore	4	13	15	position	0
bore	5	13	15	position	0
p_keyseat_side	6	5	5	distance	16
p_keyseat_side	7	5	5	distance	6
p_keyseat_side	8	5	5	distance	16
p_keyseat_side	9	5	5	distance	8
p_keyseat_side	10	5	5	distance	19
p_keyseat_side	11	5	5	distance	10
p_keyseat_side	12	5	5	distance	19
p_keyseat_side	13	5	5	distance	12

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
plane	14	30	20	plane	3	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
plane	14	30	20	angularity	18

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
plane	17	30	42	plane	3	iad7084

Figure 5.10 Inspection Plan for the Part Shown in Figure 5.8 (Cont.)

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
plane	17	30	42	angularity	18

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
box_face	15	68	130	plane	3	iad7084
bore	19	40	130	cylinder	5	iad7084
bore	20	20	130	circle	3	iad7084
bore	21	10	130	circle	3	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
box_face	15	68	130	flatness	
box_face	15	68	130	perpendicularity	18
bore	19	40	130	diameter	
bore	20	20	130	diameter	
bore	21	10	130	diameter	
bore	19	40	130	cylindricity	
bore	19	40	130	position	0
bore	20	20	130	position	0
bore	21	10	130	position	0

Change Probe Orientation

Figure 5.10 Inspection Plan for the Part Shown in Figure 5.8 (Cont.)

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
p_keyseat_side	23	5	40	point	1	iad7084
p_keyseat_side	24	5	40	point	1	iad7084
bore	22	15	20	circle	3	iad7084
box_face	16	68	130	plan	3	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
p_keyseat_side	23	5	40	distance	18
P_keyseat_side	24	5	40	distance	23
box_face	16	68	130	perpendicularity	18
bore	22	15	20	diameter	
bore	22	15	20	position	0
box_face	16	68	130	flatness	

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
bore	25	12	10	cylinder	5	iad7084

****
** Geometric Tolerancing **
****

Figure 5.10 Inspection Plan for the Part Shown in Figure 5.8 (Cont.)

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
bore	25	12	10	diameter	
bore	25	12	10	angularity	18

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
bore	26	25	130	cylinder	5	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
bore	26	25	130	diameter	
bore	26	25	130	cylindricity	
bore	26	25	130	position	0

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
plane	27	10	34	plane	3	iad7084

****
** Geometric Tolerancing **
****

Figure 5.10 Inspection Plan for the Part Shown in Figure 5.8 (Cont.)

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
plane	27	10	34	angularity	16

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
plane	28	10	34	plane	3	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
plane	28	10	34	angularity	16

Change Probe Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
p_keyseat_side	30	7	20	point	1	iad7084
p_keyseat_side	31	7	20	point	1	iad7084
p_keyseat_side	32	7	20	point	1	iad7084
p_keyseat_side	33	7	20	point	1	iad7084
cylinder_side	29	21	5	circle	3	iad7084

****
** Geometric Tolerancing **
****

Figure 5.10 Inspection Plan for the Part Shown in Figure 5.8 (Cont.)

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
p_keyseat_side	30	7	20	distance	34
p_keyseat_side	31	7	20	distance	30
p_keyseat_side	32	7	20	distance	34
p_keyseat_side	33	7	20	distance	32
cylinder_side	29	21	5	diameter	
cylinder_side	29	21	5	position	0

Change Part Orientation

****
** Measurement **
****

Feature	No.	Dim.1	Dim.2	Element	Min.points	Probe
box_face	34	130	130	plane	3	iad7084
bore	35	30	20	circle	3	iad7084
bore	36	36	20	circle	3	iad7084
bore	37	25	40	circle	3	iad7084

****
** Geometric Tolerancing **
****

Feature	No.	Dim.1	Dim.2	Tolerancing	Datum
box_face	34	130	130	flatness	
box_face	34	130	130	parallelism	18
bore	35	30	20	diameter	
bore	36	36	20	diameter	
bore	37	25	40	diameter	
bore	35	30	20	position	0
bore	36	36	20	position	0
bore	37	25	40	position	0
bore	36	36	20	distance	35

Planning is finished

Figure 5.10 Inspection Plan for the Part Shown in Figure 5.8 (Cont.)

5.11 Discussions and Conclusions

The generative expert inspection task planner described in this chapter is developed to utilize domain specific knowledge and rules which generate inspection plans, for both rotational and prismatic parts, to be used for coordinate measuring machines. It is integrated with the FDDL and the feature-based modeler. It allows a user to input through the feature-based modeler, by editing the source language or by directly inputting the target language based on the FDDL syntax. However, the code generator creates input which is considered as syntax error free. The developed system has been implemented on a SUN 3/260 in PROLOG. The PROLOG language with its built-in search procedure, pattern matching and unification with backward chaining is ideally suited to this application. The current system consists of more than 72 frames to describe features and about 150 rules including some rules for manipulating data structures.

Some new ideas for automating inspection planning using AI techniques are presented and have been successfully implemented. These include feature accessibility, inspection datum features first and grouping inspection task according to geometric features, type of tolerances and measuring probe and datum features to improve the efficiency of part geometric and dimensional inspection. There are still many issues to be examined such as the need for an automated programming system to generate a program for the CMM based on the produced inspection plan, the use of machine vision to guide the inspection plan execution and to locate features in different orientations, and probe selection expertise to enlarge the scope of applications.

CHAPTER 6

KNOWLEDGE-BASED ASSEMBLY SEQUENCE PLANNER

6.1 Introduction

6.1.1 Background

Previous chapters have discussed the feature-based design description language, feature-based modeler, cellular manufacturing and inspection planning. The feature-based modeling using the FDDL can carry out not only individual components modeling, but also whole product modeling. After all individual components are designed, machined and inspected and/or purchased, products are assembled. This chapter deals with assembly sequence planning.

Recently, assembly robots have gained popularity in industry as a means to increase assembly productivity and reduce labour costs. Hence, much research is being conducted in robotic assembly and its related fields. In association with robotic assembly, robot programming systems at the task, object and manipulator level, are being developed (Dufay and Latombe, 1983), (Latombe and Mazer, 1981), (Lieberman and Wesley, 1977), (Mazer, 1983), (Popplestone and Ambler, 1983), (Rondeau and ElMaraghy, 1989) and (Tang and ElMaraghy, 1986). Low level assembly planning systems operate in the way that the human programmers specify the assembly sequence. Further increases in the automatic level of integrating product design with the assembly robots task planning and programming require the automatic generation of the assembly sequence.

If the structures of products are examined, it can be understood that product assembly is not a pattern matching process. That is to say, the correct assembly sequence may not be generated only by matching the relations between the parts. The whole product structure, the part's geometry and functions become important. Therefore, a successful assembly requires human assembly mechanics or engineers to carefully determine the sequence using their assembly knowledge. In this chapter, a knowledge-based approach is proposed and implemented in combination with question and answer techniques, to integrate with the FDDL system.

6.1.2 Existing Methods

As discussed in chapter 1, available methods are grouped into four classes; those that use a) mating conditions, b) disassembly, c) question and answer techniques, and d) a dedicated approach for manipulators.

Ko and Lee (1987) developed the mating conditions which were insufficient and they neglected to discuss the sequence among sub-assemblies. Since the interference checking is done by the question and answer technique, it can be conducted more efficiently. The geometric interference checking can be done using some geometric modelers. It is possible that the generated sequence of assembly is geometrically feasible but it is not practical. Also, they were not concerned with the representation of objects, the description in both geometric and technical characteristics, and the relations with other parts in a CAD system.

Disassembly, as the reverse of assembly, is mainly based on interference checking using simulation or employing question and answer receiving by a human user (Sekiguchi et al, 1983) and (Sedas and Talukdar, 1987). Logically, it is feasible; practically, it may not make a significant

difference in comparison with the direct assembly. The simulation technique can still be used for assembly instead of disassembly.

Both Bourault's (1984) and Defazio's and Whitney's (1987) techniques can generate all possible and valid sequences for the given assembly. Since only one sequence is required for the assembly, the best sequence must be chosen among those available. The example given by Defazio and Whitney (1987) shows that some generated assembly sequences are similar or slightly different. These differences are not significant at all. For a ball pen assembly example, they derived 12 choices for 6 parts and 5 relations. Another example, which is more complicated, has 11 parts and 18 relations. They discovered that there were 440 ways in which the product can be assembled. These 440 sequences have to be evaluated and the best has been chosen. This technique requires that the users be either assembly engineers or mechanics with knowledge of assembly. The information is required from assembly drawings and a relation network of assembly is given by the user. Extra effort must be made to choose the best assembly sequence among the choices available. Currently, this cannot be used to integrate computer-aided design and automated assembly.

For some dedicated approaches for robot assembly planning such as those by Chang and Wee (1988), it is clear that the sequence of assembly is implicitly specified already by the user. For example, when the component role is defined as main, it must first be assembled on the foundation. This is implied by the rules. If the part's roles are defined, the sequences have been determined accordingly. By this, the robot program or assembly plan for the robot can be generated. These are mainly concerned with robot assembly tasks planning and not sequences planning.

6.1.3 Proposed Approach

The assembly sequence planning is a complex problem and requires specialized knowledge. A knowledge-base approach is proposed for the assembly sequence planning to integrate this system with the FDDL system. Since current assembly knowledge is not adequate, a question and answer technique is used for ordering the sequence. However, the number of questions is minimized in comparison to Bourault's, and Defazio's and Whitney's methods.

6.2 Code Generator for Assembly Sequence Planning

The code generator creates the code for the assembly sequence planner based on a defined syntax. Before the syntax is defined, the information and format required for the assembly sequence planner has to be determined. The description of a product in the FDDL has been discussed previously, the information for the assembly sequence planner includes a part identifier, part type, features, dimensions, relations and related parts identifier. The syntax for the target language of assembly has been defined in Appendix B.

The code generator for assembly planning not only checks the syntax-semantics of language, but also guarantees the input for the assembly planning system with error free syntax conditions. Since the part type is not considered as an element of the language, the code generator asks the user to provide the type name for each defined part. The code generator scans the source (pre-processed) to prove and verify the syntax-semantics of the language. If it is error free, the target language is created, otherwise, an error is indicated. The other code generators used to process the language have been discussed in previous chapters. The

principles are similar. An example of format follows:

part ID part type [relation list] terminator

For a real example,

1 cap [bore, 12, pufit, 3] eee.

6.3 Development of Assembly Sequence Planner

6.3.1 System Structure

The system structure is shown in Figure 6.1. It is mainly composed of a knowledge base and control structure. The hardware used is a SUN 3/260 Workstation and the software environment is MPROLOG. The system consists of several modules. The modular programming allows the system to be easily modified and extended. The knowledge base includes information about features, relations, part roles and assembly principles. The features are used to describe the components. The relations are defined to describe how parts are physically related to one another. Part roles potentially indicate information about whether or not a part can be a base part in the assembly. The assembly principles, such as the assembly sequence determined by part functions, the determination of base parts, the forming of sub-assemblies, and the strategies for sequencing, are developed in this research. The control strategies use information in the knowledge base to create the assembly sequence plan, based upon the input product. The planning process is accompanied with symbolic data structure manipulation.

6.3.2 Knowledge Representation

As discussed in Chapter 5, a combination scheme of frame and production rules is developed to represent the assembly knowledge. For the product description, a frame representation is used to describe the object's attributes, such as part identifier, type, and relation list. Stored

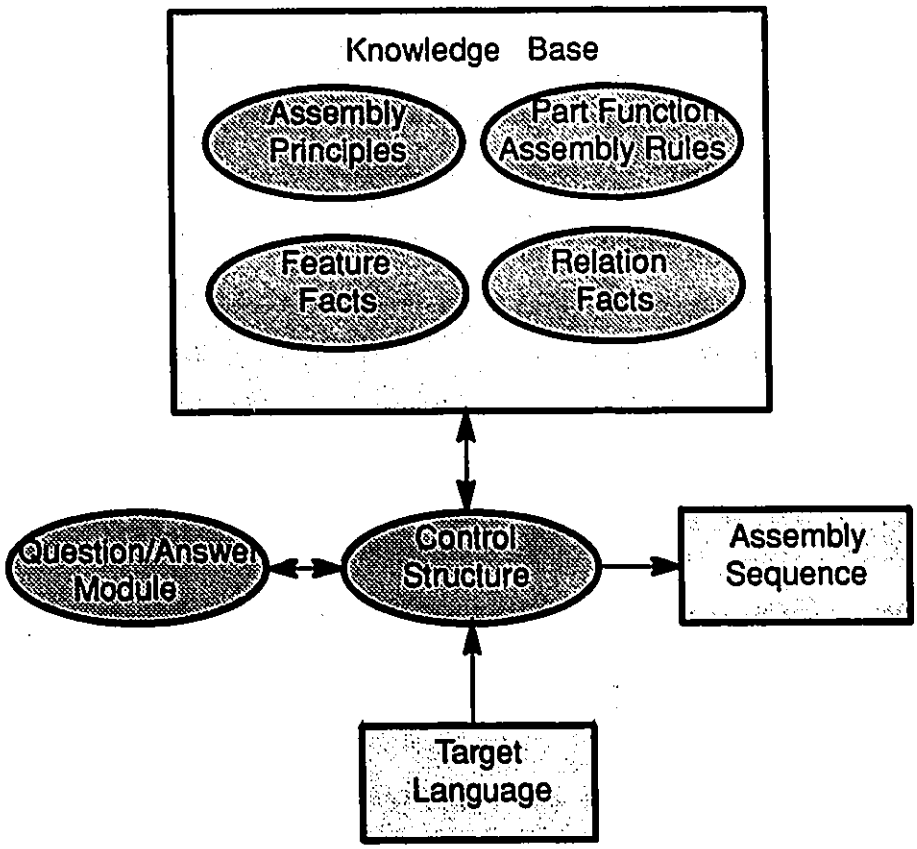


Figure 6.1 Assembly Sequence Planning System Structure

in the relation list, is the information regarding feature, dimension, relation and related part identifier.

6.3.3 Control Strategy

The control strategy uses the knowledge base information to infer the sequence for the sub-assemblies and final assembly. If all parts separated are defined as the initial state, the goal state is the completed product. Thus, the control strategy uses the knowledge and input product, to reason the goal state. The search algorithm is the depth-first, which is built-in the PROLOG language. The control strategies consist of following steps:

1. read the product in and arrange it in the data structure in which every component occupies a list and all parts lists are included in another list;
2. count all parts secured relations and find out base parts;
3. form sub-assemblies for each base part;
4. distribute all remaining parts into groups, in which these parts are related with some non-base parts;
5. order the sequence in the subassemblies;
6. ask the user to confirm the sequence. If yes go to 7; otherwise, go to the question and answer module;
7. order the sequence of sub-assemblies;
8. print all assembly sequence for the product.

6.3.4 Knowledge Base

In the knowledge base, the following knowledge is included: features and their properties, relations, possible base parts, impossible base parts, and so forth to describe the input component of the geometric and

physical relations of parts with other parts as well as their roles in the product structure.

(1). Relations defined in the research, associated with assembly properties, such as fit with pressure are secure assembly;

relation(pfit,secure).

relation(tfit,secure).

relation(contact_free,unsecured).

.

.

.

(2). Features and associated properties, such as bore are internal features;

feature(cylinder_side,cylinder,solid).

feature(bore, cylinder, empty).

feature(cylinder_front,plane,solid).

feature(box_face,plane,solid).

.

.

.

(3). Possible base parts known, such as shaft, housing, body, foundations, etc;

part(body).

part(housing).

part(shaft).

part(vessel).

.

.

.

(4). Impossible base part known, such as washer, screw, bearings, etc;

un_base([ID,NAME|T]) :- NAME=washer.

un_base([ID,NAME|T]) :- NAME=bearing_b.

un_base([ID,NAME|T]) :- NAME=screw.

.

.

.

Assembly principles knowledge regarding the part functions such as shaft class parts, housing class parts, bearings, screws, washers;

(5). For the shaft type of parts:

i. The planner first searches for the biggest diameter feature with a fitting relation;

ii. Determine if other parts have contact with this part;

iii. If yes,the part is put behind the above one in the list;

iv. If no, the part is searched to relate its adjacent feature on the shaft;

v. Go to 1 again until all features are examined.

(6). For the housing part:

i. The planner first looks for the minimum diameter of hole features to a relation to other part;

ii. Determine if other parts have contact with this part;

iii. If yes, the part is put behind the above one;

iv. If no, search the adjacent feature with relations;

v. Go to 1 until all features are examined.

(7). There are different kinds of bearings. For the sliding bearings, the planner should know which should be assembled on the body, i.e. first it is assembled with its outside cylindrical surface, then a shaft is mounted inside its hole. For thrust bearings, assembly takes place on the shaft immediately after the parts with which they have contact. For example:

IF a part is a sliding bearing,

THEN it should be assembled on the body, not on the shaft.

Washers should be assembled between the parts with which they contact.

IF a part is washer,

THEN it should go between the parts with which it makes contact.

Screws should be assembled once the parts to be fixed are assembled.

(8). knowledge of forming the sub-assembly and associated base parts;

i. Count all the parts with secured relations and order them in the order of decent;

ii. Find out that which is maximum, and form the sub-assembly; follow this procedure until all possible bases are done;

iii. find groups for the remaining parts.

6.3.5 Formation and Ordering of Sub-assemblies

Each part may have different types and numbers of relations. These relations may have different properties such as secured and unsecured relations. Some parts may have more secured relations than others and these parts are temporarily defined as base parts to group the parts

together. In essence, this means that in the assembly process, there are more parts to be assembled on these parts with secured conditions. Thus, it is required that these relations should be calculated to determine the bases. Due to the fact that the relations belong to individual features, it is possible that more than one of the features may have different relations with a same part. For example, a shaft may exist that consists of several cylinders and other features where one cylinder has a fitting relation with a bearing hole and the face of another adjacent cylinder contacts the bearing face. In such a case, the relation count for this shaft is one. Thus, when the relations are counted in order to form sub-assemblies, only the different parts are concerned. In order to compare the relations, a list for each part is created to store the related part identifiers.

To create a list, in which all relations are counted, it is preferable to arrange them in descending order. Once the order is decided, the corresponding full list of parts must be sorted and associated with that decided order. This can be done by identifying the number of relations for each part. The reference list includes the relation number and identifier. Once this is done, the parts list is re-arranged again based on the reference list order.

After the two lists in the same order are formed and arranged, the formation of the sub-assemblies should follow. This is done in a recursive manner whereby the first part in the list is analyzed to identify whether the part belongs to the non-base part category (such as bearings, washers, screws). If the part can be thought of as a base, the corresponding sub-assembly is formed. A list of relations accompanies the base part in which component identifiers are included. These identifiers are, in turn, used to

access the part lists to extract the part sub-lists and the sub-assembly list is formed for the base. The data structure is as follows:

[[base part list], [part list],

[part list], ...,]

For every base, a list is created in which all related parts are included. It is very common that a base part be a member of another sub-assembly at a higher level. In such situations, it must be decided which part is a base part, taking into consideration its possible status in the higher level sub-assembly or the final assembly. Also, some interactions between the parts and bases are involved. Some rules or help from the user are required to determine the sub-assemblies for cases in which two base parts have their own groups and are also members of each other. Thus, once the case is decided, the part is grouped into the other sub-assemblies and will have a special symbol to distinguish the situation for plan generation later. An example is shown in Figure 6.2, if part A is a base of W group, and B is the base part of Y group, at the same time, then A is a member of Y group and B is the member of W group too. When processing, the system first picks up the B as a member of W. Later, the system realizes that the A can also be member of Y. If there is confirmation that B is member of W, then the system creates a special list [part ID, base] to represent the absence of B in the Y group. For some situations where the part, such as a bearing, may be a member in two sub-assemblies, the system will decide its membership by using assembly knowledge. For example, a sliding bearing should be grouped into the body group, rather than the shaft group. For this case, the system will use a special mark to indicate that the part is absent until it is assembled onto another part. For some

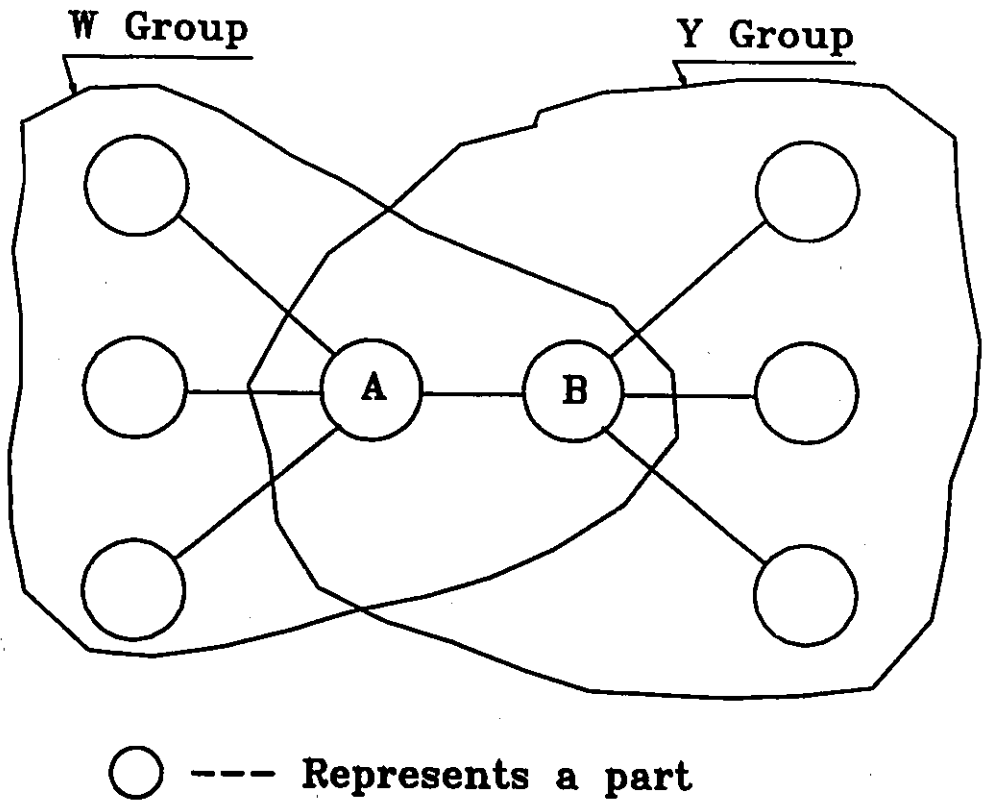


Figure 6.2 An Example for Determining Sub-assembly

situations, there may be some parts which do not relate to the base part directly. These must be sent to groups in which some members have relations with these remaining parts. Finally, we get all sub-assemblies with a different level of assembly.

The ordering assembly sequence for those inside the sub-assemblies is performed by using the assembly knowledge described above. If the assembly knowledge does not apply to a particular sequencing, a dimension checking algorithm is used to order the sequences. This algorithm compares the dimensions of the fitting features in a sub-assembly and their sequences are determined based on the order of fitting dimensions from small to large. In this way feasible sequences may be generated.

6.3.8 Question and Answer Techniques

For each sub-assembly, the assembly principles in the knowledge base and geometric dimensions checking are used to order the sequences. These strategies may not be enough to cover all parts and assembly situations. At present the sequencing results require a confirmation of the interference simulation. This confirmation is currently conducted by the user, instead of simulating interference checking. It is possible that the sequence is not feasible. In such cases, the question and answer module is used to determine the sequence. This technique, when compared with the Defazio and Whitney simplified method, is further simplified. The system shows the user the base part and all the related group members, and gives the user the input data structure which is very simple. For example, the part chosen is:

[part ID, name,[relation list]]

The system asks the user to input all the parts which must be assembled

before this part. All parts are shown as follows:

[[part ID, name,[relation list], [part ID, name,...],...]

If the user inputs only one part, the system will check the number of the remaining parts, to determine if there is more than one part left. The system continues this process which is summarized here:

i. display the base part for the group;

ii. if the number of parts are more than one, go to iii; otherwise, terminate;

iii. pick the first part in this group;

vi. ask the user to input all parts that must be assembled before this part is done;

iv. If input is more than one part, go to iii; otherwise, go to ii.

The worst case is when all sequences are not feasible for every sub-assembly. In such a case, the number of questions for the whole product is the maximum:

$$\Sigma (N_i - 1) \tag{6.1}$$

where i is the ith sub-assembly and N_i is the total number of parts in the ith subassembly. In the simplest case (worst case), the number of parts equals the number of relations. In comparison to Defazio's and Whitney's technique, the number of questions saved is as follows:

$$M = 2N - \Sigma(N_i - 1) \tag{6.2}$$

$i = 1, M = 2N - (N-1) = N + 1$

$i = 2, M = 2N - (N_1 - 1) + (N_2 - 1) = N + 2$

where $N_1 + N_2 = N$

$i = 3, M = N + 3$

$$i = p, M = N + p \tag{6.3}$$

Where M is the number of saved questions and P is the number of sub-assemblies in an assembly. Thus, when the sub-assembly number P and part number N are large, this method can save a large number of questions and answers.

In the sub-assembly process, every group is considered as a sub-assembly. Thus it is necessary to decide the level of the sub-assemblies such that the system can tell which sub-assembly must be done before others and so that the final sub-assembly will be conducted last. It is understood that some parts in the hierarchical structure are located at the same level as other sub-assemblies meaning these parts are assembled with some sub-assemblies. Thus, they have to wait until other sub-assemblies are done. This situation is treated by following the approach:

i. check whether the sub-assembly planning is done. If yes,terminate, otherwise, go to ii;

ii look for the subassembly which has all its members, i.e. all members in this group are present. If it is true, go to iv; otherwise, continue;

iii. if there is no such sub-assembly, find one which contains the parts in which there are members of other sub-assemblies and go to iv;

iv. generate the plan for the subassembly;

6.4 A Case Study

In this case study, three examples are chosen to demonstrate the described ideas and the assembly planning system.

6.4.1 Flash Light Assembly

The flash light shown in Figure 6.3 is composed of nine parts. This

example is taken from Sedas and Talukdar (1987). They are described with FDDL shown in Appendix E. This source language is first processed by the syntax checking phase of the language system. The code generator of the assembly creates the input for the assembly planning system, which is shown Figure 6.4. The planning system first reads in the description and forms an initial list. By computing all secure relations, the system forms three sub-assemblies with three corresponding base parts, those being the reflector, bottom_cap and lens_cap. The reflector group consists of the reflector itself, bulb, lens_cap and lens. The lens_cap group consists of itself and the case. The bottom_cap group contains the bottom_cap, case and spring. Since the two batteries are not directly related to the base parts, they are omitted. After examining the relation of the batteries, the system picks the battery with number 5 and puts it in the group of the bottom_cap, because the battery is related to the spring. Then, the other battery is sent to this group too, the reason is that it contacts the battery which exists in the group. The next step of planning is to order the sequence inside of each sub-assembly. Due to the current system not having specific knowledge about the bottom_cap, spring or battery, the main effective rules are the dimension checking which is based on the related feature dimensions, and the relations of the sequence. For example, the spring and case are directly assembled on the bottom_cap, comparing their dimensions, the system figures out the spring goes first, then the case is assembled. Since the spring is assembled, the battery having a relation with the spring is assembled before that battery without relation with spring. During the inference, the system prints out the sequence for user confirmation:

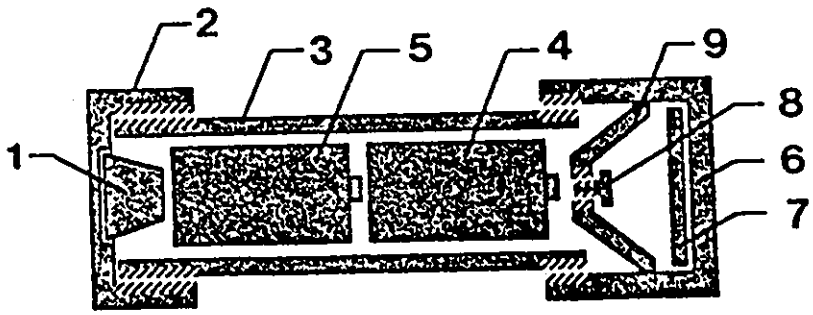


Figure 6.3 Flash Light Assembly (Sedas and Talukdar, 1987)

part01 spring [cylinder_side,12.0e0,pufit,part02,t_cone_front,10.0e0,contact_free,part05] eee.
part02 bottom_cap [h_thread_nf_side,32.0e0,screw_fix,part03,bore,12.0e0,pfit,part01] eee.
part03 case
[thread_nf_side,32.0e0,screw_fix,part02,thread_nf_side,32.0e0,screw_fix,part06] eee.
part04 battery [cylinder_front,20.0e0,contact_free,part05] eee.
part05 battery
[cylinder_front,20.0e0,contact_free,part01,cylinder_front,5.0e0,contact_free,part04] eee.
part06 lens_cap [h_thread_nf_side,32.0e0,screw_fix,part03,bore,32.0e0,pfit,part09] eee.
part07 lens [cylinder_front,30.0e0,contact_free,part09] eee.
part08 bulb [thread_nf_side,6.0e0,screw_fix,part09] eee.
part09 reflector
[cylinder_side,33.0e0,pfit,part06,cylinder_front,33.0e0,contact_free,part07,h_thread_nf_side,6.0e0,screw_fix,part08] eee.
stop input eee.

Figure 6.4 Target Language of the Flash Light Shown in Figure 6.3

Base Part ID	Base Part
part02	bottom_cap

Part ID	Part Name
part01	spring
part03	case
part05	battery
part04	battery

Is this sequence correct y/n?

After the user confirms, the system reasons about another group. For the lens_cap group, it has only the part case, which is currently in another group. The ordering is simple. The reflector group sequencing is also based on the dimension checking:

Base Part ID	Base Part
part09	reflector

Part ID	Part Name
part08	bulb
part07	lens
part06	lens_cap

Is this sequence correct y/n ?

When all individual groups are completed, the system infers a final plan. Since the part case is in the group of the bottom_cap, the lens_cap group must wait for the group which has the case done first. Thus, the

group bottom_cap is planned first, because it has complete by virtue of having all its members, i.e. no member is in other group. Then, the reflector group is done for the same reason. Finally, the lens_cap group is planned. The results are shown in the Figure 6.5. For this particular example, if the assembly sequence is carried out using question and answer module, compared with Defazio and Whitney (1987) the number of questions saved is at least:

$$M = N + p = 9 + 3 = 12$$

6.4.2 Ball Point Pen Assembly

This example is chosen from Defazio and Whitney (1987). It consists of 6 parts. The relations are defined and shown in the Figure 6.6. The part cap has only one relation with the body. The head is fitted onto the body and the tube is put onto the head, thus it has two relations with the body and tube. The body has three relations with the cap, head and button. The part tube has two relations with the head and ink which is assumed as a solid part. The button, however, has only one relation with the body. The input to the assembly sequence planning system is shown in Figure 6.7. The system first calculates the relations number and determines that the body can be the base. Also, by computation, the head is chosen as another base. Then the sub-assemblies are formed for these two base parts. The body group members are the cap, button and head. The head group consists of the head and tube. The remaining part, the ink, has a relation with the tube, therefore, it is grouped into the head group too. Since the head is an absent member of the body group, the head group is planned first. Then the body group is planned. As above, the knowledge base does not have specific knowledge for the pen. The dimension and

Assembly Sequence for Product

Part Name	ID	Feature	Relation	Mate Part	ID
spring	part01	cylinder_side	pufit	bottom_cap	part02
case	part03	thread_nf_side	screw_fix	bottom_cap	part02
battery	part05	cylinder_front	contact_free	spring	part01
battery	part04	cylinder_front	contact_free	battery	part05
bulb	part08	thread_nf_side	screw_fix	reflector	part09
lens	part07	cylinder_front	contact_free	reflector	part09
lens_cap	part06	bore	pfit	reflector	part09
case	part03	thread_nf_side	screw_fix	lens_cap	part06

The assembly sequence planning is finished

Figure 6.5 Assembly Sequence Plan of the Flash Light

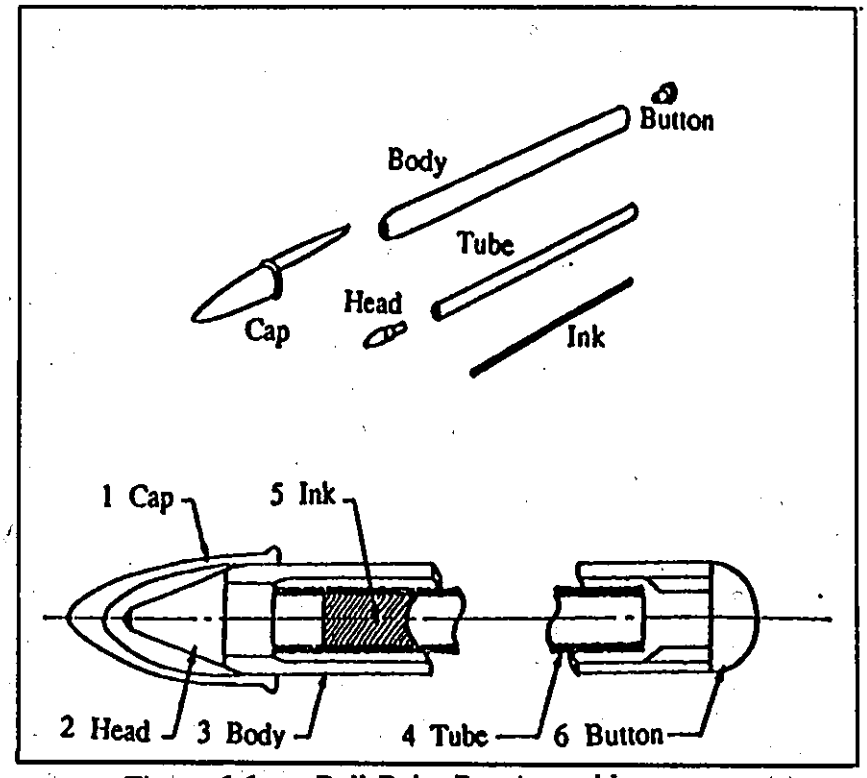


Figure 6.6 Ball-Point Pen Assembly
(Dafazio and Whitney, 1987)

relation reasoning is applied to infer the plan and is shown in Figure 6.8. Also, if the assembly sequence is planned using the question and answer module, compared with Defazio and Whitney (1987) the number of questions saved is at least:

$$M = N + p = 6 + 2 = 8$$

6.4.3 DC Motor Assembly

A DC motor shown in Figure 6.9 was redesigned for ease of assembly by ElMaraghy et al (1988). Made of 14 parts, its input is given in Figure 6.10. In this target language, the parts relations are defined clearly. These relations are associated with the parts geometry to determine the sub-assemblies. Firstly, the assembly planning system computes the secured relations for each part. Based on the calculation and knowledge in knowledge base, the three base parts are chosen as temporary bases for formation of sub-assemblies. These are the housing (7 relations), end_cap (7 relations) and armature (2 relations). The corresponding three sub-assemblies are formed. The armature sub-assembly consists of the armature and two ball bearings. The housing group is composed of two magnets and three brackets. The end_cap group has the remaining parts, plus one bearing in the armature group and the housing in the housing group. The housing part is a base and the corresponding knowledge has been developed. For this case, it uses the knowledge to infer the sequence. As with the armature, the sequence of the two bearings is not important, and the reasoning results are also given in their natural sequence. The final group is the end_cap. The specific knowledge about the end_cap is not included in the knowledge base, so the dimension checking regarding the fitting feature dimensions and relations is applied and the final results are

```
1 cap [bore,12,pufit,3] eee.
2 head [cylinder_side,8,pfit,3,cylinder_side,6,pufit,4] eee.
3 body [bore,8,pfit,2,cylinder_side,12,pufit,1,bore,9,pufit,6] eee.
4 tube [bore,6,pufit,2,bore,6,lfit,5] eee.
5 ink [cylinder_side,6,lfit,4] eee.
6 button [cylinder_side,9,pufit,3] eee.
stop input eee.
```

Figure 6.7 Input Target Language of the Pen Shown in Figure 6.5

Assembly Sequence for Product

Part Name	ID	Feature	Relation	Mate Part	ID
tube	4	bore	pufit	head	2
ink	5	cylinder_side	lfit	tube	4
head	2	cylinder_side	pfit	body	3
button	6	cylinder_side	pufit	body	3
cap	1	bore	pufit	body	3

The assembly sequence planning is finished

Figure 6.8 Assembly Sequence Plan of the Pen Shown in Figure 6.6

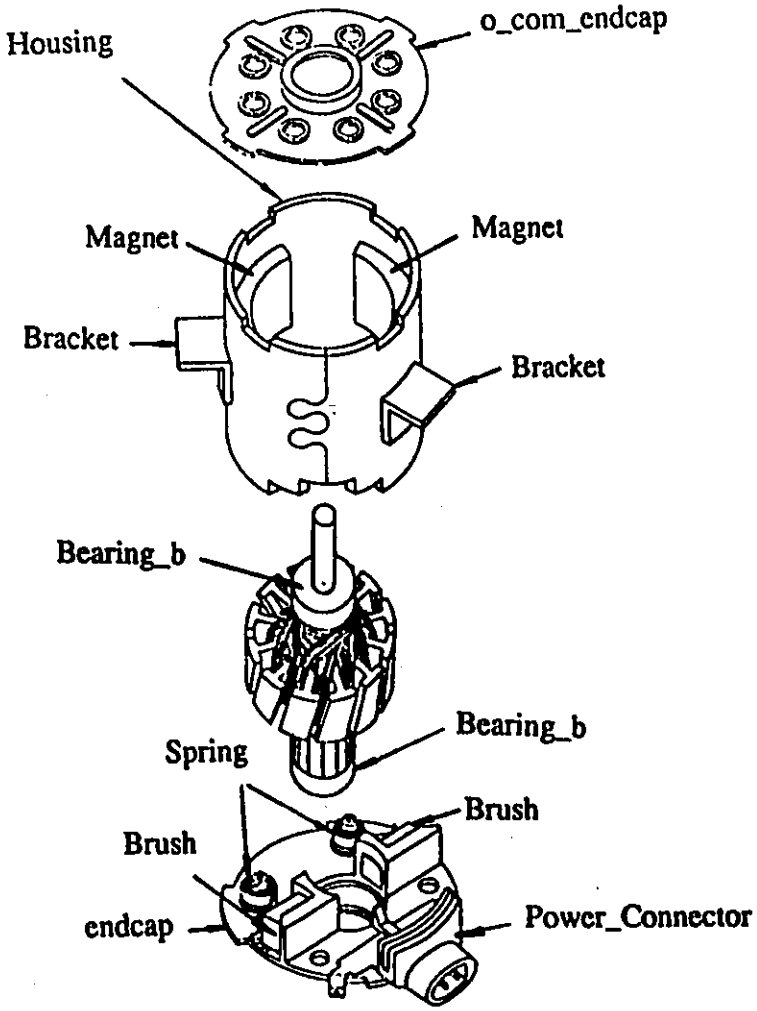


Figure 6.9 D.C. Motor Assembly (ElMaraghy et al., 1988)

```
1 housing
[cylinder_side,50,weld,2,cylinder_side,50,weld,3,cylinder_side,50,weld,4,bore,40,cure,5,
bore,40,cure,6,bore,40,pufit,7,cylinder_front,50,contact,8] eee.
2 bracket [bore,50,weld,1] eee.
3 bracket [bore,50,weld,1] eee.
4 bracket [bore,50,weld,1] eee.
5 magnet [cylinder_side,40,cure,1] eee.
6 magnet [cylinder_side,40,cure,1] eee.
7 end_cap
[cylinder_side,40,pufit,1,bore,30,pfit,10,squre_hole,8,mfit,12,squre_hole,5,mfit,13,squre
_hole,5,mfit,14,cylinder_side,4,pufit,15,cylinder_side,4,pufit,16] eee.
8 o_com_endcap [bore,30,pufit,11,cylinder_front,50,contact_free,1] eee.
9 armature [cylinder_side,10,pfit,10,cylinder_side,10,pfit,11] eee.
10 bearing_b [bore,10,pfit,9,cylinder_side,30, pfit,7] eee.
11 bearing_b [bore,10,pfit,9,cylinder_side,30, pfit,8] eee.
12 power_connector [box,8,mfit,7] eee.
13 brush [box,5,mfit,7] eee.
14 brush [box,5,mfit,7] eee.
15 spring [bore,4,pufit,7] eee.
16 spring [bore,4,pufit,7] eee.
stop input eee.
```

Figure 6.10 Input Target Language of the D.C. Motor Shown in Figure 6.9

correct. Since the housing is also a part in the end_cap sub-assembly and non of its members are absent, its group is planned first. Then the armature's members are available, and the planning can be carried out. Finally, the end_cap group is planned because all sub-assemblies are done. Whole process is provide as follows:

S(System): Is the part with ID 1 assembled on the part (ID) 7 ? y/n

U(User): y (because housing is assembled on end_cap.)

S: Please indicate the following base part

[7,end_cap,[cylinder_side,40,pufit,1,bore, ...,]]

is suitable to assemble on the part with ID 1 ? y/n

U: n (because user has answered part 1 is assembled on 7.)

S: Please indicate the following part

[10,bearing_b,[bore,10,pfit,9, ...,]]

is suitable to assemble on the part with ID 9 ? y/n

U: y (because bearing should be assembled on armature.)

S: The system thinks the part end_cap is a base. Do you agree? y/n

U: y

S: Please find out which class this base part belongs:

Shaft

Housing

Body

U: none

System goes to dimension checking module and infers:

S: Base Part ID Base Part

7 end_cap

Part ID Part Name

15 spring

16 spring

13 brush

14 brush

12 power_connector

10 bearing_b

1 housing

8 o_com_endcap

Is this sequence correct? y/n

U: y

S: The system thinks the part armature is a base. Do you agree? y/n

U: y

S: Please find out which class this base part belongs:

Shaft

Housing

Body

U: shaft

The system infers the sequence using knowledge of shaft parts.

S: Base Part ID Base Part

9 armature

Part ID Part Name

10 bearing_b

11 bearing_b

Is this sequence correct? y/n

S: y

S: The system thinks the part base(housing) is a base.

Do you agree? y/n

U: y

S: Please find out which class this base part belongs:

Shaft

Housing

Body

U: housing

The system infers the sequence using knowledge of housing parts.

S:

Base Part ID	Base Part
1	base
Part ID	**Part Name**
5	magnet
6	magnet
2	bracket
3	bracket
4	bracket

Is this sequence correct? y/n

S: y

Then the system reasons the final plan and the final result is given in Figure 6.11. Again, if the assembly sequence is planned using the question and answer module, compared with Defazio and Whitney (1987) the number of questions saved is at least:

$$M = N + p = 14 + 3 = 17$$

Assembly Sequence for Product

Part Name	ID	Feature	Relation	Mate Part	ID
magnet	5	cylinder_side	cure	housing	1
magnet	6	cylinder_side	cure	housing	1
bracket	2	bore	weld	housing	1
bracket	3	bore	weld	housing	1
bracket	4	bore	weld	housing	1
bearing_b	10	bore	pfit	armature	9
bearing_b	11	bore	pfit	armature	9
spring	15	bore	pufit	end_cap	7
spring	16	bore	pufit	end_cap	7
brush	13	box	mfit	end_cap	7
brush	14	box	mfit	end_cap	7
power_connector	12	box	mfit	end_cap	7
bearing_b	10	cylinder_side	pfit	end_cap	7
housing	1	bore	pufit	end_cap	7
o_com_endcap	8	bore	pufit	bearing_b	11

The assembly sequence planning is finished

Figure 6.11 Assembly Sequence Plan of the Motor Shown in Figure 6.9

6.5 Conclusions

In this chapter, an integration scheme of parts modeling with FDDL and assembly sequence planning is proposed. The FDDL can be used to model the product, and the corresponding language processing system is developed to create a target language (input) for the semi-automated assembly planning system. The developed knowledge-base planning system uses a combination scheme of knowledge representation, frames and production rules to represent the product and the assembly knowledge. The developed assembly knowledge can be generally useful for other assembly planning systems. As a means of assisting the planning system, the question and answer technique is proposed and implemented in the system. This produces considerable savings in effort when compared with existing techniques. Three examples show the feasibility.

Obviously, the assembly sequence planning is a complex problem, and the knowledge base in the developed planning module is not adequate. In this chapter, only the ideas and the approach were presented to show that the FDDL can be used for whole product assembly planning. In order to make the system useful and the knowledge acquisition process easier, explanation-based machine learning techniques (Segre, 1988) should be associated with the question and answer process. Therefore, whenever the user inputs the sequence for an assembly, the corresponding rules are reasoned and placed in the knowledge base.

CHAPTER 7

CONCLUSIONS

7.1 Introduction

The main objective of this thesis is to explore new approaches and knowledge for the integration and individual aspects of feature-based modeling and various manufacturing tasks planning. An analysis of the traditional and computer-aided design and process planning has been carried out and important gaps between CAD systems and the automated process planning systems have been identified. These gaps are due to the lack in a uniform representation scheme of the parts and products, and the absence of an effective communication medium for design and process planning systems. The language method has been proposed and implemented to precipitate communication between feature-based modeling and manufacturing tasks planning, in association with a feature representation. Some artificial intelligence techniques, such as expert systems, pattern recognition and languages, have been applied to the integration of feature-based modeling, cellular manufacturing planning, inspection planning, and assembly sequence planning. Also, new knowledge rules have been developed for the feature-based cellular manufacturing planning, feature-based inspection planning and assembly sequence planning.

7.2 Conclusions

The original contributions to the field of integration of CAD and manufacturing tasks planning are summarized as follows:

1. Most research efforts in the past have been concerned with the individual aspect of computer-aided geometric modeling, process planning, expert systems for process planning, and assembly sequence generation. Thus, the individual application results in the use of a very restricted representation method and incomplete information for part descriptions. These simplified methods ignore essential information that is required by other processes, and the difficulties of information exchange that exist amongst the various systems. The presented research effort takes into consideration several aspects ranging from feature-based modeling to parts assignment, inspection and assembly planning. The results show that this approach is feasible and can be used to increase the level of automation.

2. A feature-based modeling and manufacturing tasks planning system has been designed and implemented, which includes a prototype feature-based modeler, a design description language system, cellular manufacturing planning, inspection planning and assembly sequence planning systems. The significant point is that the feasibility of true integration of CAD and manufacturing tasks planning has been demonstrated. Also, the traditional process planning concept is extended into manufacturing tasks planning to include parts assignment planning, inspection and assembly planning. Uniform feature representation of parts and products has made it possible for a design to be integrated with machining, inspection and assembly, aspects in which different information is required.

3. Another significant contribution is that a new language, FDDL, has been proposed and designed. Its syntax, semantics and

vocabulary are defined with considerations made to the human user, engineering representation compatibility, and the implementation in computer software. The FDDL system has been developed consisting of a number of lexical analyzers, parser and code generators. A utility lexical analyzer has also been developed to assist the user by indicating the syntax errors. This language system can be input either directly using text file or through a feature-based modeler. Once the products or parts are modeled by the FDDL or by the feature-based modeler, inputs that are syntax error free are created for the manufacturing tasks planning systems using the FDDL system.

4. The geometric modelers currently available cannot drive manufacturing application systems such as process planning systems. A prototype of a feature-based modeler has been developed. It includes two separate sets of data for the display and driving manufacturing application systems. An expert tolerancing consultant module has been developed to assist the user in decision making. The hierarchical knowledge base structure and the control algorithm and the knowledge base have been separated in the tolerancing module. The features defined in the modeler can be represented both by a geometric modeler and the FDDL, and can also be used by the manufacturing tasks planning systems. Thus, the feature representation can bridge the gap between the geometric modelers and the application systems.

5. Cluster-seeking algorithms and optimization techniques have been proposed to design machine cells and to form part families. Three algorithms have been implemented in this research. When compared with previous works, the results show that the combination of the Isodata

algorithm and optimization techniques can provide different configurations for a given production by manipulating the parameters to form desired cells. Thus, this flexible method easily adapts to real cellular manufacturing systems. Based on the formed cells and part families, the Feature-Based Assignment System has been developed to integrate the design and cellular manufacturing. Knowledge regarding the features and possible tolerance specifications and surface finish has been developed in the form of production rules. A decomposition and combination technique is used to generate possible machining routes. The Automata technique is used for developing a finite-state acceptor for filtering these routes. Decision functions are formulated based on the combination of pattern recognition techniques and manufacturing knowledge for the assignment of parts to the appropriate cells. A new part modeled by the FDDL or Feature-Based Modeler can be sent directly to an appropriate cell using this system.

6. As an important component of the CIM and FMS systems, CMM has not been thoroughly investigated in terms of operational efficiency, its integration with CAD and automated inspection task planning. The traditional inspection process has been systematically analyzed to develop some fundamental principles. A study of CMM characteristics, tolerancing theory, features representation, part structure and geometry has also been performed and hence, new inspection strategies and knowledge have been created. The relationships between the features with all possible tolerances and geometric elements used by CMM have been defined. A set of original inspection strategies has been created and implemented. These include the determination of feature accessibilities,

datum features inspection, the grouping of all accessible features which can be inspected as similar geometric elements, the inspecting of all features before changing probes or their orientation as well as part orientation, and the checking of similar tolerance types together and the tolerance measurement of all related features before changing a datum. A combined knowledge representation scheme of the production rule and frame has been used in the planner. A part in the initial state of planning is included in a super list in which each feature occupies the list as a frame with a varying number of slots. The goal state of planning is an empty list. Thus, elegant symbolic data manipulating techniques are required to conduct the planning process. Since the inspection of a new component designed by the FDDL or Feature-Based Modeler can be planned, this planner is of a generative system in nature. This planner is also integrated with the FDDL and Feature-Based Modeler.

7. Automated robot assembly tasks planning still requires a programmer to specify the assembly sequence. A knowledge-based approach has been proposed to integrate the CAD and assembly task planning. A prototype of this assembly sequence planner has been developed for generating the assembly sequence for products directly from the design. An assembly is decomposed into a number of sub-assemblies and for each sub-assembly a base part is identified. Assembly principles and part function knowledge are developed to sequence the sub-assemblies and the final assembly. These assembly principles along with other knowledge have been implemented in this system. Obviously, the assembly knowledge collection takes time and the implemented assembly rules are limited. As a supplement for the current system, a question and answer module is

designed to ask a minimum of number of questions if the assembly sequence produced by the planner is not feasible. Applications show that the FDDL can be used not only for single component tasks planning, but also for whole product assembly.

7.3 Suggestions for Future Research

Obviously, it is impossible for a single thesis to tackle all aspects of this broad multi-faceted topic in depth. The development of feature-based modeling, cellular manufacturing planning, inspection planning and assembly sequence planning requires continuous research effort. The following are suggestions to be used for future research:

1. Develop a design environment integrated with a commercial solid modeler. This environment would allow the user to describe directly the designed product with visual aids. In order to make the modeler useful, it is desirable that a facility be provided for users to define features, and for the system to learn from its user. Also, this environment should be designed to be integrated with the FDDL system and to be easily extended to include expert systems for design in the future.

2. Further improve and extend the FDDL so that it can be used in industry.

3. The part assignment system can be further integrated with a detailed process planning system and scheduling module for real cellular manufacturing production.

4. Based on the developed inspection planner and prototype feature-based modeler, an automatic programming system can be developed for CMM. This programming system can be further integrated with the inspection planner, feature-based modeler, a vision system, and a loading

unloading robot to form an inspection cell.

5. New knowledge rules need to be developed to complete the assembly sequence planner. Also, machine learning techniques should be applied to the question and answer module for the assembly sequence planner in order that some useful rules be deduced.

REFERENCES

Aho, A.V., Sethi, R. and Ullman, J.D., "Compilers Principles, Techniques, and Tools", Addison-Wesley Publishing Company, 1985.

Baer, A, Eastman, A.B. and Henrion, M., 1979, "Geometric Modeling: A Survey", Computer-Aided Design, Vol. 11, No. 5, Sept, pp. 253-272.

Berenji, H.R. and Khoshnevis, B., 1986, "Use of Artificial Intelligence in Automated Process Planning", Computers in Mechanical Engineering, September, pp. 47-55.

Bound, A.H. and Chang, K.J., 1988, "Feature-Based Process Planning for Machined Parts", Proc. of the 1988 ASME International Computers in Engineering Conference and Exhibition, August, pp. 571-576.

Bourjault, A., 1984, "Contribution a Une Approche methodologique de l'assemblage automatise: Elaboration automatique des sequences operatiores," Thesis to obtain Grade de Docteur des Sciences Physiques at L'Universite de Franche-Comte, November.

Brown, C.M. and Voelcker, H.B., "The PADL Project", Proc. of 7th NSF Conference, Production Research and Technology, Sept. 1979.

Burbidge, J.L., 1971, "Production Flow Analysis", The Production Engineer, April/May issue, pp. 139-152.

Chan, H.M. and Milner, D.A., 1982, "Direct Clustering Algorithm for Group Formation in Cellular Manufacturing", Journal of Manufacturing System, Volume 1, Number 1, pp. 65-74.

Chang, T.C., Anderson, D.C. and Mitchell, O.R., 1988, "QTC - An

Integrated Design/Manufacturing/Inspection System for Prismatic Parts", Proc. of the 1988 ASME International Computers in Engineering Conference and Exhibition, August, pp. 417-426.

Chang, K.H. and Wee, W.G., 1988, "A Knowledge-Based Planning System for Mechanical Assembly Using Robots", IEEE Expert, spring, Volume 3 Number 1, pp. 18-30.

Chang, T.C. and Wysk, R.A., 1983, "CAD/Generative Process Planning with TIPPS", Journal of Manufacturing Systems, Volume 2 No. 2, pp. 127-135.

Cunningham, J.J. and Dixon, J.R., 1988, "Designing with Features: The Origin of Features", Proc. of the 1988 ASME International Computers in Engineering Conference and Exhibition, August, pp. 237-243.

Cutkosky, M.R., Tenenbaum, J.M. and Muller, D., "Features in Process-Based Design", Proc. of the 1988 ASME International Computers in Engineering Conference and Exhibition, August, pp. 557-562.

Davies, B.J. and Darbyshire, I.L., 1984, "The Use of Expert Systems in Process-Planning", Annals of the CIRP Vol. 33/1, pp. 303-306.

Defazio, T.L. and Whitney, D., 1987, "Simplified Generation of All Mechanical Assembly Sequences", IEEE Journal of Robotics and Automation, Vol. RA-3, No. 6, pp. 640-658.

Descotte, Y. and Latombe, J.C., 1984, "GARI: An Expert System for Process Planning", Solid Modeling by Computers, Edited by Pickett, S. and Boyse, J. W. Plenum Press, N.Y. pp. 329-345.

Dixon, J.R., 1988, "Designing with Features: Building Manufacturing Knowledge into More Intelligent CAD Systems", Proc. of Manufacturing International'88, Vol.1, pp 51-57.

Dufay B. and Latombe J.C., 1983, "An approach to automatic robot

programming based on inductive learning", The Int. J. of Rob. Res., Vol.3, No. 4, pp. 3-20.

ElMaraghy, H.A. and Gu, P.H., 1988, "Knowledge-Based Assignment of Parts to Machine Cells", Int. J. of Advanced Manufacturing Technology, 3(1), pp. 34-44.

ElMaraghy, H.A., Knoll, L., Johns, B., and Pavlik, M., 1988, "Design for Assembly of a Direct Current Motor", Proc. of 9th International Conference on Assembly Automation, Seminar Notes Product Design for Assembly, Volume IFS Pub, March 15-17, London, U.K. pp. 73-82.

ElMaraghy, H.A. and Gu, P.H., 1987, "Expert System for Inspection Planning", Annals of the CIRP, 37/1, pp. 85-89.

Eversheim, W. and Fuchs, H., 1980, "Integrated Generation of Drawings, Process Plans and NC-tapes", Advanced Manufacturing Technology, Black, P. ed., IFIP Conference, North Holland.

Foster, L.W., 1986, " Geo-METRICS ", Addison-Wesley Publishing Company, INC.

Fu, K.S., 1982, "Syntactical Pattern Recognition: Theory and Applications", Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Gu, P.H. and ElMaraghy, H.A., 1988, " Expert Tolerancing for Geometric Modeling ", Proc. of ASME Manufacturing Int.'88 Conf., Vol. 1, pp. 17-22.

Gu, P.H. and ElMaraghy, H.A., 1989, "Formation of Manufacturing Cells by Cluster-Seeking Algorithms", Journal of Mechanical Working Technology, Vol. 20, pp. 403-413, Elsvier Press, Amsterdam, Netherlands.

Gu, P.H., ElMaraghy, H.A. and Hamid, L., 1989, "FDDL: Feature-Based Design Description Language", Proc. of First Int. ASME Conf. on Design Theory and Methodology, September, Montreal, Canada.

Han, C. and Ham, I., 1986, "Multiobjective Cluster Analysis for Part Family Formations", Journal of Manufacturing Systems, Volume 5, Number 4, pp. 223-230.

Inui, M., Suzuki, H., Kimura, F. and Sata, T., 1987, "Existing Process Planning Capacities with Dynamic Manipulation of Product Models", 19th CIRP International Seminar on Manufacturing Systems Computer-Aided Process Planning, Penn State, U.S.A. June, pp. 273-280.

Joshi, S.B. and Chang, T.C., 1987, "CAD Interface for Automated Process Planning", 19th CIRP International Seminar on Manufacturing Systems Computer-Aided Process Planning, Penn State, U.S.A. June, pp. 39-45.

Kimura, F., 1984, "GEOMAP-III, Designing Solid with Free-Form Surfaces", IEEE Computer Graphics and Applications, Vol. 4, No.6, pp. 58-72.

Kimura, F., Kawabe, S. and Sata, T., 1984, "A study on Product Modeling for Integration of CAD/CAM", Integration of CAD/CAD, D. Kochan(ed.) North-Holland, IFIP, pp. 227-244.

King, J.R., 1980, "Machine Component Group Formation Using ROC Algorithm", Int. J. of Production Research, April issue, pp.213-231

Ko, H. and Lee, K, 1987, "Automatic Assembling Procedure Generation from Mating Conditions", Computer-Aided Design, Vol. 19 No. 1, Jan/Feb, pp. 3-10.

Laperriere, L. and ElMaraghy, H.A., 1989, "Automatic Generation of Robotic Assembly Sequence", Proc. of ASME Flexible Assembly Conf., September, Montreal, Canada.

Latombe J.C. and Mazer E., 1981, "LM: a high-level programming language for controlling assembly robots", Proc. 11th ISIR, Tokyo, Japan, pp. 683-690.

Lee, K and Andrews, G., 1985, "Inference of the Positions of Components in an Assembly: Part 2" Computer-Aided Design, Vol. 17, No. 1, Jan/Feb, pp. 20-24.

Lee, K and Gossard, D.C., 1985, "A Hierarchical Data Structure for Representing Assemblies: Part 1", Computer-Aided Design Vol. 17, No.1, Jan/Feb, pp. 15-19.

Lee, Y.C. and Fu, K.S., 1987, "Machine Understanding of CSG: Extraction and Unification of Manufacturing Features", IEEE CG&A, Vol. 7, No. 1, Jan., pp. 20-32.

Li, J., Han, C. and Ham, I., 1987, " CORE-CAPP: Company-Oriented Semi-Generative Computer Automated Process-Planning System, Proc. of 19th CIRP International Seminar on Manufacturing System Computer-Aided Process Planning, Penn State U.S.A. June, pp. 219-225.

Lieberman, L.I. and Wesley, M.A., 1977, "AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly", IBM J. Res. Develop., July, pp. 321-333.

Link, C.H., 1976, "CAPP-CAM Automated Process Planning System", 13th Numerical Control Society Conference Proceedings, Cincinnati, Ohio.

Luby, S.C., Dixon, J.R. and Simmons, M.K., 1986, "Creating and Using a Feature Data Base", Computer in Mechanical Engineering, November, pp. 25-33.

Matsushima, K, Okada, N and Sata, T., 1982, "The Integration of CAD and CAM by Application of Artificial Intelligence Techniques", Annals of the CIRP, Vol. 31/1, pp. 329-332.

Mazer E., 1983, "Geometric programming of assembly robot - LM-GEO", Proc. Advanced Software in Robotics, Liege, Belgium, pp. 99-110.

McAulley, J., 1972, "Machine Grouping for Efficient Production", The Production Engineer, February issue, pp.53-57.

McCormick, W.T., Schweitzer, P.J. and White, T., 1972, "Problem decomposition and Data Reorganization by a Clustering Technique", Operation research, pp. 993-1009.

Newman, W.M. and Sproul, R.F., 1979, "Principles of Interactive Computer Graphics", 2nd ed., McGraw-Hill.

Nilsson, N., 1980, "Principles of Artificial Intelligence", Palo Alto, Calif.: Tioga Press.

Oberg, e., Jones, F.D. and Horton, H.L., 1988, " Machinery's Handbook ", 23rd Edition, Industrial Press INC.

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

Peklenik, J. and Grum, J., 1980, "Investigation of the Computer Aided Classification of Parts", Annals of the CIRP 29/1, pp. 319-323.

Peklenik, J., Grum, J. and Logar, B., 1985, "An Integrated Approach to CAD/CAPP/CAM and Group Technology by Pattern Recognition", Manufacturing System Vol. 14 No.1, pp. 17-37.

Phillips, R.H., Arunthavanathan, V. and Zhou, X. D., 1987, " An Integrated Intelligent Design and Process Planning System", 19th CIRP International Seminar on Manufacturing Systems Computer-Aided Process Planning, Penn State, U.S.A. June, pp. 17-22.

Popplestone, R.J. and Ambler, A.P., 1983, "RAPT: A language for specifying robot manipulations", In:(A. Pugh, ed.) Robotic technology, A. Pugh, ed. Peter Peregrinus, Londcn (Pub.), pp. 125-141.

Pratt, M.J., 1988, "Synthesis of an Optimal Approach to Form Feature

Modeling", Proc. of the 1988 ASME International Computers in Engineering Conference and Exhibition, August, pp. 263-274.

Rangak, P.S. and Fridshal, R., 1988, "Features for Tolerancing a Solid Model", Proc. of the 1988 ASME International Computers in Engineering Conference and Exhibition, August, pp. 275-280.

Rembold, U., Blume, C. and Dillmann, R., 1985, Computer-Integrated Manufacturing Technology And Systems, MARCEL DEKKER, INC. New York.

Requicha, A.A.G., 1980, "Representations for Rigid Solids: Theory, Methods, and Systems", Acm Computing Surveys, Vol. 12, pp. 437-464.

Requicha, A.A.G. 1983, "Toward to a Theory of Geometric Tolerancing", Robotics and Computer-Integrated Manufacturing, Vol. 2, No. 4, Winter, pp 45-60.

Requicha, A.A.G., 1980, "Representation of Tolerances in Solid Modeling: Issues and Alternative Approaches", Solid Modeling by Computers, Edited by Pickett, S. and Boyse, J. W. Plenum Press, N.Y., pp. 3-18.

Requicha, A.A.G. and Voelcker, H.B., 1983, "Solid Modeling: Current Status and Research Directions", IEEE CG&A Vol. 3, No. 7, Oct., pp. 25-37.

Requicha, A.A.G. and Voelcker, H.B., 1982, "Solid Modeling: A Historical Summary and Contemporary Assessment", IEEE CG&A Vol. 2, No. 2, March, pp. 9-24.

Rocheleau, D.N. and Lee, K., 1987, "System for Interactive Assembly Modeling", Computer-Aided Design, Vol. 19 No. 2, March, pp. 65-72.

Ross, G.J.S., 1969, "Algorithms AS13, AS14 and AS15", Applied Statistics, 18, (i), pp. 103-110.

Rondeau, J.M. and ElMaraghy, H.A., 1989, "ROBOPLAN: A Knowledge-based Robot Task Planning System", Proc. of ASME 1st Flexible Assembly Conf.,

September., Montreal, Canada.

Sata, T., Kimura, F., Suzuki, H. and Fujita, T., 1985, "Designing Machine Assembly Structure Using Geometric Constraints in Product Modeling", Annals of the CIRP 34/1, pp. 169-172.

Schaffer, G., 1980, "GT via Automated Process Planning", American Mechanist.

Sedas, S.W. and Talukdar, S.N., 1987, "A Disassembly Planner for Redesign", Intelligent and Integrated Manufacturing Analysis and Synthesis, Edited by C.R. Liu, A. Requicha and S. Chandraseker, pp. 95-107.

Segre, A.M., 1988, "Machine Learning of Robot Assembly Plans", Kluwer Academic Publishers (KAP).

Sekiguchi, H., Kojima, T. Inoue, K and Honda, T., 1983, "Study on Automatic Determination of Assembly Sequence", Annals of the CIRP, Vol. 32/1.

Shah, J.J. and Rogers, M.T., 1988, "Feature Based Modeling Shell: Design and Implementation", Proc. of the 1988 ASME International Computers in Engineering Conference and Exhibition, August, pp. 255-261.

Shah, J.J and Rogers, M.T., 1988, "Functional Requirements and Conceptual Design of the Feature-Based Modeling System", Computer-Aided Engineering Journal, Vol. 5, pp. 9-15.

Siddall, J.M., 1982, "Optimal Engineering Design Principle and Application", New York, M. Dekker (Pub.).

Stefik, M and Bobrow, D.G., 1986, "Object-Oriented Programming: Themes and Variations", AI Magazine, Vol. 6, Nol. 4, pp. 40-62.

Tang, P. and ElMaraghy, H., 1986, "Programming of intelligent robots", In: Proc. of Int. Conf. on Computer-Aided Production Engineering, University

of Edinburgh, pp. 293-299.

Tang, R., Ma, D. and Lu, L., 1987, "Some Strategies Related to the Development of a CAD/CAM System for Mechanical Products Based on Solid Modeling Techniques", Computer in Industry, Vol. 8, pp. 27-34.

Tang, Z., Gu, K. and Sun, J., 1987, "Computer Aided Geometric Modeling System - CAGMS", Computer in Industry, Vol. 8, pp. 19-26.

Tikerpuu, J. and Ullman, D.G., 1988, "General Feature-Based Frame Representation for Describing Mechanical Engineering Design Developed From Empirical Data", Proc. of the 1988 ASME International Computers in Engineering Conference and Exhibition, August, pp. 245-263.

TIPS-1, '83 Version, System Manual, Oct. 1984, Computer Aided Manufacturing-International, INC.

Tou, J.T. and Gonzalez, R.C., 1974. Pattern Recognition Principles, Addison-Wesley Publishing Company, Inc.

Tulkoff, J., 1987, "Process Planning: An Historical Review and Future Prospects", Proc. of 19th CIRP International Seminar on Manufacturing System Computer-Aided Process Planning, Penn State U.S.A. June, pp 207-210.

Tulkoff, J., 1981, "Lockheeds's GENPLAN", 18th Numerical Control Society Conference Proceedings, Dallas, Texas.

Turnar, G.P. and Anderson, D.C. 1988, "An Object-Oriented Approach to Interactive Feature-Based Design for Quick Turnaround Manufacturing", Proc. of the 1988 ASME International Computers in Engineering Conference and Exhibition, August, pp. 551-555.

Unger, M.B. and Ray, S.R., 1988, "Feature-Based Process Planning in the AMRF", Proc. of the 1988 ASME International Computers in Engineering

Conference and Exhibition, August, pp. 563-569.

Van't Erve, A.H. and Kals, H.J.J., 1986, "XPLANE, A Generative Computer-Aided Process Planning System for Part Manufacturing", Annals of the CIRP, Vol. 35/1, pp. 325-330.

Van't Erve, A.H. and Kals, H.J.J., 1986, "XPLANE, a knowledge-based driven process planning expert system", Proc. of International Conference on Computer-Aided Production Engineering, Edinburgh-April, pp. 41-46.

Vere, S., 1983, "Planning in Time: Windows and Durations for Activities and Goals", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 3, May, pp. 264-267.

Vogel, S.A. and Adlard, E.J., 1981, "The AUTOPLAN Process Planning System", 18th Numerical Control Society Technical Conference Proceedings, Dallas, Texas.

Wang, W., 1984, "Solid Modeling for Mold Design and Manufacture", Ph.D Thesis, Cornell University.

Wang, H.P. and Wysk, R.A., 1987, "TURBO-CAPP: A Knowledge-Based Computer Aided Process Planning System", Proc. of 19th CIRP International Seminar on Manufacturing System Computer-Aided Process Planning, Penn State U.S.A. June, pp. 161-169.

Wang, H.P. and Wysk, R.A., 1987, "Intelligent Reasoning for Process Planning", Computers in Industry Vol. 8, pp. 293-309.

Wesley, M.A., Lozano-Perez, T., Lieberman, L.I., Lavin, M.A. and Grossman, D.D., 1980, "A Geometric Modelling System for Automated Mechanical Assembly", IBM J. Res. Develop., Vol. 24, No. 1, Jan., pp. 64-74.

Wright, A.J., Darbyshire, I.L., Park, M.W. and Davies, B.J., 1987, "EXCAP and ICAPP: Integrated Knowledge-Based Systems for Process-Planning

Components", Proc. of 19th CIRP International Seminar on Manufacturing System Computer-Aided Process Planning, Penn State U.S.A. June, pp. 309-313.

Wu, H.L., Venugopal, R. and Barash, M.M., 1986, "Design of a Cellular Manufacturing System: A syntactic Pattern Recognition Approach" Journal of Manufacturing System, Volume 5, Number 2, pp. 81-88.

APPENDIX A LANGUAGES SYNTAX DEFINITIONS

A.1 FDDL Syntax Defintion

The language specification has two levels; the grammar and the lexical level. The symbols used must be clearly specified.

A.1.1 Symbol Definitions

The following will define the symbols used in the grammar and the lexical specifications in the next two sections.

In the grammar level, the following symbols are defined:

Strings in capital letters ----	terminals for grammar level
Anything with angle brackets --	non-terminals
Operator "----->" ----------	signifies a rewrite rule (called a production).
Operator "|" ----------	signifies that what follows is an alternate right side of production.

In the lexical level, the following definitions are given:

Strings in capital letters -----	non-terminals
Anything in quotations ---------	actual text
Operator" ----->" ----------	signifies a rewrite rule (called a production)
Operator "|" ----------	signifies that what follows is an alternate right side of production.

Asterisk "*" ----------	the expression preceding "*" may be repeated zero or more times.
Parentheses, (, and), NOT ----- surrounded by quotations.	the string contained is a regular expression.

A.1.2 Grammar Specification of Language

<start> -------> <parts> LB <features> RB
| <start> <parts> LB <features> RB

<parts> -------> PART LB <partdes> RB

<partdes> -------> <partkey>
| <partdes> <partkey>

<partkey> -------> NAME LB IDENTIFIER RB
| CLASS LB NUMBER RB
| MATERIAL LB MATERIALKEY RB
| HEAT_TREATING LB HEATKEY RB

<features> ------> <fspec>
| <features> <fspec>

<fspec> ------> FEATURE LB <featdes> RB

<featdes> ------> <featureid> COMMA <featuretype> COMMA <location> COMMA <surface>
| <featureid> COMMA <featurekey> COMMA <featuretype> COMMA <location> COMMA <surface>

<featureid> ----> NAME LB IDENTIFIER RB

<featurekey> ----> <tole_relation>
| <featurekey> <tole_relation>

<tole_relation> -> TOLERANCE LB TOLEKEY1 COMMA NUMBER RB
| TOLERANCE LB TOLEKEY2 COMMA NUMBER
COMMA NUMBER RB
| TOLERANCE LB TOLEKEY3 COMMA NUMBER
COMMA NUMBER COMMA IDENTIFIER RB
| TOLERANCE LB TOLEKEY4 COMMA NUMBER
COMMA <datums> RB
| TOLERANCE LB TOLEKEY5 COMMA NUMBER
<datuml> RB
| RELATION LB RELATIONKEY COMMA
IDENTIFIER RB

<surface> ------> SURFACE_FINISH LB NUMBER RB

<featuretype> --> TYPE LB FEAKEY1 COMMA NUMBER RB
| TYPE LB FEAKEY2 COMMA NUMBER COMMA
NUMBER RB
| TYPE LB FEAKEY3 COMMA NUMBER COMMA
NUMBER COMMA NUMBER RB
| TYPE LB FEAKEY4 COMMA <list_number> RB

<list_number> ---> NUMBER COMMA NUMBER COMMA NUMBER
| <list number> COMMA NUMBER

<location> ------> LCCATION LB NUMBER COMMA NUMBER COMMA
NUMBER COMMA NUMBER COMMA NUMBER
COMMA NUMBER RB

<datums> ------> IDENTIFIER
| IDENTIFIER COMMA IDENTIFIER
| IDENTIFIER COMMA IDENTIFIER COMMA

IDENTIFIER

<datuml> ------> NULL | IDENTIFIER

| IDENTIFIER COMMA IDENTIFIER

| IDENTIFIER COMMA IDENTIFIER COMMA

IDENTIFIER

A.1.3 Lexical Definitions

CLASS -------> "class"

HEAT-TREATING ---> "heat_treating"

MATERIAL-------> "material"

NAME -------> "name"

LOCATION-------> "location"

TYPE -------> "type"

RELATION-------> "relation"

SURFACE-FINISH-> "surface_finish"

TOLERANCE----> "tolerance"

NUMBER -------> DIGIT DIGIT*

| "-" DIGIT DIGIT*

| DIGIT DIGIT* "." DIGIT*

| DIGIT* "." DIGIT DIGIT*

| "-" DIGIT DIGIT* "." DIGIT*

| "-" DIGIT* "." DIGIT DIGIT*

DIGIT -------> "0" | "1" | "2" | "3" | "4" | "5"

| "6" | "7" | "8" | "9"

PART -------> "part"

FEATURE -------> "feature"

IDENTIFIER --> LETTER (LETTER | DIGIT | "_")*

```
LETTER   ------->   "a" | "b" | "c" | "d" | "e"
                    | "f" | "g" | "h" | "i"
                    | "j" | "k" | "l" | "m"
                    | "n" | "o" | "p" | "q"
                    | "r" | "s" | "t" | "u"
                    | "v" | "w" | "x" | "y" | "z"
LB       ------->   "("
RB       ------->   ")"
COMMA    ------->   ","
FEAKEY1  ------->   "cylinder_front" | "t_cone_front"
                    | "bore_front"    | "h_t_cone_front"
FEAKEY2  ------->   "a_slot" | "a_slot_a" | "a_slot_b"
                    | "box_face" | "slot_side"
                    | "h_box_face"
                    | "cylinder_side" | "bore"
                    | "h_slot_side"
                    | "p_keyseat_corner" | "p_keyseat_side"
                    | "rectangle" | "slot_face"
                    | "t_f_slot_side" | "thread_nc_front"
                    | "thread_nf_front" | "w_keyseat_front"
                    | "h_hexagon_face" | "h_slot_face"
                    | "r_cylinder
FEAKEY3  ------->   "chamfer" | "fillet" | "gear_front"
                    | "h_chamfer" | "h_fillet"
                    | "h_spline_side" | "h_t_cone_side"
                    |  "h_thread_nc_side"  |  "h_thread_nf_side"
```

```
                    | "t_cone_side" | "t_cylinder_side"
                    | "thread_nc_side" | "thread_nf_side"
                    | "w_keyseat_side" | "spline_front"
FEAKEY4 -------->   "gear_side" | "spline_side"
FEAKEY5 -------->   "hexagon" | "polygon"
HEATKEY -------->   "aging" | "annealing" | "austempering"
                    | "bluing" | "brunorizing" | "burnt"
                    | "carburizing" | "cementation"
                    | "hardening"
RELATIONKEY -->     "pfit" | "pufit" | "mfit" | "lfit"
                    | "tfit" | "sfit"  | "rfit" | "kfit"
                    | "contact_fix"  | "contact_free"
                    | "contact_taper" | "screw_fix"
                    | "screw_tra"    | "spring_contact"
                    | "spring_fix"   | "flexible_connect"
                    | "weld" | "hfit" | "cure"
TOLEKEY1-------->   "roundness" | "flatness"
                    | "cylindricity" | "straightness"
TOLEKEY2-------->   "diameter" | "radius"
TOLEKEY3-------->   "distance"
TOLEKEY4 ------->   "angularity" | "coaxiality"
                    | "concentricity" | "perpendicularity"
                    | "parallelism" | "position"
                    | "runout_c" | "runout_t"
                    | "symmetricity"
TOLEKEY5-------->   "profile_l" | "profile_s"
```

MATERIALKEY --> "aisi_1009" | "aisi_1010" | "aisi_1012"
| "aisi_1015" | "aisi_1016" | "aisi_1017"
| "aisi_1018" | "aisi_1019" | "aisi_1020"
| "aisi_1021" | "aisi_1022" | "aisi_1023"
| "aisi_1024" | "aisi_1025" | "aisi_1026"
| "aisi_1027" | "aisi_1030" | "aisi_1033"
| "aisi_1035" | "aisi_1036" | "aisi_1037"
| "aisi_1038" | "aisi_1039" | "aisi_1040"
| "aisi_1041" | "aisi_1042" | "aisi_1043"
| "aisi_1045" | "aisi_1046" | "aisi_1048"
| "aisi_1049" | "aisi_1050" | "aisi_1052"

.

.

.

A.2 Syntax of Target Language for Parts Assignment

Target language syntax for the feature-based part assignment system is described as follows:

<description> --> <keylist><stop>
| <keylist><description>

<keylist> --> <featurekey> <parameters>
<tolerance list><surface-finish list>
| <featurekey><parameters>
<surface-finish list>

<tolerance list> --> <tolerancekey><tolerance values>
| <tolerancekey><tolerance values>
<tolerance list>

<parameters> --> <positive number><positive number>*

<tolerance values> --> <number><number>*

<number> --> <positive number>

| <negative number>

<positive number> --> <digit><digit>*

| <digit><digit>*<dot><digit>*

| <digit>*<dot><digit><digit>*

<negative number> --> <minus><digit><digit>*

| <minus><digit>*<dot><digit><digit>*

| <minus><digit><digit>*<dot><digit>*

<surface-finish list> -> <surface-finish><positive number>

<surface-finish> --> "surface_finish"

<stop> --> "stop"

<dot> --> "."

<minus> --> "-"

<digit> --> "0" | "1" | "2" | "3" | "4" | "5"

| "6" | "7" | "8" | "9"

<featurekey> --> "cylinder_front" | "t_cone_front"

| "bore_front" | "h_t_cone_front"

.

.

.

<tolerancekey> --> "roundness" | "flatness"

| "cylindricity" | "straightness"

| "diameter" | "radius"

.

All nonterminals are defined by angle brackets and all terminals are quoted. The sign * means repeating zero or more times.

A.3 Syntax of Target Language for Inspection Planning

<description> --> <keylist><stop input> |
<keylist><description>

<keylist> --> <id><featurekey> <main dimensions>
<tolerance list><location>

<tolerance list> --> <tolerancekey>
| <tolerancekey><datums>
| <tolerance list>

<id> --> <positive itegre>

<main dimensions> --> <real><real>

<location> --> "["<real>","<real>","<real>","
"<real>"," <real>","<real>"]"

<datums> --> <integer>
| <integer><integer>
| <integer><integer><integer>

<positive integer> --> <digit><digit>*

<real> --> <digit><digit>*<dot><digit>
<digit>*<e><digit><digit>*
| <digit><digit>*<dot><digit>
<digit>*<e><minus><digit><digit>*
|<minus><digit><digit>*<dot>
<digit><digit>*<e><digit><digit>*

```
                    |<minus><digit><digit>*<dot><digit>
                    <digit>*<e><minus><digit><digit>*
<dot>           -->  "."
<e>             -->  "e"
<minus>         -->  "-"
<digit>         -->  "0" | "1" | "2" | "3" | "4" | "5"
                     | "6" | "7" | "8" | "9"
<featurekey>    -->  "cylinder_front" | "t_cone_front"
                     | "bore_front" | "h_t_cone_front"
                     | "a_slot" | "a_slot_a" | "a_slot_b"
                     | "anulus" | "box_face" | "slot_side"
                     | "h_thread_nc_front" | "h_box_face"
                     | "cylinder_side" | "bore"
                     | "h_slot_side" | "h_spline_front"
                     | "p_keyseat_corner" | "p_keyseat_side"
                          .
                          .
                          .
<tolerancekey>  -->  "roundness" | "flatness"
                     | "cylindricity" | "straightness"
                     | "diameter" | "radius" | "distance"
                     | "angularity" | "coaxiality"
                     | "concentricity" | "perpendicularity"
                     | "parallelism" | "position"
                     | "runout_c" | "runout_t"
                     | "symmetricity"
```

| "profile_l" | "profile_s"

All nonterminals are defined by angle brackets and all terminals are quoted.

A.4 Syntax of Target Language for Assembly Planning

The syntax given below is used to create input for assembly planning system by code generator of assembly. The cited strings are terminals for syntax and all string with angle brackets are nonterminals.

<part> --> <partdes><featuredes><end>
<stopsign><end>
|<partdes><featuredes><end>
<part>

<partdes> --> <identifier><name>

<featuredes> --> <ls><relalist><rs><end>

<relalist> --> <feature><comma><dimension>
<comma><relation><comma>
<identifier>
|<feature><comma><dimension>
<comma><relation><comma>
<identifier><relalist>

<dimension> --> <digit><digit>*<dot><digit>
<digit>*<e><digit><digit>*
|<digit><digit>*<dot><digit>
<digit>*<e><minus><digit><digit>*

<identifier> --> <letter>(<letter> | <digit> | "_")*

<name> --> <letter><letter>*
| <letter><letter>*"-"<letter>*

<end> --> "eee."

<stopsign> --> "stop input"

<comma> --> ","

<ls> --> "["

<rs> --> "]"

<dot> --> "."

<e> --> "e"

<minus> --> "-"

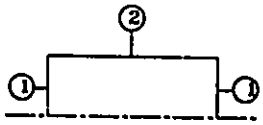
<digit> --> "0" | "1" | "2" | "3" | "4" | "5"
| "6" | "7" | "8" | "9"

<letter> --> "a" | "b" | "c" | "d" | "e" | "f"
| "g" | "h" | "i" | "j" | "l" | "m"
| "n" | "o" | "p" | "q" | "r" | "s"
| "t" | "u" | "v" | "w" | "x" | "y"
| "z"

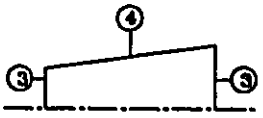
<feature> --> "cylinder_front" | "cylinder_side" |
"box_face" | "t_cone_front" |
"bore" | "bore_front" |
.
.
.

<relation> --> "pfit" | "pufit" | "mfit" | "lfit"
| "sfit" | "contact_free"
.
.
.

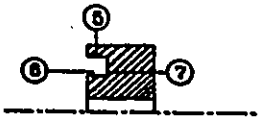
APPENDIX B FEATURE DEFINITIONS



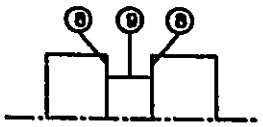
1. cylinder_front, diameter, location and orientation
2. cylinder_side, diameter, length, location and orientation



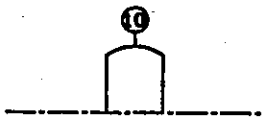
3. t_cone_front, diameter, location and orientation
4. t_cone_side, diameter1, diameter2, length, location and orientation



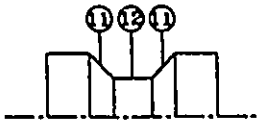
5. a_slot_a, diameter, length, location and orientation
6. a_slot_b, diameter, length, location and orientation
7. a_slot, diameter1, diameter2, location and orientation



8. slot_face, diameter1, diameter2, location and orientation
9. slot_side, diameter, length, location and orientation

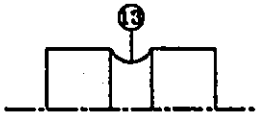


10. r_cylinder, radius1, radius2, angle, location and orientation

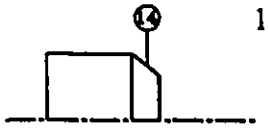


11. t_f_slot_a, diameter1, diameter2, length(or angle), location and orientation

12. t_f_slot, diameter, length, location and orientation



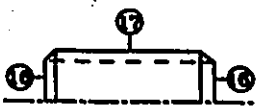
13. s_c_slot, radius1, radius2, angle, location and orientation



14. chamfer, diameter1, diameter2, length, location and orientation

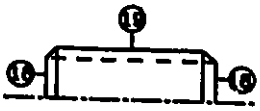
15. fillet, radius1, radius2, angle, location and orientation

16. thread_nf_front, diameter, location and orientation



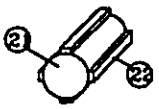
17. thread_nf_side, diameter, length, location and orientation

18. thread_nc_front, diameter, location and orientation



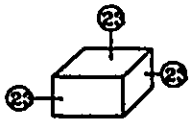
19. thread_nc_side, diameter, length, pitch, location and orientation

20. t_cylinder_top, length, width, location and orientation



21. spline_front, diameter, number of keys, location and orientation

22. spline_side, diameter, length, number of keys, location and orientation

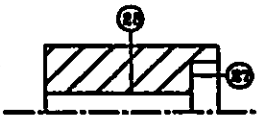


23. box_face, length, width, location and orientation

24. polygon_face, coordinates of all points, location and orientation

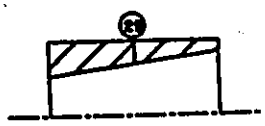
25. gear_front, diameter, number of teeth, modular, location and orientation

26. gear_side, diameter, length, number of teeth, modular location and orientation

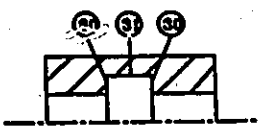


27. bore_front, diameter, location and orientation

28. bore, diameter, length, location and orientation

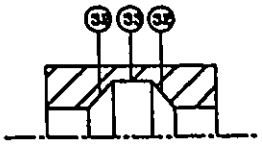


29. h_t_cone_side, diameter1, diameter2, location and orientation



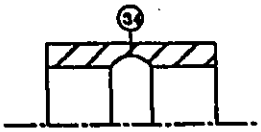
30. h_slot_face, diameter1, diameter2, location and orientation

31. h_slot_side, diameter, length, location and orientation

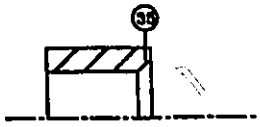


32. h_t_f_slot_a, diameter1, diameter2, length(or angle), location and orientation

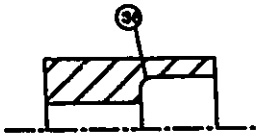
33. h_t_f_slot, diameter, length, location and orientation



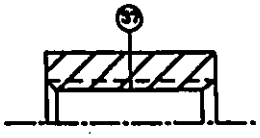
34. h_s_c_slot, radius1, radius2, angle, location and orientation



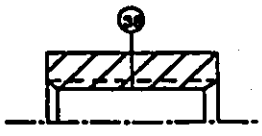
35. h_chamfer, diameter1, diameter2, length, location and orientation



36. h_fillet, radius1, radius2, angle, location and orientation

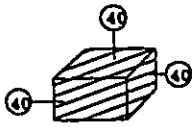


37. h_thread_nf_side, diameter, length, location and orientation



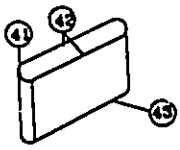
38. h_thread_nc_side, diameter, length, pitch, location and orientation

39. h_spline_side, diameter, length, number of keys, location and orientation



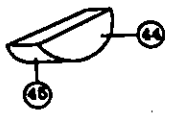
40. h_box_face, length, width, location and orientation

41. p_keyseat_corner, radius, depth, location and orientation



42. p_keyseat_side, length, depth, location and orientation

43. p_keyseat_bottom, length, width, location and orientation



44. w_keyseat_front, radius, angle, location and orientation

45. w_keyseat_side, diameter, width, length, location and orientation

APPENDIX C HEAT-TREATING

aging

annealing

austempering

bluing

brunorizing

burnt

carburizing

cementation

hardening

APPENDIX D CONNECTING RELATIONS

Fitting Relations

pfit	------	pressure fit
pufit	------	push fit
mfit	------	movable fit
lfit	------	loose fit with big clearance
tfit	------	taper fit
sfit	------	spline fit
rfit	------	ring fit
kfit	------	key fit

Contacting Relations

contact_fix	------	contact with fix
contact_free	------	contact without fix
contact_taper	------	contact by taper

Screw Relations

screw_fix	------	fixing and positioning screw
screw_tra	------	transmission screw

Spring Relations

spring_contact	------	contact without hooking
spring_fix	------	spring with hooking

Other Relations

flexible_connect	-	flexible connecting
weld	------	welding together
hfit	------	red assembly
cure	------	cure assembly

APPENDIX E FEATURES DEFINED IN INSPECTION PLANNER

```
feature(bore, cylinder, empty, machine).
feature(bore, circle, empty, machine).
feature(bore_front, plane, solid, machine).
feature(bore_front, point, solid, machine).
feature(cylinder_side, cylinder, solid, machine).
feature(cylinder_side, circle, solid, machine).
feature(cylinder_front, plane, solid, machine).
feature(cylinder_front, point, solid, machine).
feature(a_slot_a, cylinder, empty, machine).
feature(a_slot_b, cylinder, solid, machine).
feature(a_slot, plane, solid, machine).
feature(a_slot_a, circle, empty, machine).
feature(a_slot_b, circle, solid, machine).
feature(a_slot, point, solid, machine).
feature(t_f_slot_a, cone, solid, machine).
feature(t_f_slot_b, cylinder, machine).
feature(t_f_slot_b, circle, machine).
feature(spline_side, cylinder, solid, machine).
feature(spline_front, plane, solid, machine).
feature(spline_side, circle, solid, machine).
feature(spline_front, point, solid, machine).
feature(t_cylinder_top, plane, solid, machine).
```

feature(t_cylinder_side, cylinder, solid, machine).

feature(t_cylinder_front, plane, solid, machine).

feature(t_cylinder_side, circle, solid, machine).

feature(t_cylinder_front, point, solid, machine).

feature(t_cylinder_top, point, solid, machine).

feature(h_t_f_slot_a, cone, empty, manual).

feature(h_t_f_slot, cylinder, empty, manual).

feature(h_slot_face, plane, empty, manual).

feature(h_slot_side, cylinder, empty, manual).

feature(p_keyseat_side, plane, solid, machine).

feature(p_keyseat_corner, cylinder, empty, machine).

feature(p_keyseat_bottom, plane, solid, machine).

feature(p_keyseat_side, point, solid, machine).

feature(p_keyseat_corner, circle, empty, machine).

feature(p_keyseat_bottom, point, solid, machine).

feature(box_face, plane, solid, machine).

feature(box_face, point, solid, machine).

feature(h_box_face, plane, empty, machine).

feature(h_box_face, point, empty, machine).

feature(polygon, plane, solid, machine).

feature(polygon, point, solid, machine).

feature(t_cone_face, plane, solid, machine).

feature(t_cone_side, cone, solid, machine).

feature(t_cone_face, point, solid, machine).

feature(h_t_cone_front, plane, solid, machine).

feature(h_t_cone_front, point, solid, machine).

feature(h_t_cone_side, cone, empty, machine).

feature(chamfer_bore, cone, empty, machine).

feature(chamfer, cone, solid, machine).

feature(thread_nc_front, plane, solid, machine).

feature(thread_nc_side, cylinder, solid, manual).

feature(thread_nc_front, point, solid, machine).

feature(thread_nf_front, plane, solid, machine).

feature(thread_nf_side, cylinder, solid, manual).

feature(thread_nf_front, point, solid, machine).

feature(h_thread_nc_front, plane, solid, machine).

feature(h_thread_nc_side, cylinder, empty, manual).

feature(h_thread_nc_front, point, solid, machine).

feature(h_thread_nf_front, plane, solid, machine).

feature(h_thread_nf_side, cylinder, empty, manual).

feature(h_thread_nf_front, point, solid, machine).

feature(h_t_cone_front, plane, solid, machine).

feature(h_t_cone_side, cone, empty, machine).

feature(h_t_cone, point, solid, machine).

feature(w_keyseat_front, point, solid, machine).

feature(w_keyseat_side, circle, empty, machine).

feature(geartooth, profile, solid, manual).

feature(h_geartooth, profile, empty, manual).

feature(sphere, sphere, solid, machine).

APPENDIX F SOURCE LANGUAGE OF FLASH LIGHT

```
part (name (part01)
        , class (10)
        , material (aisi_1330)
        , heat_treating (aging)
        )
    (feature (name (face2)
            , type (cylinder_front, 12.0)
            , location (0.0, 0.0, 0.0, 180, 90, -90)
            , surface_finish (3.0)
            )
        feature (name (cylinder1)
            , type (cylinder_side, 12.0, 3.0)
            , location (0.0, 0.0, 0.0, 180, 90, -90)
            , relation (pufit, part02)
            , tolerance (diameter, 1.8, 0.0)
            , surface_finish (4.0)
            )
        feature (name (cone3)
            , type (t_cone_side, 12.0, 9.0, 10, 0)
            , location (3.0, 0.0, 0.0, 180.0, 90.0, -90.0)
            , surface_finish (5.0)
            )
        feature (name (face4)
            , type (t_cone_front, 10.0)
            , location (12.0, 0.0, 0.0, 0.0, -90.0, 90.0)
            , surface_finish (7.0)
            , relation (contact_free, part05)
            )
        )
    part (name (part02)
        , class (3)
        , material (aisi_1038)
        , heat_treating (annealing)
        )
     (feature (name (face1)
            , type (cylinder_front, 43.0)
            , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
            , surface_finish (7.0)
            )
        feature (name (cylinder2)
            , type (cylinder_side, 43.0, 20.0)
            , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
            , surface_finish (4.0)
            )
```

```
        feature (name (face3)
                , type (cylinder_front, 43.0)
                , location (20.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (7.0)
                )
        feature (name (hole_thread4)
                , type (h_thread_nf_side, 32.0, 15.0, 1.5)
                , location (5.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (2.5)
                , relation (screw_fix, part03)
                )
        feature (name (face5)
                , type (bore_front, 32.0)
                , location (5.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (4.0)
                )
        feature (name (bore6)
                , type (bore, 12.0, 3.0)
                , location (2.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , tolerance (diameter, -0.02, 0.01)
                , surface_finish (1.5)
                , relation (pfit, part01)
                )
        )
part (name (part03)
        , class (3)
        , material (aisi_1038)
        , heat_treating (annealing)
        )
 (feature (name (face1)
                , type (thread_nf_front, 32.0)
                , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (2.5)
                )
        feature (name (screw2)
                , type (thread_nf_side, 32.0, 12.0, 1.5)
                , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (2.5)
                , relation (screw_fix, part02)
                )
        feature (name (cylinder3)
                , type (cylinder_side, 32.0, 65.0)
                , location (12.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (3.0)
                )
        feature (name (screw4)
                , type (thread_nf_side, 32.0, 12.0, 1.5)
                , location (77.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (2.5)
```

```
                , relation (screw_fix, part06)
                )


        feature (name (face5)
                , type (thread_nf_front, 32.0)
                , location (0.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (2.5)
                )
        feature (name (bore6)
                , type (bore, 24.0, 89.0)
                , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (3.5)
                )
        )
part (name (part04)
        , class (3)
        , material (aisi_1038)
        , heat_treating (annealing)
        )
 (feature (name (face1)
                , type (cylinder_front, 20.0)
                , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (4.0)
                , relation (contact_free, part05)
                )
        feature (name (cylinder2)
                , type (cylinder_side, 20.0, 34.0)
                , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (4.0)
                )
        feature (name (face3)
                , type (cylinder_front, 20.0)
                , location (34.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (4.0)
                )
        feature (name (cylinder4)
                , type (cylinder_side, 5.0, 2.0)
                , location (34.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (4.0)
                )
        feature (name (face5)
                , type (cylinder_front, 5.0)
                , location (34.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (4.0)
                )
        )
part (name (part05)
        , class (3)
        , material (aisi_1038)
        , heat_treating (annealing)
```

```
      )
(feature (name (face1)
            , type (cylinder_front, 20.0)
            , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
            , surface_finish (4.0)
            , relation (contact_free, part01)
            )
      feature (name (cylinder2)
            , type (cylinder_side, 20.0, 34.0)
            , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
            , surface_finish (4.0)
            )
      feature (name (face3)
            , type (cylinder_front, 20.0)
            , location (34.0, 0.0, 0.0, 0.0, -90.0, 90.0)
            , surface_finish (4.0)
            )
      feature (name (cylinder4)
            , type (cylinder_side, 5.0, 2.0)
            , location (34.0, 0.0, 0.0, 0.0, -90.0, 90.0)
            , surface_finish (4.0)
            )
      feature (name (face5)
            , type (cylinder_front, 5.0)
            , location (34.0, 0.0, 0.0, 0.0, -90.0, 90.0)
            , surface_finish (4.0)
            , relation (contact_free, part04)
            )
      )
part (name (part06)
      , class (3)
      , material (aisi_1038)
      , heat_treating (annealing)
      )
 (feature (name (face1)
            , type (cylinder_front, 43.0)
            , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
            , surface_finish (5.0)
            )
      feature (name (cylinder2)
            , type (cylinder_side, 42.0, 33.0)
            , location (33.0, 0.0, 0.0, 180.0, 90.0, -90.0)
            , surface_finish (5.0)
            )
      feature (name (face3)
            , type (cylinder_front, 43.0)
            , location (33.0, 0.0, 0.0, 0.0, -90.0, 90.0)
            , surface_finish (5.0)
            )
      feature (name (hole_thread4)
            , type (h_thread_nf_side, 32.0, 8.0, 1.5)
            , location (5.0, 0.0, 0.0, 0.0, -90.0, 90.0)
```

```
                , surface_finish (2.5)
                , relation (screw_fix, part03)
                )
        feature (name (bore6)
                , type (bore, 32.0, 3.0)
                , location (8.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , tolerance (diameter, -0.02, 0.01)
                , surface_finish (1.5)
                , relation (pfit, part09)
                )
        )
part (name (part07)
        , class (3)
        , material (aisi_1038)
        , heat_treating (annealing)
        )
 (feature (name (face1)
                , type (cylinder_front, 30.0)
                , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (5.0)
                , relation (contact_free, part09)
                )
        feature (name (cylinder2)
                , type (cylinder_side, 30.0, 2.0)
                , location (2.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (5.0)
                )
        feature (name (face3)
                , type (cylinder_front, 30.0)
                , location (2.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (5.0)
                )
        )
part (name (part08)
        , class (3)
        , material (aisi_1038)
        , heat_treating (annealing)
        )
 (feature (name (face1)
                , type (thread_nf_front, 6.0)
                , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (4.5)
                )
        feature (name (screw2)
                , type (thread_nf_side, 6.0, 3.0, 1.0)
                , location (3.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (2.5)
                , relation (screw_fix, part09)
                )
        feature (name (cylinder3)
                , type (cylinder_side, 6.0, 2.0)
                , location (3.0, 0.0, 0.0, 180.0, 90.0, -90.0)
```

```
                , surface_finish (3.0)
                )
        feature (name (face4)
                , type (cylinder_front, 8.0)
                , location (5.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (4.5)
                )
        feature (name (cylinder5)
                , type (cylinder_side, 8.0, 2.0)
                , location (5.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (3.0)
                )
        feature (name (face6)
                , type (cylinder_front, 8.0)
                , location (7.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (4.5)
                )
        )
part (name (part09)
        , class (3)
        , material (aisi_1038)
        , heat_treating (annealing)
        )
 (feature (name (face1)
                , type (cylinder_front, 13.0)
                , location (0.0 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (4.0)
                )
        feature (name (cone2)
                , type (t_cone_side, 40.0, 13.0, 12.0)
                , location (12.0, 0.0, 0.0, 180.0, 90.0, -90.0)
                , surface_finish (4.5)
                )
        feature (name (cylinder3)
                , type (cylinder_side, 33.0, 5.0)
                , location (12.0, 0.0, 0.0, 180, 90, -90)
                , relation (pfit, part03)
                , tolerance (diameter, 1.8, 0.0)
                , surface_finish (4.0)
                )
        feature (name (face4)
                , type (cylinder_front, 33.0)
                , location (17.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (4.0)
                )
        feature (name (h_cone5)
                , type (h_t_cone_side, 27.0, 8.0, 12.0)
                , location (17.0, 0.0, 0.0, 0.0, -90.0, 90.0)
                , surface_finish (4.5)
                )
        feature (name (face6)
                , type (bore_front, 8.0)
```

```
        , location (5.0, 0.0, 0.0, 0.0, -90.0, 90.0)
        , surface_finish (4.0)
        )
feature (name (hole_thread7)
        , type (h_thread_nf_side, 6.0, 3.0, 1.0)
        , location (0.0, 0.0, 0.0, 180.0, 90.0, -90.0)
        , surface_finish (4.5)
        , relation (screw_fix, part08)
        )
)
```