TIPPS - A Totally Integrated Process Planning System

by

Tien-Chien Chang

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
in
Industrial Engineering & Operations Research

APPROVED:

R. A. Wysk, Chairman

M. H. Agee

R. P. Davis

C. E. Nunnally

J. M. A. Tanchoco

November, 1982
Blacksburg, Virginia

AN INTRODUCTION TO AUTOMATED PROCESS PLANNING SYSTEMS AND TIPPS

by

Tien Chien Chang

(ABSTRACT)

Computer-aided process planning is an essential interface between computer-aided design and computer-aided manufacturing. In this thesis, a computer-aided process planning system - TIPPS is developed. A CAD design model is used for direct input in the TIPPS system. A boundary internal model of the engineering part drawing is displayed on a CRT screen, and using an interactive procedure, surface which require machining can be marked. A backward planning scheme searches a process knowledge data base to find a sequence of manufacturing processes which can achieve the design specification. The process knowledge data base consists of process description statements. A process description statement (mathematical terms are used) represents the capabilities of a single process. TIPPS also selects process parameters - feed and cutting speed automatically. TIPPS provides: 1) direct CAD interface, 2) external process capability description language, 3) modular structure, and 4) interactive surface identification.

Various approaches used in designing and process planning are discussed. A review of CAD, process engineering, planning decision methods and current process planning systems is included.

ACKNOWLEDGEMENTS

I would like to take this opportunity to acknowledge the help and support I received in my study.

First my sincere gratitude and special thanks are due my major professor, Dr. Richard A. Wysk. His guidance, constant encouragement and continued interest throughout my graduate study is deeply appreciated. He has been patient especially in reading and correcting my manuscript. I will always feel grateful for my association with Dr. Robert P. Davis. He always provided encouragement and helpful suggestions. I have received so much from him, not only knowledge but also an attitude for being a teacher and researcher.

I would also like to offer my thanks to Drs. Jose M. A. Tanchoco, Charles E. Nunnally, Marvin H. Agee and Timothy Greene for their interest in this research. They have provided their time and suggestions. Thanks are due the faculty and graduate students at the Department of Industrial Engineering & Operations Research. They have provided me such a wonderful environment to learn and for fellowship.

Special thanks go to Ms. Donna Lovern who typed the original manuscript.

Last but not least, I thank my wife Li-Fen, without her help, understanding and love this task would have been infinitely more difficult.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter I

INTRODUCTION TO MANUFACTURING

1.1 MANUFACTURING INDUSTRIES

A production facility may be the most complex system which confronts today's modern engineer. In spite of the fact that the production function is the primary purpose of a manufacturing industry, the production process often appears completely forgotten in the midst of marketing, accounting, or other organizational planning activities. Perhaps it is the lack of recognition, and attention, given to manufacturing systems that has led to our decline in industrial productivity.

The impact of our declining productivity is so severe that productivity improvement has probably become the most challanging problem facing the United States today. Two of the effects of our declining productivity include:

1) A reduced position in the international marketplace (the direct result of this would be a reduction in the U.S. "standard of living").

2) Manufacturing superiority usually indicates a nation's ability to produce military hardware; there-

by, providing tactical superiority as well as a deterrent to aggressive countries.

It is not an appealing legacy to leave the next generation of Americans with a poorer standard of living than has been experienced in the recent past. Never in the United States has this occurred. Similarly, many historians cite both tactical as well as industrial deficiencies as a major cause of aggression (again, not a very appealing legacy to leave to future generations). If our ability to produce quality products provides us with the basis for our standard of living and for our defense, then "Why is it that we spend the majority of our efforts (money and time) with ancillary issues such as marketing (packaging, advertising, etc.), accounting and other administrative issues?" A product's (and company's) success must begin with a well conceived product and production process; other activities serve only to slightly embellish a product.

In spite of the general impression given, manufacturing problems are not easily separated issues. Management organization does impact manufacturing; however, other issues such as material management, shop floor control, and the like, more directly affect a manufacturing system. The major purpose of this thesis will be to identify a structure by which different manufacturing issues might be addressed.

The majority of its content will focus on process planning systems, since process planning provides one of the major integration issues in a factory environment.

However, before embarking on a more technical discussion, it is worthwhile to trace some of the history of manufacturing. By viewing this history, one might glean a perspective of where present industry stands in the evolution of manufacturing.

The history of manufacturing can be said to have begun when man first tried to alter the geometry of a workpiece by rotating either a tool or a workpiece on a spindle to remove pieces of the material from the workpiece. This was important to create weapons to protect man against natural enemies as well as unfriendly tribes. The oldest evidence discovered so far is a fragment of an Etruscan wooden bowl found in the 'Tomb of the Warrior' at Corneto in 700 B.C. [Rolt 1965]. However, it was not until the fifteenth century that man began machining metal. Eighteenth century industrialization ushered in the demand for production type machine tools. Machine shops were established to create more machines. Production systems became more organized, and metal processing became a trade carried out by experienced craftsman whose trade was passed from generation to generation. Small production volume with no standards made every product a unique one. A typical example of this era is that it took James Watt twenty-five years (after completing the design) to produce the first steam engine. The tolerance between the piston and cylinder was "no greater than a worn shilling" -- a true feat for the time.

The advent of interchangeable parts can be attributed to the realization that we could not easily repair damaged firearms. This concept in manufacturing is credited to Eli Whitney (1765-1825). As the pioneer of this new form of manufacturing, he signed (in 1798) a contract with the U.S. government for 12,000 muskets -- 4,000 to be delivered in the first year. Although only 500 muskets were delivered by 1801, and it was 1806 when he finally finished the contract, a new era in manufacturing history began.

Interchangeability in manufactured parts not only meant a new technique in machining and inspection, but also created a better, more uniform planning method. A conventional machine shop had previously used the skill of a machinist to determine the entire operation of a product. Until this new era began, as long as the product functioned as the original design specified, the product was acceptable. (No guarantee on the sameness of the parts constrained production.) However, the new manufacturing concept (interchangeability) demanded that every part be made the same with only a small variance. A more thorough plan for production was obviously necessary. Over the years, as the demand for a product grew (whether it be weaponary or consumer products) mass production became the standard for economically satisfying the demand.

Scientific study in metal cutting and mass production are products of the twentieth century, when market demand contributed to the development of mass production techniques. Increased demand also promoted the scientific study of manufacturing. Scientific research and development focused on production processes, planning and technology. Perhaps the greatest impact came from scientific management, pioneered by Frederick W. Taylor (1856-1915). "Methods" for production by machines as well as by people were studied. Standards for human operators and metal machining were established.

Taylor is best known for two major contributions: one is scientific management, and the other is his investigations of metal cutting. He conducted metal cutting experiments, at Midvale Steel Company, which lasted for 26 years. His studies not only resulted in the invention of high speed steel, which revolutionized metal cutting efficiency, but also established machinability tables. In his 1905 paper, <u>On the Art of Cutting Metals</u>, which was based on the results of some 50,000 experiments and 800,000 pounds of metal chips, he proposed a relationship of tool life to feed and speed. A better understanding of how metal is cut and how to regulate these variables (feed and speed) in cutting was provided. The Taylor tool life equation and his metal cutting experiments still play an important role in process planning.

Although manufacturing industry continued to evolve, it was not until the 1950's that the next major development occurred. For some time, strides to reduce man's involvement in manufacturing were being taken. Speciality machines using cams and other "hardwired" logic controllers had been developed. The U.S. Air Force recognized the development required to produce this special equipment and that the time required to make only small sequence changes was excessive. As a result, the Air Force commissioned the Massachusetts Institute of Technology to demonstrate programmable, or Numerically Controlled (NC), machines (softwired machines). With this first demonstration in 1952 came the beginning of a new era in manufacturing. Since then, digital computers have been used to produce input in either a directed manner to many NC machines, Direct Numerical Control (DNC), or in a more dedicated control sense, Computer Numerical Control (CNC). Today, machine control languages such as APT (Automatic Programming Tool) have become the standard for creating tool control for NC machines.

It is interesting to note that much of the evolution in manufacturing came as a response to particular changes during different periods of time. For instance, the technology which evolved in the 19th century brought with it the need for higher precision machining (This resulted in the creation of many new machine tools, a more refined machine design, and new production processes). The early 1900's brought an era of prosperity and industrialization that created the demand necessary for mass production techniques. In the 1950's, it was estimated that, as the speed of an aircraft increased, the cost of manufacturing the aircraft (because of geometric complexity) increased proportionately with the speed. The result of this stimulus was the development of NC technology.

A few tangential notes on this history include the following. As the volume of parts manufactured increases, the production cost for the parts decreases (This is generally known as "Economy of Scale"). Some of the change in production cost is due to fixed versus variable costs. For instance, if only a single part is to be produced (like a space vehicle), all of the fixed costs for planning and design (both product and process) must be absorbed by the single item. If, however, several parts are produced, then the fixed charges can be distributed over several parts. Chang-

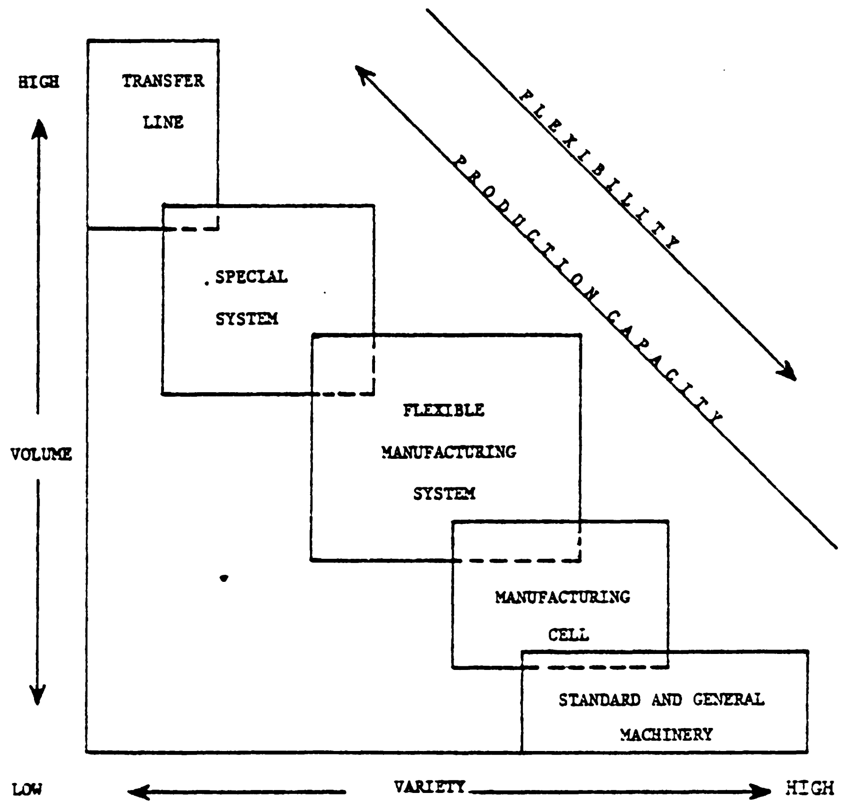


Figure 1.1 System Selection Versus Volume and Variety of Parts

es in production cost, not reflected in this simple fixed versus variable cost relationship, are usually the result of different manufacturing procedures -- transfer line techniques for high volume items versus job shop procedures for low volume items. Figure 1.1 illustrates the fundamental relationship of volume versus production system, and Figure 1.2 depicts cost versus volume for different types of systems.

Also of interest to note is that as the tolerance specification is tightened, the cost of manufacturing the product increases. This is either the result of requiring "finished machining operations", or because a reduced speed and feed rate is necessary to obtain the required geometric specificiation. This relationship is illustrated in Figure 1.3.

Finally with these cost curves and figures, one can look ahead to the 1980's and 1990's. The U.S. Department of Commerce has pointed out that, in the United States, 95% of all products are produced in lots of size 50 or fewer. This indicates that although high volume techniques are desireable from a consumer standpoint (lower cost), these techniques are not appropriate from a manufacturing standpoint because the volume will not offset the setup expenses. The manufacturing alternative to produce those parts is

▨ - Conventional Machines

○ - Transfer Lines

▨ - Numerical Control

◌ - Special Automatics

⊖ - Flexible Manufacturing Systems

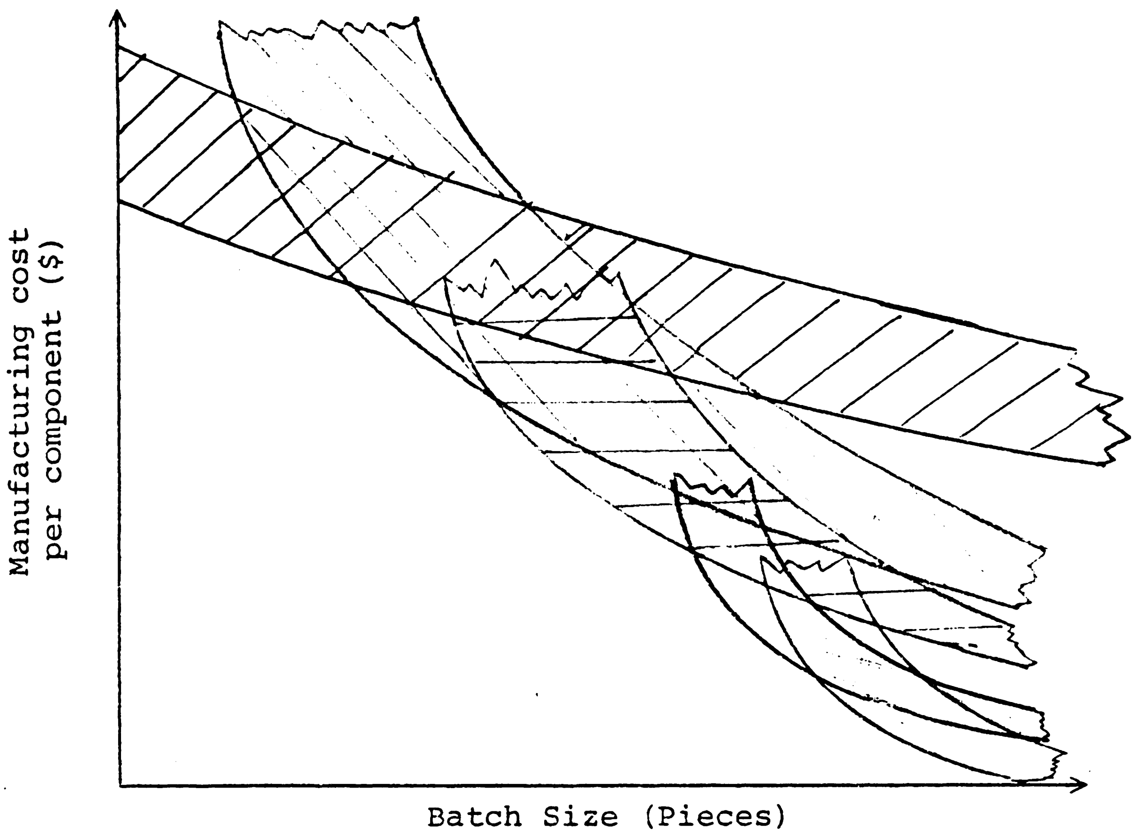


Figure 1.2 Comparison of average cost of various machining systems as a function of number of pieces per production run.

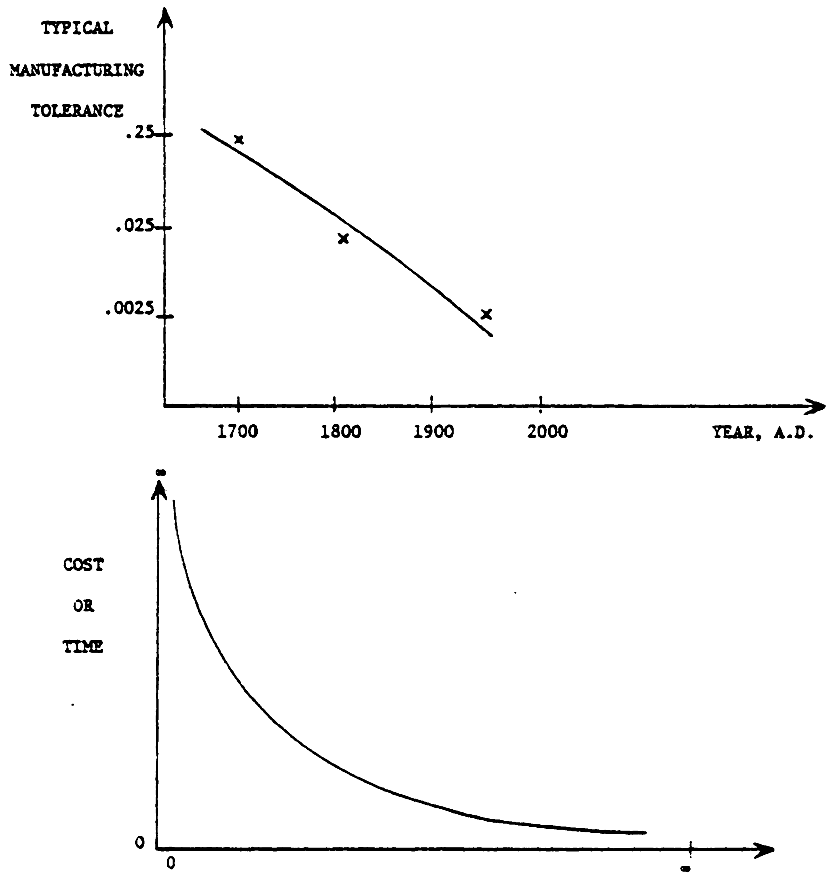


Figure 1.3 History of Tolerance Specification and Cost Versus Tolerance

through the use of Flexible Manufacturing Systems (FMS). These systems are nothing more than programmable job shops. However, a major economic expense still stands before one can begin employing such systems more fully. This obsticle is that a considerable set-up (planning) expense is still required. The alternative to eliminate this expensive set-up is through integration of Computer-Aided Design and Computer-Aided Manufacturing (CAD/CAM). In an integrated CAD/CAM system, parts will be detailed using a computer graphics system. This system will store the geometric information necessary to create process plans and generate the machine instructions necessary to control the machine tools. Some estimates suggest this approach will reduce planning time for FMS parts by more than 95%.

1.2 BASIC TAXANOMY OF MANUFACTURING

1.2.1 Discrete versus Continuous Manufacturing

Manufacturing systems can be classified into two major categories: discrete part manufacturing and continuous process manufacturing. Continuous process manufacturing refers to the production of a continuous product, in which the entire process is controlled through valves, pumps, heaters, etc. For instance, a chemical reaction transforms the raw material into final product. On the other hand, in discrete part manufacturing the product undergoes a finite number of production or assembly operations. This thesis will focus on discrete part manufacturing and more specifically, the manufacturing of machined parts.

1.2.2 Design for Production

Manufacturing is a means of physically transforming a design into reality. Two sets of input are required for manufacturing: design and resources. The resources include raw material, labor, and other utilities. The design includes symbols bearing the specifications and shape of the product. Since manufacturing is the only place where value is added to raw material, the reduction of manufacturing cost implies a lower product cost. Unfortunately, designers often consider their job as one of designing the product for performance, appearance, and possibly reliability, and that it is the manufacturing engineer's job to produce whatever has been designed. While the manufacturing engineer tries his best to find an applicable process to satisfy the design, a minor change in the design may reduce the manufacturing requirements without affecting the product's functionality.

For example, a non-functional surface with a surface tolerance specification simply adds an additional finishing, machining process. A flat bottom hole is more difficult to machine than a hole with a conic shaped bottom (Figure 1.4). A flat-bottomed hole that is threaded its entire length is virtually impossible to produce. Little difference results in using these design alternatives; yet, entirely different problems result for the manufacturing engineer that must execute the process design.

A designer should always keep production efficiency in mind during his design. General rules are [Boothroyd 1975]:

1. Whenever possible, use standard components.
2. Whenever possible, take advantage of the work material shape to design the components.
3. Whenever possible, use the previous design of similar jobs.
4. Whenever possible, minimize the required machining.
5. In designing the shape of the component, consider the ease of material handling, fixturing, machining and assembly.
6. Avoid over-specified tolerances and surface finish specifications.
7. Consider the kinematic principles during the initial steps of a design.

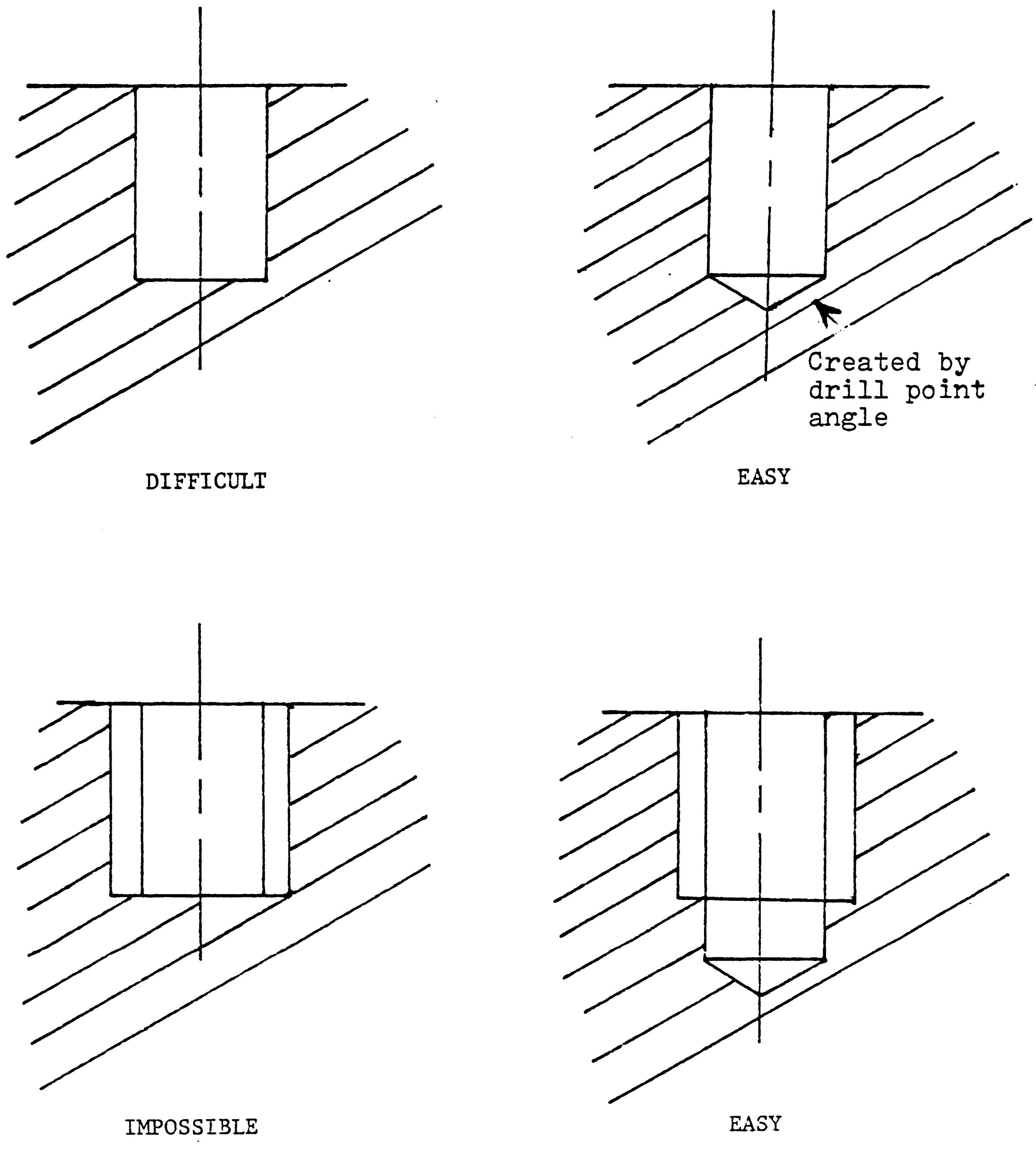


Figure 1.4 DESIGN FOR MACHINING

1.2.3 Material Processing

"Material processing" used in this thesis is a synonym for "machining". It is the basic procedure for transforming the workpiece into final component geometry by removing pieces of the workpiece material. Machine tools are employeed for this purpose. Basic machining processes include:

1. turning
2. drilling
3. reaming
4. boring
5. tapping
6. milling
7. grinding
8. broaching
9. sawing
10. EDM/ECM (Electro Discharge Machining/Electroc Chemical Machining)
11. Laser

For the past two hundred years, basic machining processes have changed little with the exception of EDM, ECM and Lasers. Only the material, structure of machine tools, power sources, and control methods have evolved from a very primitive form to today's standards. Some of the most common machining processes are illustrated in Figures 1.5. Turning and drilling are two of the most ancient machining processes. In turning, the workpiece is rotated by a stationary tool to peel off a thin layer of material. The components produced are symmetric in shape. In drilling, the tool is rotated at high rpm and uses its sharp cutting edges (normally two) to remove material in chip form. The chip is removed via the drill flutes.

Reaming and boring are similar to drilling except that they do not produce a hole, but instead enlarge holes and improve the quality of an existing hole. Reaming improves the hole diameter tolerance and surface finish. Boring provides positioning accuracy and also produces a better surface finish than does drilling.

Milling is probably the most versatile of all processes. It is used widely for external shape forming from the most simple flat surface to delicate surfaces such as a curved propeller surface.

Grinding is a finishing process which creates a very fine surface. When a fine surface finish is specified, grinding is normally used. Small amounts of metal are usually removed in grinding.

Broaching is a lesser used process that is normally efficient when machining a large volume of metal from a "flat or slotted" workpiece. Because of high tool cost, broaching is typically a high volume production process.

ECM (Electro Chemical Machining) and EDM (Electrical Discharge Machining) are non-conventional processes. ECM is similar to electroplating (the difference being that the workpiece material is removed and washed away by an electrolyte). EDM on the other hand uses small sparks created by an electrode (the tool). The sparks vaporize a small spot on the workpiece material which is then washed away by an electrolyte. Both ECM and EDM are very effective in machining very hard materials such as casting dies. Very delicate parts can also be machined with high surface finish and a small or no burr. However, special tooling is required for different jobs.

Laser (Light Amplification by the Stimulated Emission of Radiation) is a process using a light beam of high intensity and single frequency to burn and vaporize workpiece material. The laser was developed in 1958, and has been used in industry only in recent years. As mentioned earlier, laser machining is non-conventional. It can be used in place of drilling, cutting and welding. Because of the efficiency of laser beam generation and the surface reflection properities

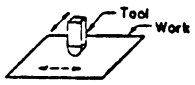
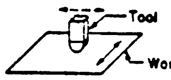
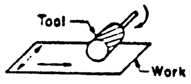
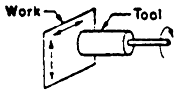
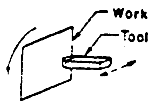
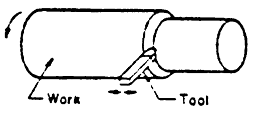
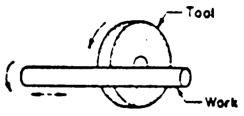
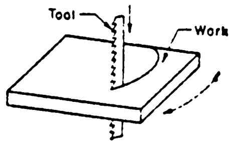
Operation	Block Diagram	Most Commonly Used Machines	Machines Less Frequently Used	Machines Seldom Used
Shaping		Horizontal shaper	Vertical shaper	
Planing		Planer		
Milling	slab milling  face milling 	Milling machine		Lathe (with special attachment)
Facing		Lathe	Boring mill	
Turning		Lathe	Boring mill	Vertical shaper Milling machine
Grinding		Cylindrical grinder		Lathe (with special attachment)
Sawing		Contour saw		

Figure 1.5: Operations and Machines for the Machining of Surfaces

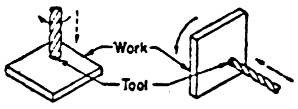
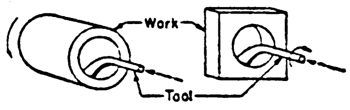
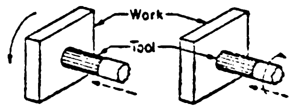
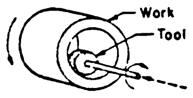
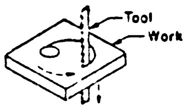
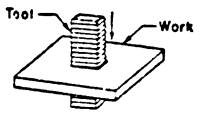
Operation	Block Diagram	Most Commonly Used Machines	Machines Less Frequently Used	Machines Seldom Used
Drilling		Drill press	Lathe	Milling machine Boring mill Horizontal boring machine
Boring		Lathe Boring mill Horizontal boring machine		Milling machine Drill press
Reaming		Lathe Drill press Boring mill Horizontal boring machine	Milling machine	
Grinding		Cylindrical grinder		Lathe (with special attachment)
Sawing		Contour saw		
Broaching		Broaching machine		

Figure 1.5: Continued

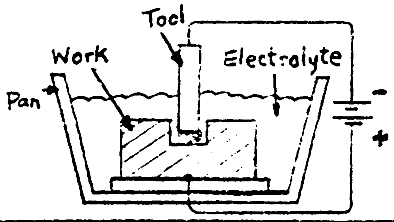
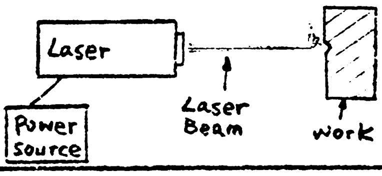
ECM				
Laser				

Figure 1.5 <u>(Continue)</u>

of the light beam, laser machining is not very (energy) efficient. It is typically used for special purpose applications.

1.2.4 Material Handling

The major objective of material handling is to deliver the right material (raw material, workpiece or finished part) at the right time to the proper location (with the right orientation). Economic considerations are extremely important since material handling is a zero value added operation. Reduction in material handling time results in an improvement in the total production efficiency. This saving however must offset additional handling costs.

1.2.4.1 Material Handling Function

In general, the material handling function in a machine shop includes:

1. Receiving
2. In-process handling
3. In-process storge
4. Workplace handling
5. Finished part storage

Receiving is the function which accepts raw material from venders/suppliers. Transportation is necessary to move raw material from receiving storage to the machine tool. In-process handling is required to move the workpiece inside the shop among machines and in-process storages. While in-process storage is used to locate parts in temporary storage areas (eg. a bin, a box, or a pallet), workplace handling requires much more care and accuracy. Loading/unloading the workpiece directly effects the machining accuracy and thus requires high positioning accuracy. After the part has been completed, it is transported to the finished part storage area (warehouse). This stage is the final material handling stage in the machine shop.

1.2.4.2 Material Handling Methods

Several properties of the material affect the material handling method. Such as,

1. Material properties
2. Shape
3. Size
4. Weight

Small workpieces or parts are normally carried in boxes, bins or trays. Large heavy workpieces or parts are carried on pallets. The transportation of boxes, bins, trays, and pallets within the shop, such as the receiving and in-process handling is done by either manual or mechanical equipment (Table 1.1).

1.2.5 Material Planning

Material planning is the process of choosing the appropriate material form for successive machining operations. Aside from technological requirements (such as material properties), other factors involved in material planning include:

1. Component shape and size,
2. Quantity,
3. Convenience of machining, and
4. Cost and availability

The component shape and size is a dominant factor in the final cost of a product. However, both economic as well as physical constraints must be considered. For instance, one can use a 4" diameter bar stock to make a 1/4" diameter pin. The choice of a 4" diameter bar to make a 1/4" diameter pin is a poor economic choice since the majority of material will be machined to chips and scraped. Therefore, the processing time will be exceedingly long and costly. For a V-8

Table 1.1 Material Handling Equipment

TRANSPORTATION	HANDLING
DOLLY	PUSH ROD
PUSH CART	ROBOT
FORKLIFT TRUCK	
GRAVITY CHUTE	
CONVEYOR	
ROLLER	
BELT	
POWERED ROLLER	
CRANE	
TOW CART	
MONORAIL	
AUTOMATED GUIDED VEHICLE (AGV)	
TRANSFER TABLE	

engine block, one can either machine directly from a "slab" of metal, or first cast the material into a rough form and then machine it to the final shape and finish. Since the first alternative requires excess machining of the material, it is obviously very inefficient. However, casting requires a large initial investment. Therefore the quantity also plays an important role. The choice between standard stock and casting can be expressed by a break-even equation (also see Figure 1.6).

$$C_1 + \Sigma C_i V_i \leq C_m/N + C_i' V_i' + C_1$$

C_1 : cost for preparing one workpiece from stock

C_i : cost of machining a unit volume by process i.

V_i : volume being machined by process i from stock.

C_m : cost of mold.

N : break-even quantity.

C_i' : incremental cost of making one casting.

V_i' : volume being machined by process i from the casting.

N is the break-even quantity. When the batch size is larger than N, then it is desireable to cast and then machine. If the batch size is smaller than N, then it is more economical to fabricate the part from stock. If the batch size equals N, then the two methods are indifferent.

1.2.6 Process Planning

Manufacturing planning, process planning, material processing, process engineering and machine routing are only a few of the titles given to the topic referred to here as process planning. Similarly, material processors or process planners are only two of the titles given to those people that actually perform this function. Since these titles and descriptions are so often confused, some time should be given to the definition of process planning and the use of this terminology throughout this thesis. "Process Planning" is that function within a manufacturing facility that establishes which machining processes and parameters are to be used (as well as those machines capable of performing these processes) to convert (machine) a piece-part from its initial form to a final form predetermined (usually by a Design Engineer) from an engineering drawing. Alternatively, process planning could be defined as the act of preparing detailed work instructions to produce a part. The initial material may take a number of forms; the most common of which are bar stock, plate, castings, forgings or maybe just a

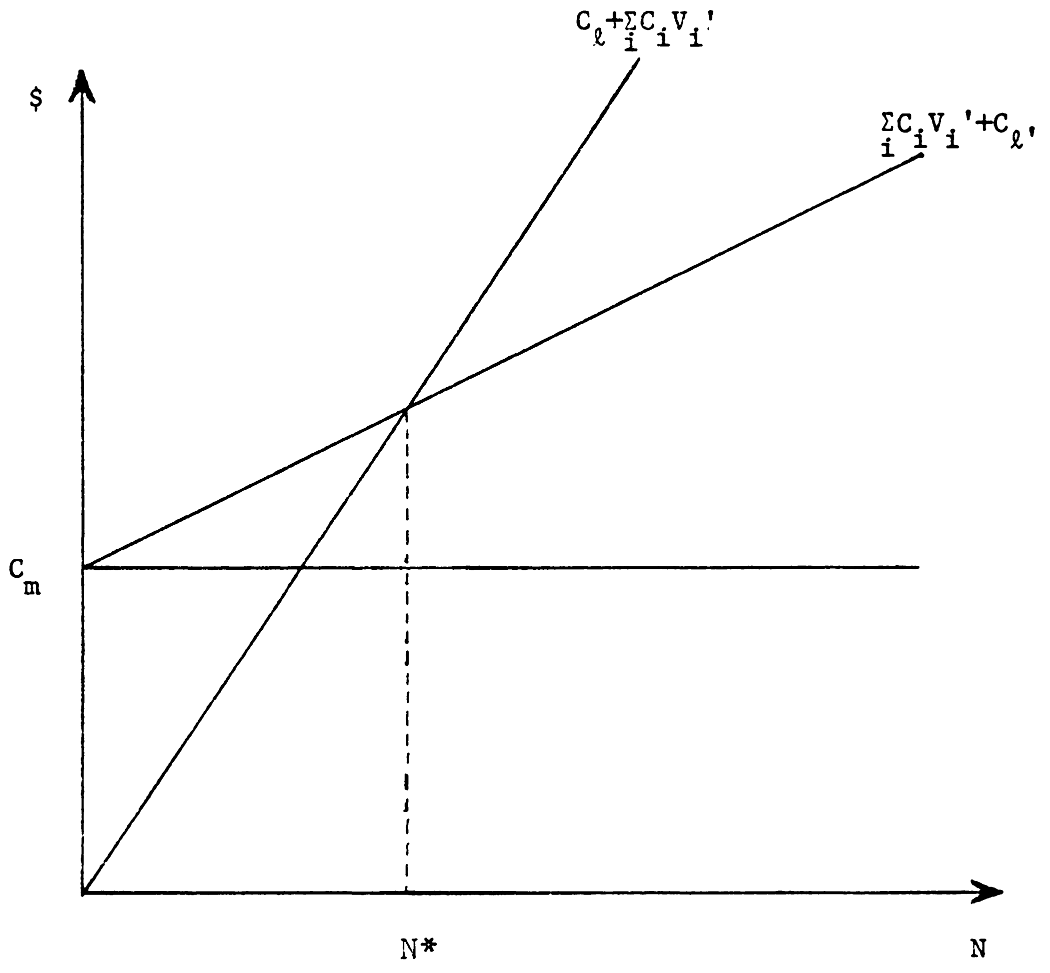


Figure 1.6 BREAK EVEN QUANTITY

slab (of any geometry) of metal (see Figure 1.7). The slab of material is normally a burn-out, cut to some rough dimension. This metal slab can consist of almost any geometry.

With these raw materials as a base, the process planner must prepare a list of processes needed to convert this normally predetermined material into a predetermined final shape. The processes used by a process planner in a discrete part metal manufacturing industry are shown in Figure 1.8. Some of the above operations are often considered subsets of some major category. Facing can be considered a subset of turning. Reaming can be considered a subset of drilling.

The process plan (Figure 1.9) is frequently called an operation sheet, route sheet, operation planning summary, etc. The detailed plan usually contains the route, processes, process parameters, machine and tool selections. In a more general sense, a process is called an operation (including manual operations). (The route is the operation sequence.) The process plan provides the instructions for production of the part. These instructions dictate the cost, quality and rate of production. Therefore process planning is of utmost importance to the production system.

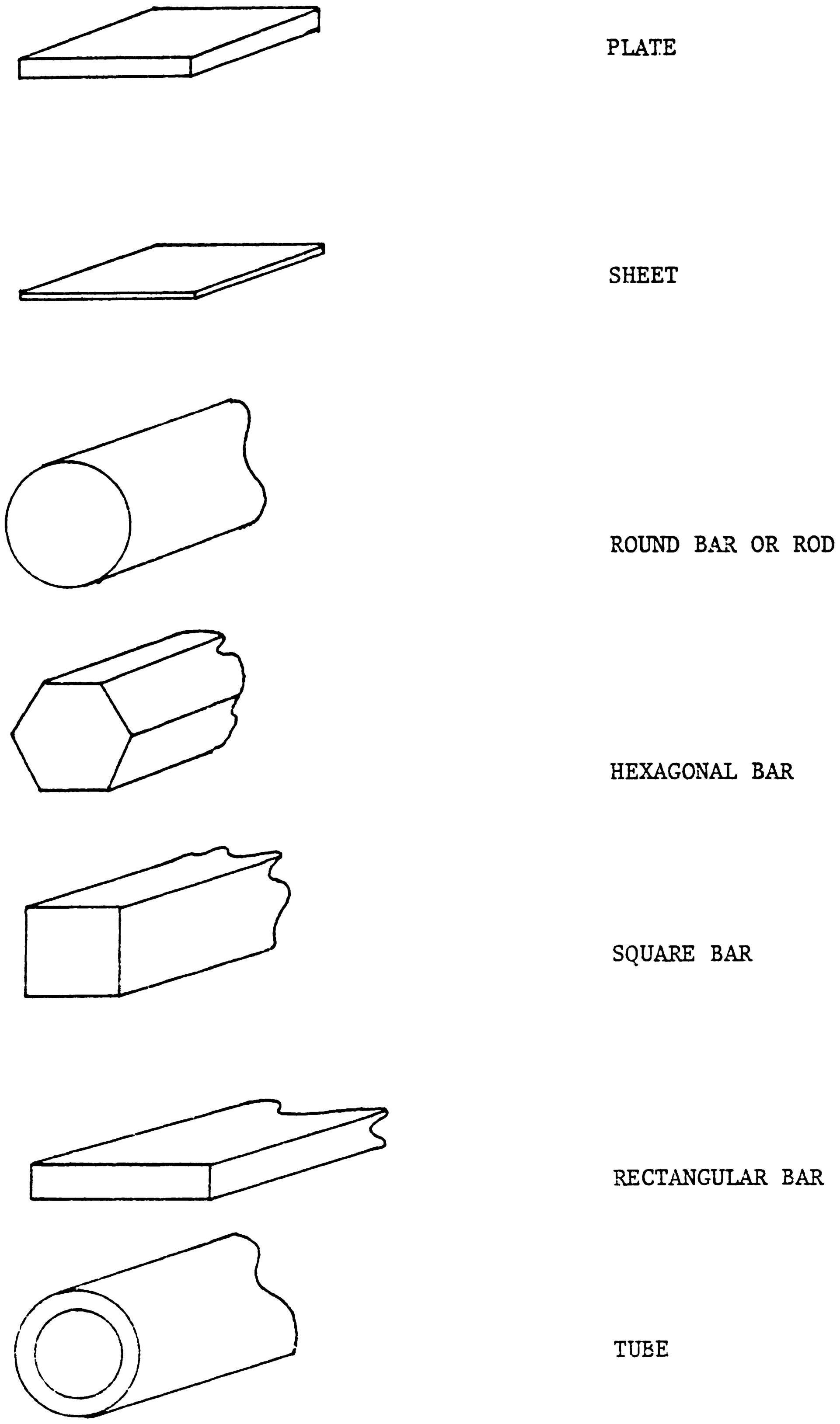


Figure 1.7 Standard Material Shapes

I. Assembly

II. Heat Treating

  A. Annealing
  B. Case hardening
  C. Through hardening
  D. Stress relieving

  .
  .
  .

III. Machining

  A. Turning
  B. Milling
  C. Drilling
  D. Boring
  E. Sawing

  .
  .
  .

IV. Inspection

  A. Measurement
  B. Sampling

  .
  .
  .

Figure 1.8 Processes used in discrete parts metal manufacturing

OPERATION SHEET (Process Plan)

Part No.	S576-67	Material	Cast Iron 9 lbs/pc
Part Name	Eccentric Strap Cap Half		
Orig.		Changes	
Checked		Approved	

No.	Operation Description	Machine	Set-up Description	Operate Hr/Unit
5	Rough and Finish Mill 2 Mating Surfaces	Cinc. Mill. (Kender #136)	Gang 6 castings in fixture.	
10	Spotface and drill two holes 33/84" D Drill 27/84" D pipe hole; tap 1/4" pipe thread.	Multi-Spindle Drill Press	Piece on table Piece in 35° drill jig.	
15	Rough & Finish Bore 6-1/4" D. Bore 6-1/2" D x 3/8" wide groove.	Bullard Vert. Boring Mill (Kender #335)	Clamp to Eccentric Connector Half (S563-5) then mount both parts in 4 jaw chuck.	

Figure 1.9 Process Plan

In a conventional production system, a process plan is created by a process planner who examines a new part (engineering drawing), and then determines the appropriate procedures to produce it. The previous experience of the process planner is critical to the success of the plan. Planning, as practiced today, is as much an art as it is a formal procedure.

As mentioned previously, there are numerous factors which affect process planning. The shape, tolerance, surface finish, size, material type, quantity, the manufacturing system itself, etc., all contribute to the selection of operations and the operation sequence. These factors will be discussed in detail in the next chapter.

In addition to operation sequencing and operation selection, the selection of tooling and jigs/fixtures is also a major part of the process planning function. The tooling portion includes selection of both the tool itself and the machine on which the tool is used. There is a limited set of commercially available tools, with different shapes, diameters, lengths, number of teeth and alternate tool materials. The most commonly used tool materials are high speed steel (HSS) and tungsten carbide (TC). TC can machine metal at higher cutting speed than HSS, but costs more and is difficult or impossible to regrind. TC is also brittle, andis not recommended for interruped cuts. All the above conditions need to be taken into account in order to select an appropriate tool.

Jigs/fixtures are devices to guide a tool or hold a workpiece for better machining. Very few standard devices are available. Even with standard devices, application is normally left to the machine operator. Process planning, however, should consider the effect of tooling, and jigs/fixtures on the quality of the product during the selection of the production operations.

1.2.7 Computer-Aided Process Planning

Process planning is a task which requires a significant amount of both time and experience. According to an Air Force study, a typical process planner is a person over 40 years of age, with long-term experience in a machine shop. Although the U.S. industry requires about 200,000 to 300,000 process planners, only 150,000 to 200,000 are currently available. Automating process planning is an obvious alternative to eliminate this imbalance.

Early attempts to automate process planning primarily consist of building computer-assisted systems for report generation, storage and retrieval of plans. When used effectively, these systems can save up to 40% of a process planner's time. A typical example can be found in Lockheed's CAP system [Tulkoff 1981]. Such a system can by no means perform the process planning tasks; rather, it helps reduce the clerical work required of the process planner.

Perhaps the best known automated process planning system is the CAM-I Automated Processs Planning system -- CAPP (CAM-I stands for Computer-Aided Manufacturing International, a non-profit industrial research organization). In the CAPP system, previously prepared process plans are stored in a data base. When a new component is planned, a process plan for a similar component is retrieved and subsequently modified by a process planner to satisfy special requirements. The technique involved is called group technology (GT), or variant planning. Both GT and variant planning will be discussed in more detail later. The typical organization of a variant process planning is illustrated in Figure 1.10.

Recent developments in computer-aided process planning have focused on eliminating the process planner from the entire planning function. Computer-aided process planning can reduce some of the decision making required during a planning process. It has the following advantages:

1. It can reduce the skill required of a planner.
2. It can reduce the process planning time.

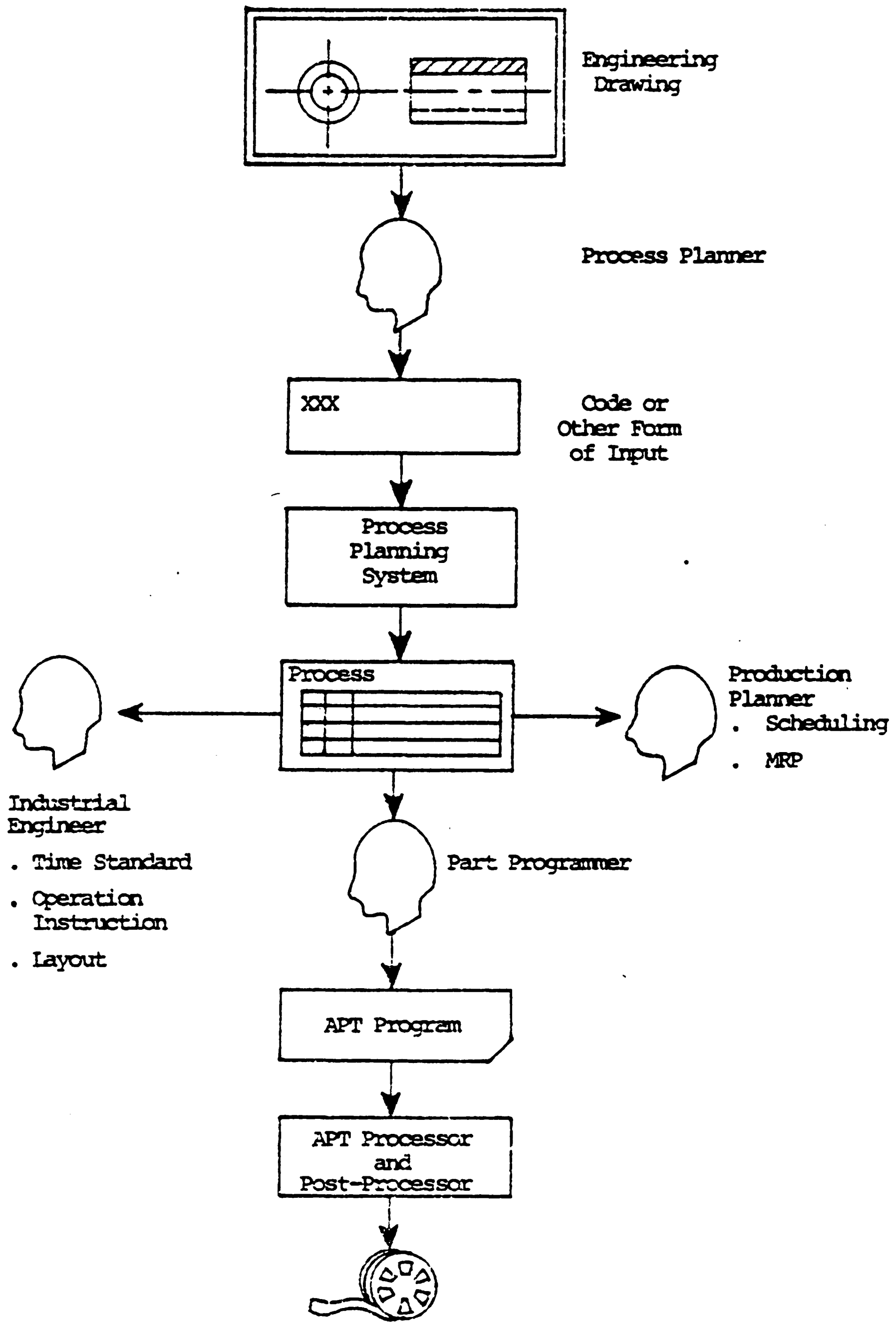


Figure 1.10 A Typical Process Planning System

3. It can reduce both process planning and manufacturing cost.
4. It can create more consistant plans.
5. It can produce more accurate plans. And finally,
6. It can increase productivity.

The benefits of computer-aided process planning systems have been documented by several industries [Vogel 1980, Tulkoff 1981, Dunn, et. al. 1978, Kotler 1980a,b]. Such systems can reduce planning time from hours to minutes.

Two approaches to computer-aided process planning are currently being pursued: variant and generative. The variant approach uses library retrieval procedures to find standard plans for similar components. The standard plans are manually created by process planners. The generative approach is considered the more advanced as well as the more difficult to develop system. In a generative process planning system, process plans are generated automatically for new components without referring to existing plans. Details of these two approaches will be discussed in Chapter 3.

Figure 1.11 represents the structure of a complete computer-aided process planning system. Although no existing turnkey system integrates all of the functions shown in Figure 1.11 (or even a goodly portion of them), the figure illustrates the functional dependencies of a complete process planning system. It also helps to illustrate some of the constraints imposed on a process planning system (eg. available machines, tooling, jigs, etc.).

In Figure 1.11 the modules are not necessarily arranged based on importance or decision sequence. The system monitor controls the execution sequence of the individual modules. Each module may require execution several times in order to obtain an optimum process plan. Iterations are required to reach feasibility as well as a good economic balance.

The input to the system will most probably be a 3-D model from a CAD data base. The model contains not only the shape and dimensioning information, but also the tolerances and special features. The process plan can be routed directly to the production planning system and production control system. Time estimates and resource requirements can be sent to the production planning system for scheduling. The part program, Cutter Location (CL) file and material handling control program can also be sent to the control system.

Process planning is the critical bridge between design and manufacturing (Figure 1.12). Design information can only be translated through process planning into

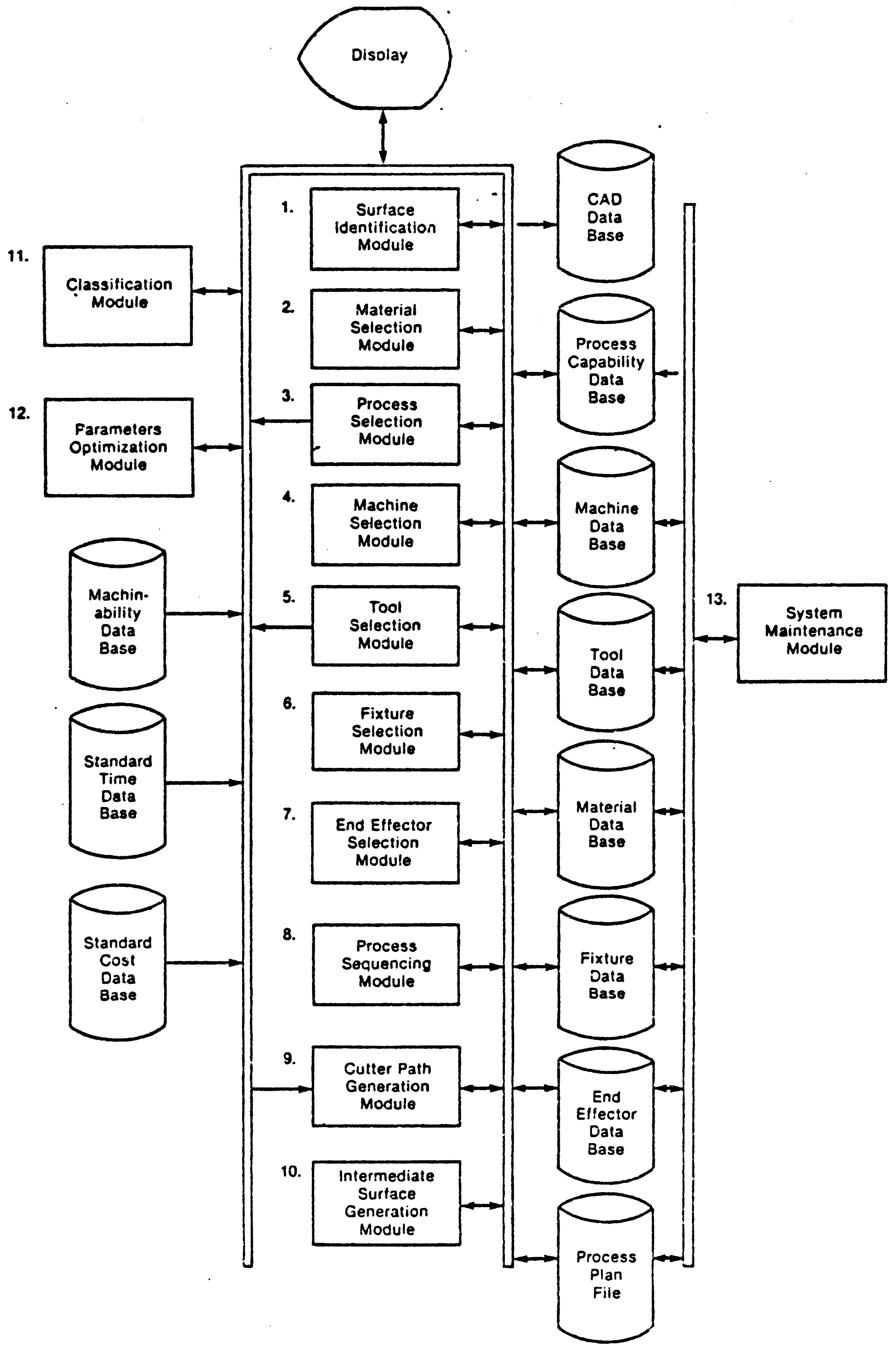


Figure 1.11 Process Planning Modules and Data Bases
(taken from [Chang 1982])

manufacturing language. Today, both automation design (CAD) and manufacturing (CAM) have been implemented. Integration or bridging these functions requires automated process planning.

1.3 GROUP TECHNOLOGY

Since the beginning of human culture, man has tried to apply reason to things. One important way to apply reason is to relate similar things. Biologists classify items into genus and species. We relate to such things as mammals, marsupials, batrachians, amphibians, fish, mollusks, crustacean, birds, reptiles, worms, insects, etc. A chicken is a bird with degenerated wings. A tiger, jaguar, and cat are all members of a single family.

The same concept applied to natural phenomena can also be applied to artificial ones. When a vast amount of information needs to be kept and ordered, a taxonomy is normally employed. Librarians use taxonomies to classify books in libraries. Similarly, in manufacturing, thousands of items are produced yearly. When one looks at the parts which construct the product, the number is exceptionally large. Each part has a different shape, size, and function. However when one look closely, one may again find similarities among

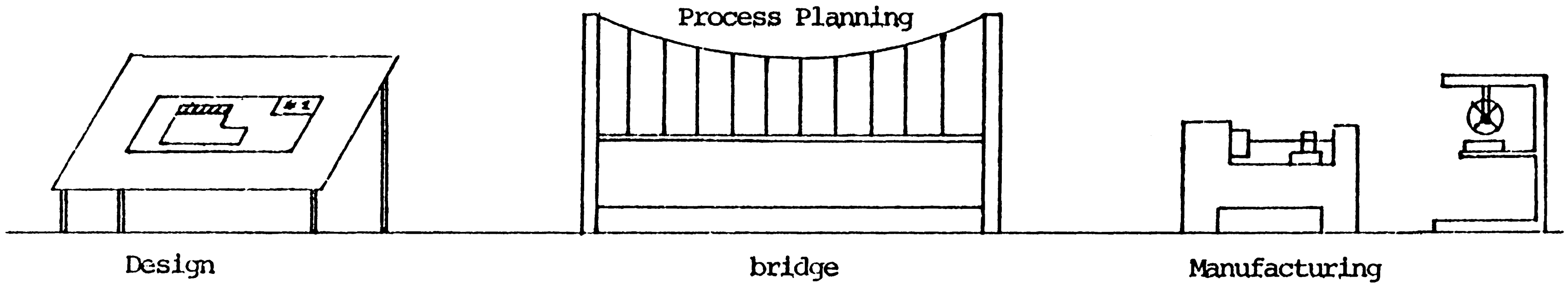


Figure 1.12 <u>Process Planning Bridges Design & Manufacturing</u>

components (Figure 1.13); a dowel and a small shaft may be very similar in appearnace but different in function. Spur gears of different sizes need about the same manufacturing processes. It, therefore, appears that manufactured components can be classified into families similar to biological families or library taxonomies. Parts classified and grouped into families produce a much more tractable data base for management.

Although this simple concept has been in existance for a long time, it was not until 1958 when S. P. Mitrofanov, a Russion engineer, formalized the concept and put together a book entitited, The Scientific Principles of Group Technology. Group technology (GT) has been defined [Solaja 1969] as:

"Group Technology is the realization that many problems are similar, and that by grouping similar problems, a single solution can be found to a set of problems thus saving time and effort".

Although the definition is quite broad, one usually relates grouptechnology to only production applications. In production systems, group technology can be applied in different areas. For component design, it is clear that many components have a similar shape (Figure 1.13). Similar components therefore can be grouped into design families. A new design can be created by modifying an existing component

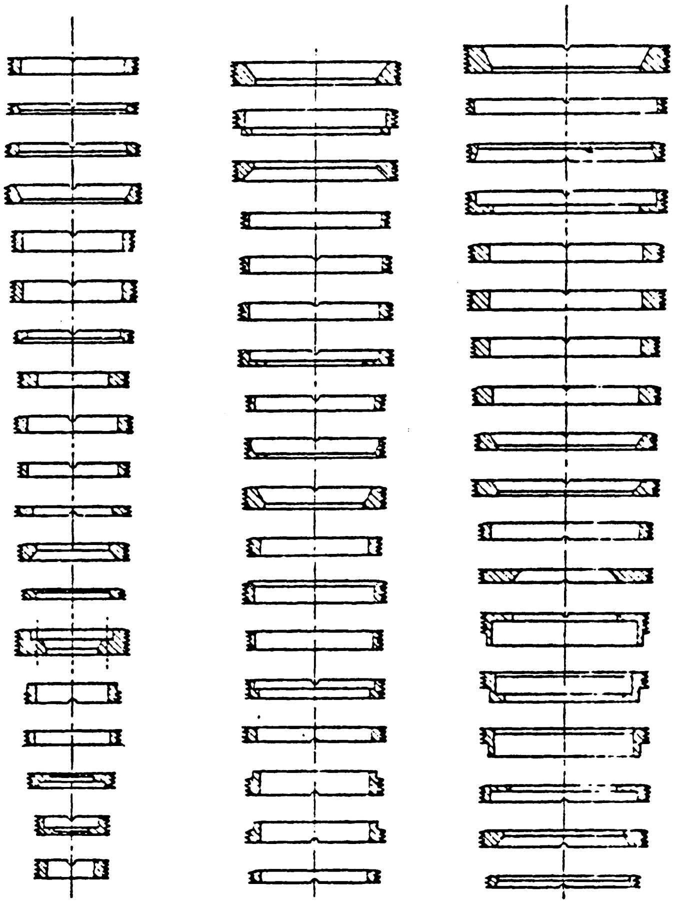


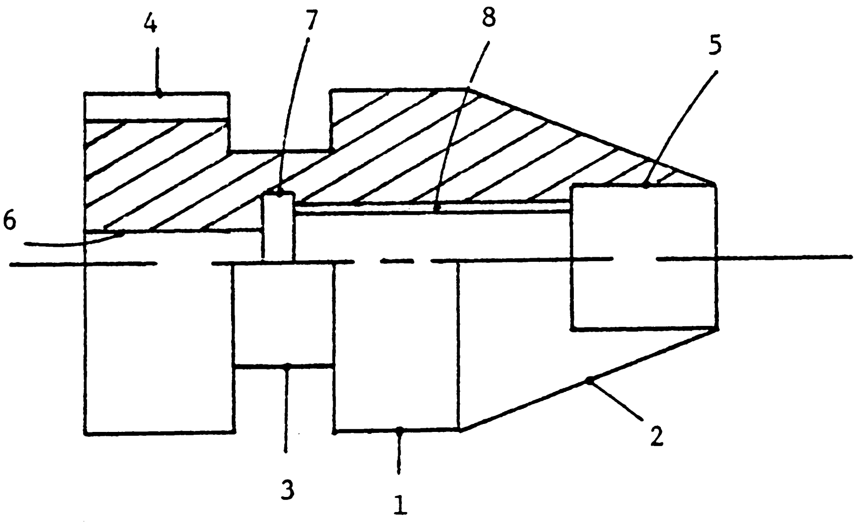
Figure 1.13 A Design Family (parts grouped based on shape)

design from the same family. Using this concept composite components can be identified. Composite components are parts which embody all the design features of a design family or design sub-family. An example is illustrated in Figure 1.14. Components in the family can be identifed from features of the composite components.

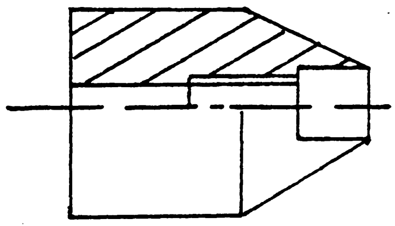
For manufacturing purposes, GT represents a greater importance than simply a design philosgraphy. Components which are not similar in shape may still require similar manufacturing processes. For example, in Figure 1.15 most components have different shapes and functions; but all of them require internal boring, face milling, hole drilling, etc. Therefore, it can be concluded that the components in Figure 1.15 are similar. The set of similar components can be called a production family. From this, process planning work can be facilitated. Since similar processes are required for all family members, a machine cell can be built to manufacture the family. This makes production planning and control much easier, since only similar components are considered for each cell. Such a cell-oriented layout is called a group technology layout.

The following techniques are employed in GT:

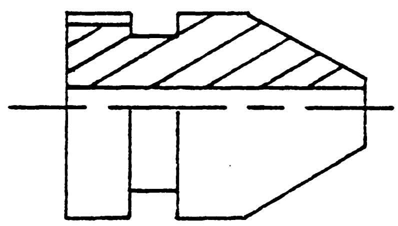
1. Coding and classification
2. Production flow analysis
3. Group layout



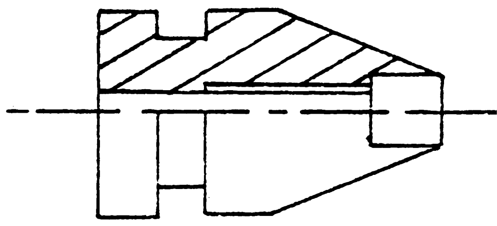
1 TO 8: FEATURE NUMBERS



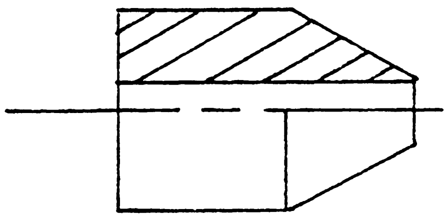
FEATURES:
(1,2,5,6,8)



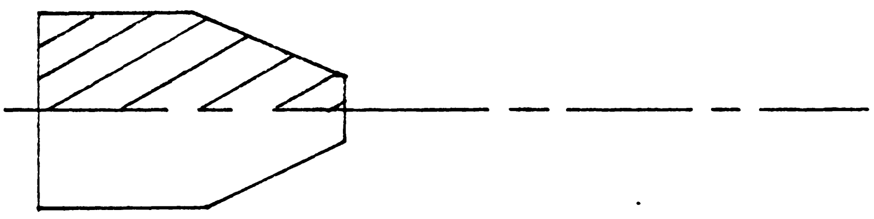
(1,2,3,4,5)



(1,2,3,5,6,8)



(1,2,5)



(1,2)

(....)

Figure 1.14 A COMPOSITE COPONENT

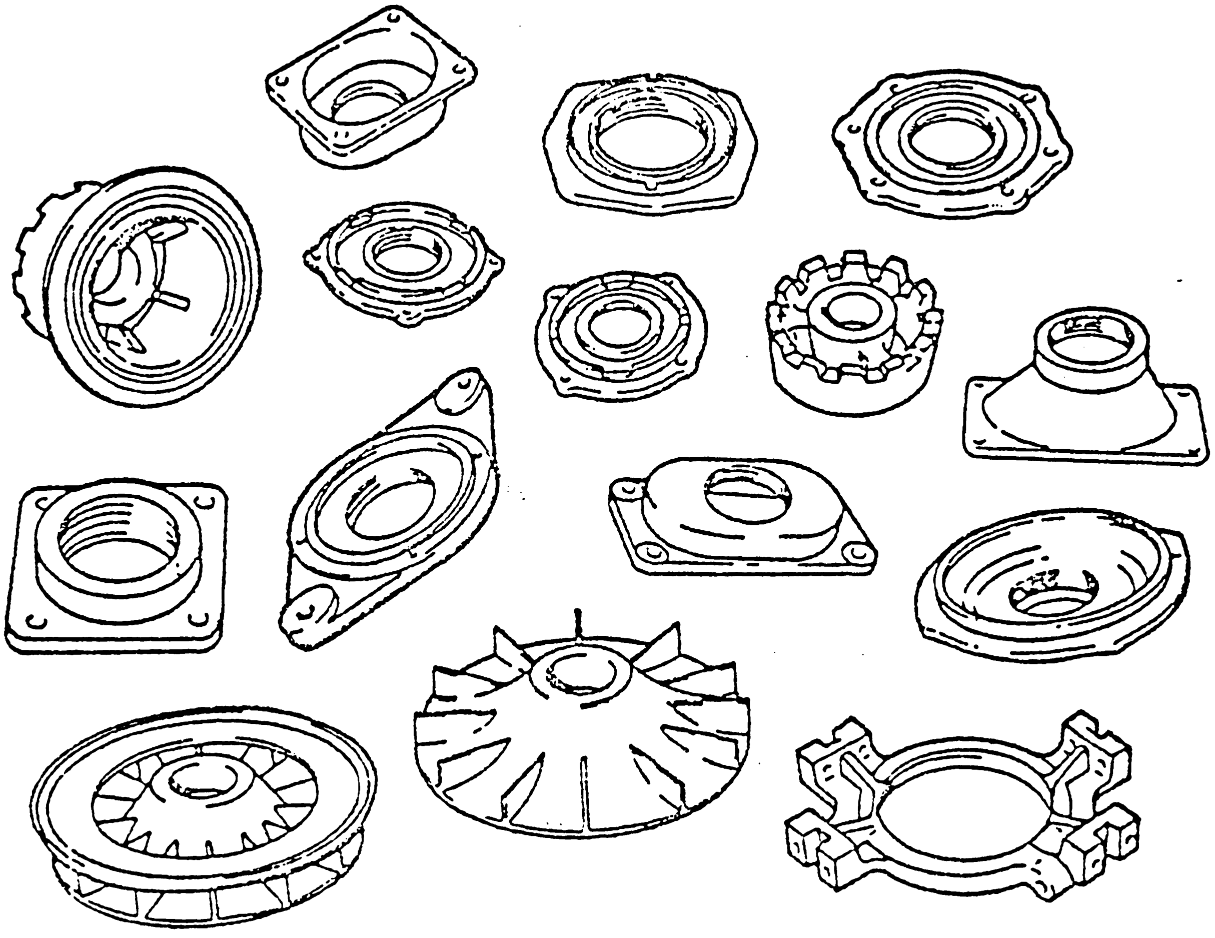


Figure 1.15 A Production Family (parts grouped based on Manufacturing Process)

Although both production flow analysis and group layout are based on coding and classification methods, they can still be distinguished as different activities. Since group layout is not "directly" related to process planning, they will not be discussed further in detail.

1.3.1 Coding and Classification

Coding is a process of establishing symbols to be used for meaningful communication. Classification is a separate process in which items are separated into groups based upon the existence or absence of characteristic attributes. Coding can be used for classification purposes, and classification requirements must be considered during the construction of a coding scheme. Therefore, the two topics are closely related.

Before a coding scheme can be constructed, a survey of all component features must be completed. Then, code values can be assigned to the features. The selection of relevant features is dependent on the application of the coding scheme. For example, tolerance is not important for design retrieval; therefore, it is not a feature in a design-oriented coding system. However, in a manufacturing-oriented coding system, tolerance is indeed an important feature.

Because the code structure affects its length, the accessibility and the expandibility of a code (and the related data base) is of importance. There are three different types of code structures in GT coding sytems. They are (1) hierachical, (2) chain, (matrix) and (3) hybrid.

A hierarchical structure is also called a monocode. In a monocode, each code number is qualified by the preceding characters. For example, in Figure 1.17, the fourth digit indicates threaded or not threaded for a 322x family. One advantage of a hierarchical structure is that it can represent a large amount of information with very few code positions. A drawback is potential complexity of the coding system. Hierarchical codes are difficult to develop because of all the branches in the hierarchy which must be defined.

A chain structure is called a polycode. Every digit in the code position represents a distinct bit of information, regardless of the previous digit. In Figure 1.17, a chain structured coding scheme is presented. A "2" in the third position always means a cross hole, no matter what numbers are given to positions one and two. Chain codes are compact, and are much easier to construct and use. The major drawback is that they cannot be as detailed as hierarchical structures with the same number of coding digits.

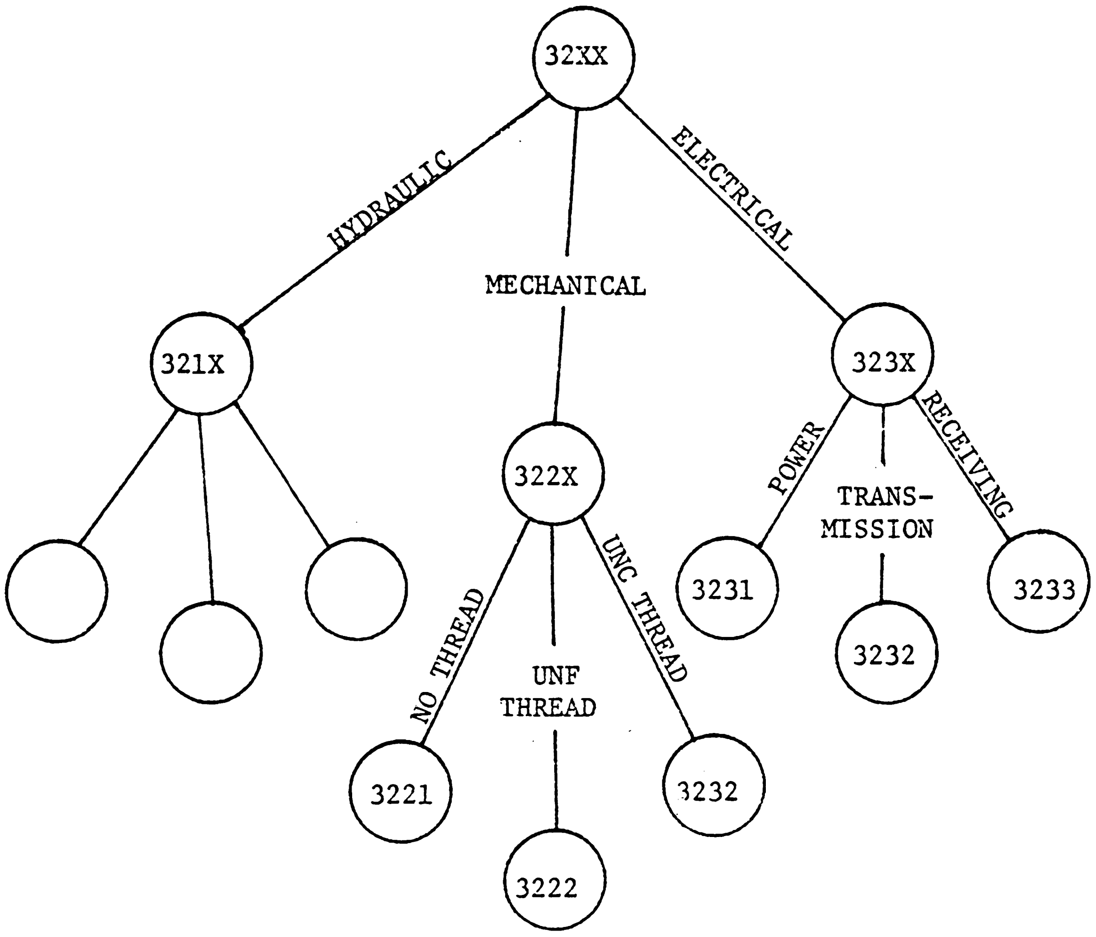


Figure 1.16 HIERARCHICAL STRUCTURE

DIGIT POSITION	1	2	3	4
CLASS OF FEATURE	EXTERNAL SHAPE	INTERNAL SHAPE	HOLES	
POSSIBLE VALVE 1	SHAPE 1	SHAPE 1	AXIAL	
2	SHAPE 2	SHAPE 2	CROSS	
3	SHAPE 3	SHAPE 3	AXIAL & CROSS	
4				

Figure 1.17 Chain Structure

The third type of structure, the hybrid structure, is a mixture of the hierarchical and chain structures (Figure 1.18). Most existing coding systems use a hybrid structure to obtain the advantages of both structures. A good example is the widely used Opitz code (Figure 1.19).

There are more than one hundred GT coding systems used in industry today. The structure selected is primarily based on the application. Table 1.2 provides a comprehensive guideline for code structure selection.

The physical coding of a component can be shown best by example. In Figure 1.20, a rotational component is coded using the Opitz system. Going through each code position, the resulting code becomes 01312. This code represents this component and all others with similar shape, and diameter. Later, in Chapter 3, GT applications to process planning will be discussed.

1.3.2 Closing Remarks

A brief review of manufacturing history coupled with a passing knowledge of today's economic situation should lead one to believe that U.S. manufacturing industries are facing a very difficult period in their evolution. In order to keep pace with rapidly developing international competion,

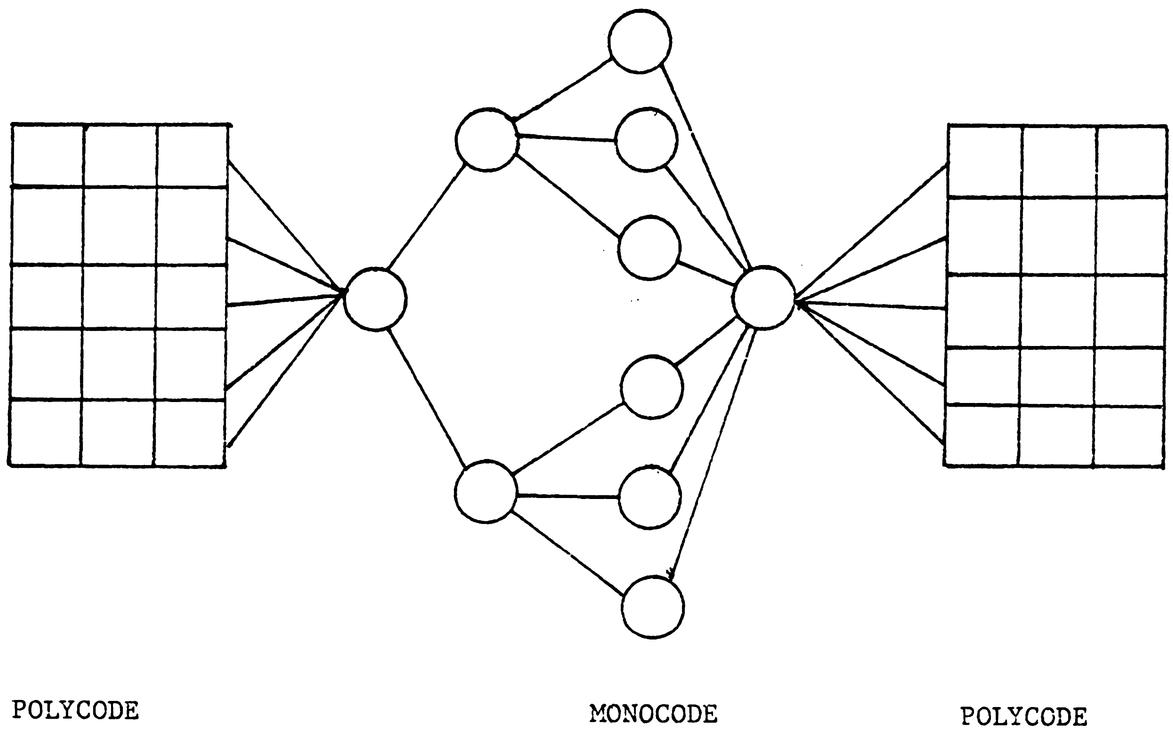


Figure 1.18 HYBRID STRUCTURE

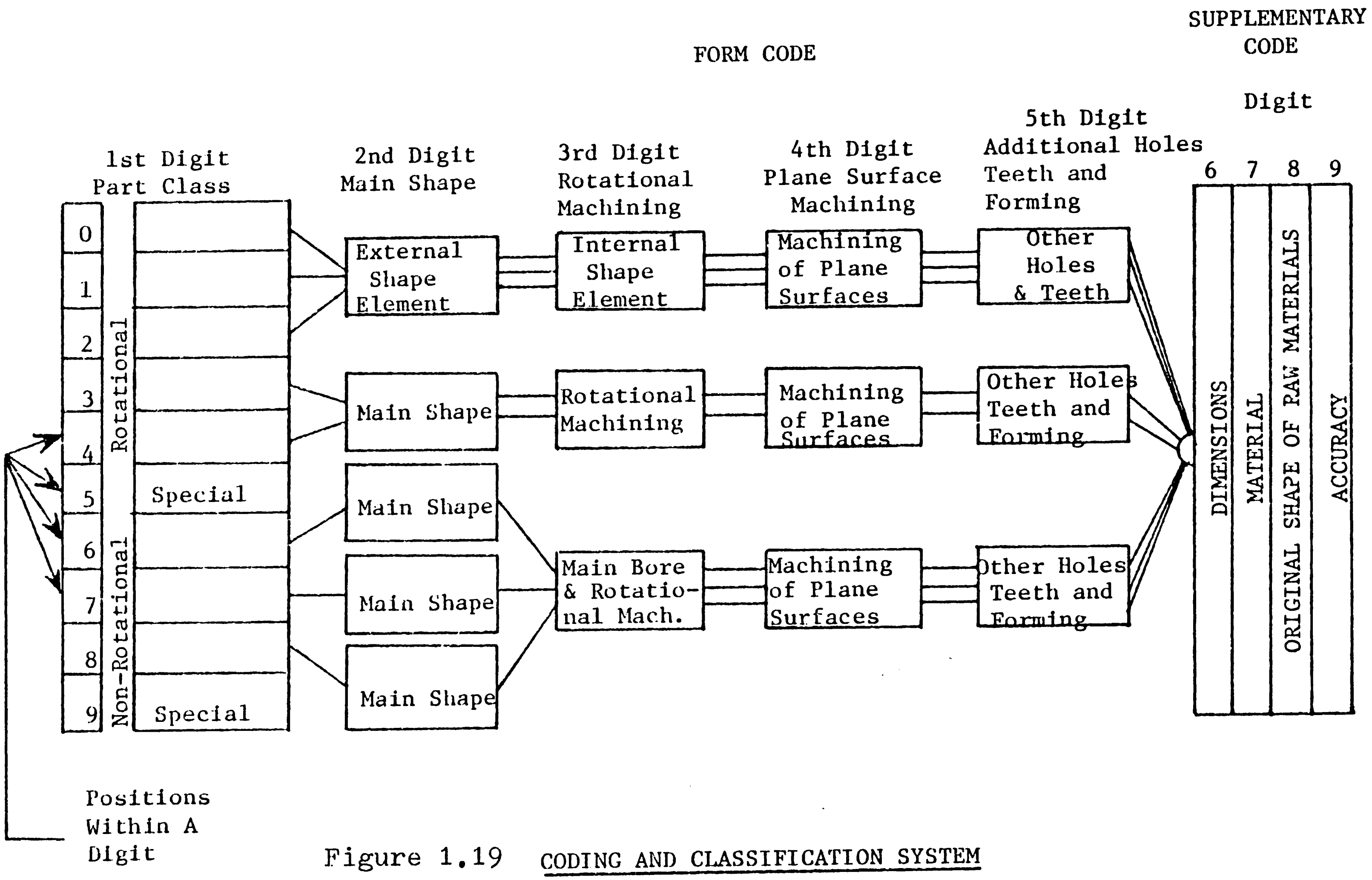

FORM CODE
SUPPLEMENTARY CODE
Digit
1st Digit Part Class
2nd Digit Main Shape
3rd Digit Rotational Machining
4th Digit Plane Surface Machining
5th Digit Additional Holes Teeth and Forming
6
7
8
9
0
1
2
3
4
5
6
7
8
9
Rotational
Non-Rotational
Special
Special
External Shape Element
Internal Shape Element
Machining of Plane Surfaces
Other Holes & Teeth
Main Shape
Rotational Machining
Machining of Plane Surfaces
Other Holes Teeth and Forming
Main Shape
Main Shape
Main Shape
Main Bore & Rotational Mach.
Machining of Plane Surfaces
Other Holes Teeth and Forming
DIMENSIONS
MATERIAL
ORIGINAL SHAPE OF RAW MATERIALS
ACCURACY
Positions Within A Digit

Figure 1.19 CODING AND CLASSIFICATION SYSTEM

TABLE 1.2

CODE STRUCTURE SELECTION

(Taken from Krag, W.B. "Toward Generative Manufacturing Technology" NCS Proc, 1978)

MAJOR ITEM CLASS	RESOLUTION REQUIRED	FLEXIBILITY NEEDED	CODE SYSTEM TYPES
Raw Materials	Moderate	Low	Hybrid (H/C)
Commercial Items	High	Low	H
Designed Piece parts	Moderate	High	C
Assemblies Models	Moderate	Moderate	Hybrid (H/C)
Machinery	Moderate	Moderate	Hybrid (H/C)
Technical Information	Moderate	Low	H
Tools-Commercial	Moderate	Low	H
Tools-Proprietary	Moderate	High	C
Guages/Fix-tures	Moderate	Low	H
Supplies	High	Low	H

H : Hierachical

C : Chain

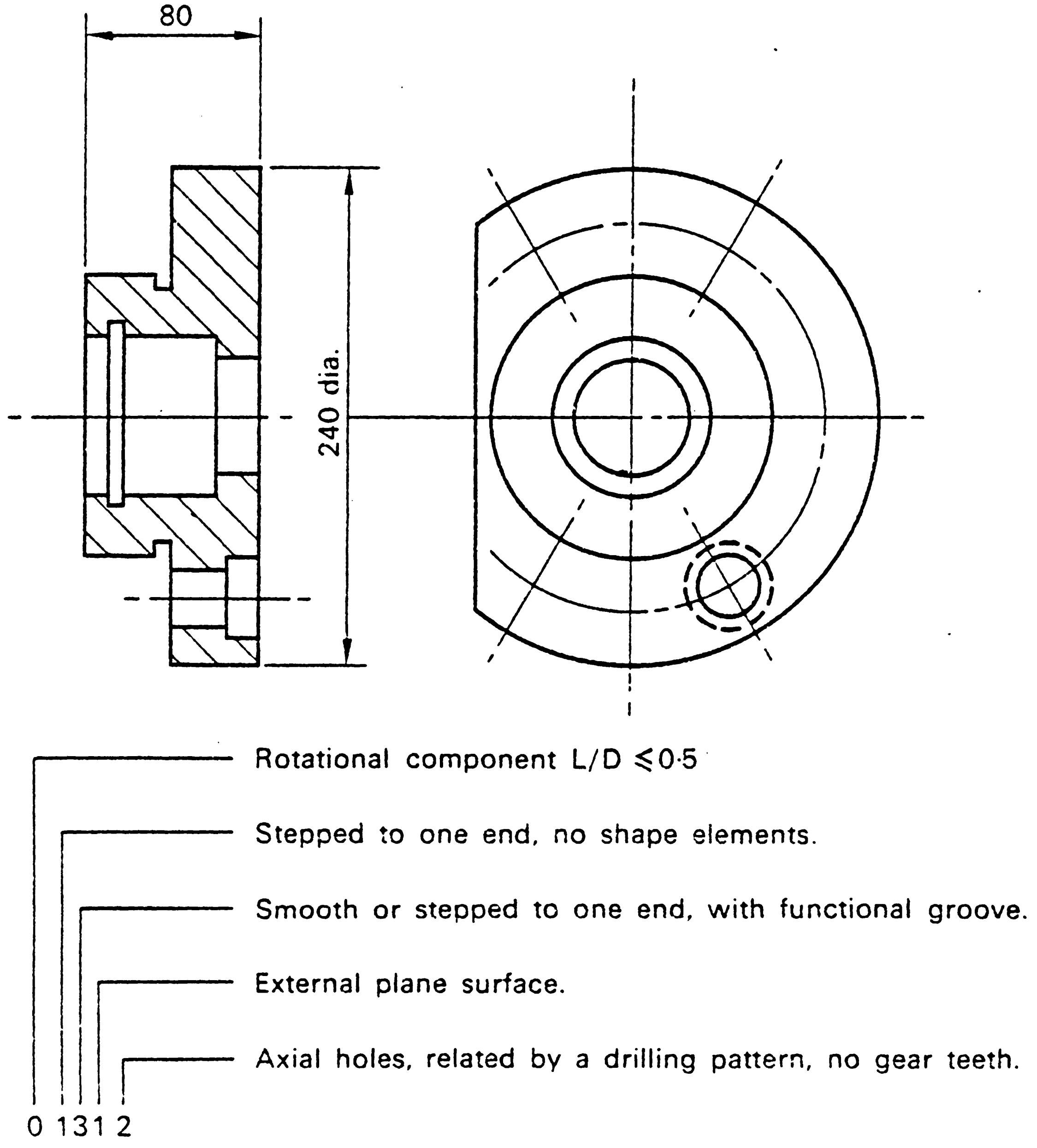

80
240 dia.
Rotational component L/D ≤0·5
Stepped to one end, no shape elements.
Smooth or stepped to one end, with functional groove.
External plane surface.
Axial holes, related by a drilling pattern, no gear teeth.
0 1 3 1 2

Figure 1.20 A Rotational Components Coded Using the Opitz System (taken from Opitz)

one must look to the most modern manufacturing technologies in order to keep pace with competition. CAD and CAM have become a "standard" for improving productivity. However, neither CAD nor CAM have proven to be answers by themselves. Integrating CAD and CAM appears to be the key to larger productivity gains. Furthermore, the basis for this integration is Automated Process Planning. The remainder of this thesis focuses on process planning technology and key integration issues which must be overcome before CAD/CAM can be a reality.

Chapter II

PART DESIGN REPRESENTATION

Process planning, like other planning activities, has a very specific objective -- to identify the detailed processes required for the production of a piece part. Product design is the input for the planning system. The Operations Planning Summary is the output. Although several different input representations are used to describe a part, the most general input representation is referred to as an engineering design. This representation is input to the planning system, which interprets it in order to find the appropriate machining processes, process parameters, etc. to satisfy the design. The knowledge/data base required for process planning must then be stored in an easily accessable storage medium, such as: human brain, handbooks, computer data base, etc.

In a computerized planning system, a formal structure and data base for this knowledge is required in order to transform the engineering design information into production design. In this chapter, information requirements for process planning systems will be discussed and ordered into a detailed data base.

2.1 <u>ENGINEERING DESIGN</u>

Engineering design is the partial realization of a designer's concept. The designer normally cannot directly tranform his concept into a physical item. Instead,the designer conveys his idea to other people through an alternate medium like an engineering drawing, and then a manufacturing engineer or machinist produces his design. Prior to our industrialization , when a farmer needed a tool for his farm, he normally went to a blacksmith and told the blacksmith the shape and size of the tool required. Because most tools were simple and did not require significant accuracy, a blacksmith could get a pretty good picture of a hoe or a plow by a verbal description from the farmer. If the blacksmith still could not understand, the farmer could roughly sketch what he wanted on the dirt floor of the blacksmith shop.

As the design became more complex, a picture of the design was necessary to relate the information to others. Multiview orthographic drawing has long been adopted by engineers as the standard tool to represent a design. Design information can be passed from the designer to others who are well trained in reading such drawings. The object a designer draws on paper can be interpreted and reconstructed into the designer's image in a viewer's mind. The capability of transforming an object from one medium to another (e.g. from a 2-D three view drawing to a 3-D picture in one's mind), and an understanding of the drawing rules is prerequisite to pass design information from the designer to other people (without error).

An "engineering drawing" therefore contains all the information necessary to represent a design (e.g., a component). Since process planning is the function which creates detailed plans for the production of a component, an engineer ing drawing is the basic input to the process planning function. Figure 2.1 illustrates the information flow for this process.

There are several methods available to represent an engineering drawing. The most conventional is drafting on paper with pencil or ink. Manual drafting is tedious and requires a tremendous amount of patience and time. Recently, computer-aided drafting systems have been implemented to improve drafting efficiency. The major objective of these systems is to assist the draftsman with tedious drawing and redrawing. Partially completed or completed drawings from a graphics tablet or screen can be digitized and stored in a computer, but retrieved when needed.

Virtually all CAD systems store drawings in a 3-D representation. Points (vertices), lines (edges), and curves are represented in X, Y, Z space. When a drawing is requested,

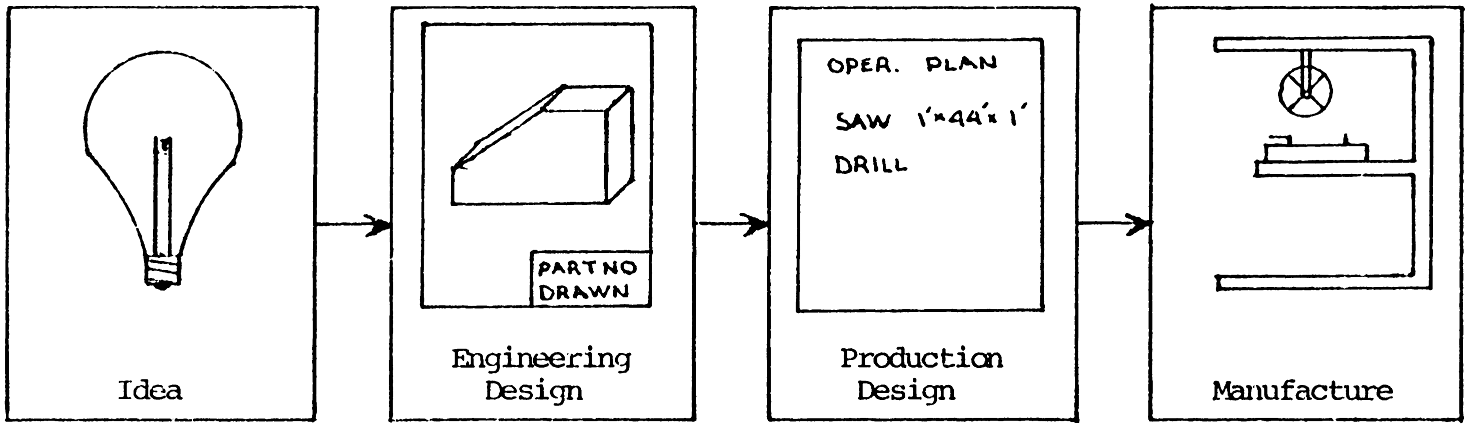


Figure 2.1 Evolution/Realization of a Product

a series of transformations are performed on the data, and a drawing is presented either in 2-D or 3-D perspective or sectional views. The resultant drawing can then be physically drawn using a plotter or simply displayed on a CRT. Such internal representations can be used not only for design drafting but also for engineering analysis, such as finite element (structural) analysis.

In recent years, the term geometric modeling in CAD has become common jargon. Geometric modeling is a technique for providing computer-compatible descriptions of the geometry of a part. Conventional CAD systems are geometric models which use surface oriented 3-D models. Recent advances in CAD have focused on the development of bounded shape models for 3-D representation. In these bounded shape models, individual surfaces are structured together to define the complete shell (boundary of a shape). Operators can then be applied to manipulate the shape.

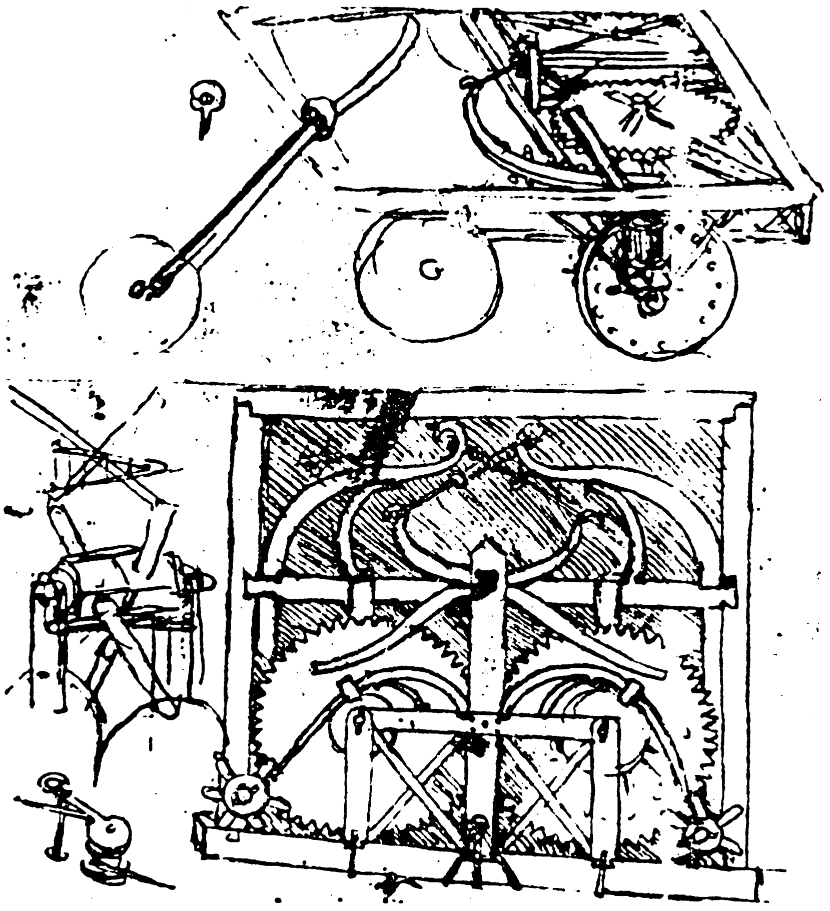
In the following sections, different design representations will be discussed.

2.2 DESIGN DRAFTING

Engineering drawing is a universal language used to represent a designer's idea to others. It is the most accepted medium of communication in all phases of industrial and engineering work. In ancient times before multiview drawing standards were adopted, perspective drawings were normally employed. The master of art during the Renaissance, Leonardo da Vinci, designed several machines and mechanical components (which still amaze contemporary designers) using perspective sketches (Figure 2.2). Today, pictoral drawings are still used to supplement other design representations.

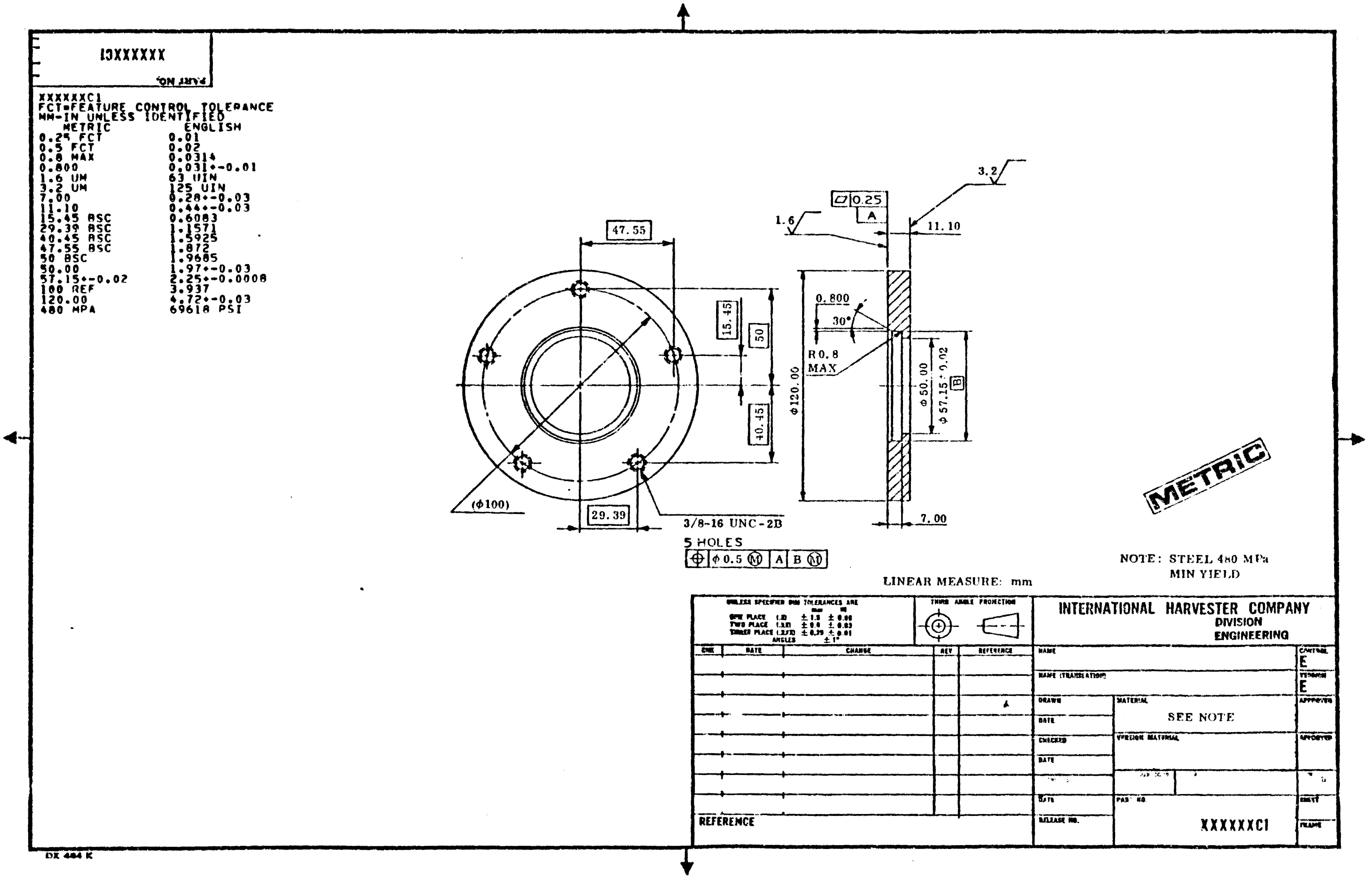
2.2.1 Multiview Drawing

In today's modern manufacturing industry, several types of drawings are acceptable. However, the standard is still the multiview drawing (Figure 2.3). A multiview drawing usually contains two or three views (front, top and side). Each view is an orthographic projection of one plane. In the United States and Canada, the third-angle projection is the system used. (See Figure 2.4). In Figure 2.4, the four quadrants of YZ plane are called I, II, III, and IV angles. For the third-angle projection, we always place the object in the third angle and project the object in three planes



The da Vinci "automobile" was to have been powered by two giant springs and steered by the tiller, at the left in the picture, attached to the small wheel.

Figure 2.2 Idea Sketch Prepared by Leonardo da Vinci (1452-1519).

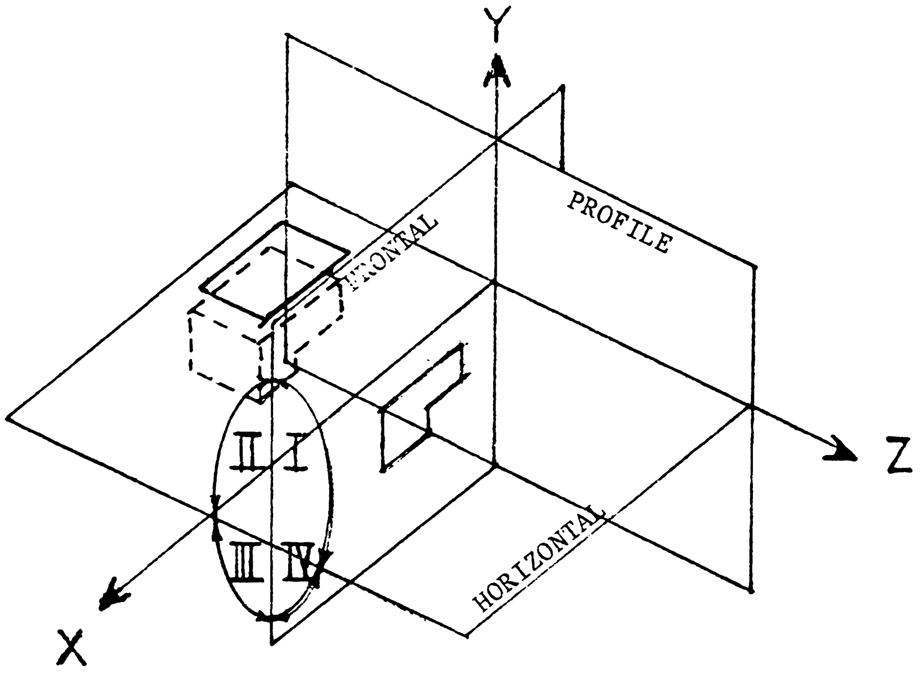


XXXXXXC1
FCT=FEATURE CONTROL TOLERANCE
MM-IN UNLESS IDENTIFIED

METRIC	ENGLISH
0.25 FCT	0.01
0.5 FCT	0.02
0.8 MAX	0.0314
0.800	0.031+-0.01
1.6 UM	63 UIN
3.2 UM	125 UIN
7.00	0.28+-0.03
11.10	0.44+-0.03
15.45 BSC	0.6083
29.39 BSC	1.1571
40.45 BSC	1.5925
47.55 BSC	1.872
50 BSC	1.9685
50.00	1.97+-0.03
57.15+-0.02	2.25+-0.0008
100 REF	3.937
120.00	4.72+-0.03
480 MPA	69618 PSI

Metric drawing. (*Courtesy International Harvester Company*)

Figure 2.3 A Multiview Drawing of a Bracket



A

The Third Angle Project

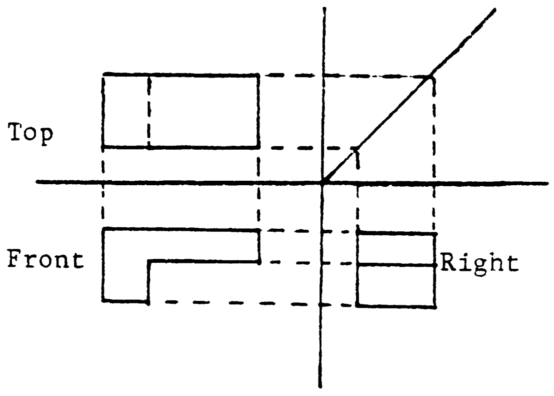


Figure 2.4 The Third Angle Projection

(looking toward the frontal plane (XY) and rotating the horizontal plane (XZ) on the X axis, (clockwise 90 degrees), the profile plan (YZ) on Y axis (clockwise 90 degrees) in order to obtain b).

2.2.2 Partial View

When drafting a symmetrical object, two views are sufficient to represent the object (typically one view is omitted). Sometimes a partial view is used to substitute one of the two views (Figure 2.5). Sectional and auxiliary views are also commonly used to present part detail. Sectional views (Figure 2.5 right) are extremely useful in displaying the detailed design of a complicated internal configuration. Casting designers often employ sectional views. When a major surface is inclined to all three projection planes, only a distorted picture can be seen. An auxiliary plane which is parallel to the major surface can be used to display an undistorted view.

2.2.3 Dimensioning

A drawing is expected to convey a complete description of every detail of a part. Dimensioning however is as important as the geometric information. In manufacturing, a drawing without dimensions is worth as much as the paper on which it is drawn.

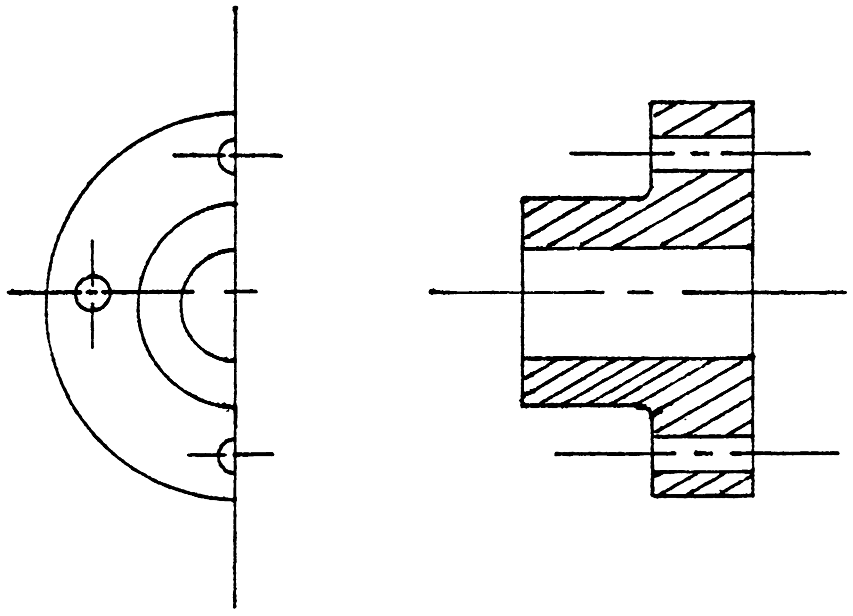


Figure 2.5 PARTIAL VIEWS

According to ASA Standards, the following are the basic rules which should be observed in dimensioning any drawing:

1. Show enough dimensions so that the intended sizes and shapes can be determined without calculating or assuming any distances.
2. State each dimension clearly, so that it can be interpreted in only one way.
3. Show the dimensions between points, lines or surfaces which have a necessary and specific relation to each other or which control the location of other components or mating parts.
4. Select and arrange dimensions to avoid accumulations of tolerances that may permit various interpretations and cause unsatisfactory mating of parts and failure in use.
5. Show each dimension only once.
6. Where possible, dimension each feature in the view where it appears in profile, and where its true shape appears.
7. Wherever possible, specify dimensions to make use of readily available materials, parts, tools and guages. Savings are often possible when drawings specify: (a) commonly used materials in stock sizes, (b) parts generally recognized as

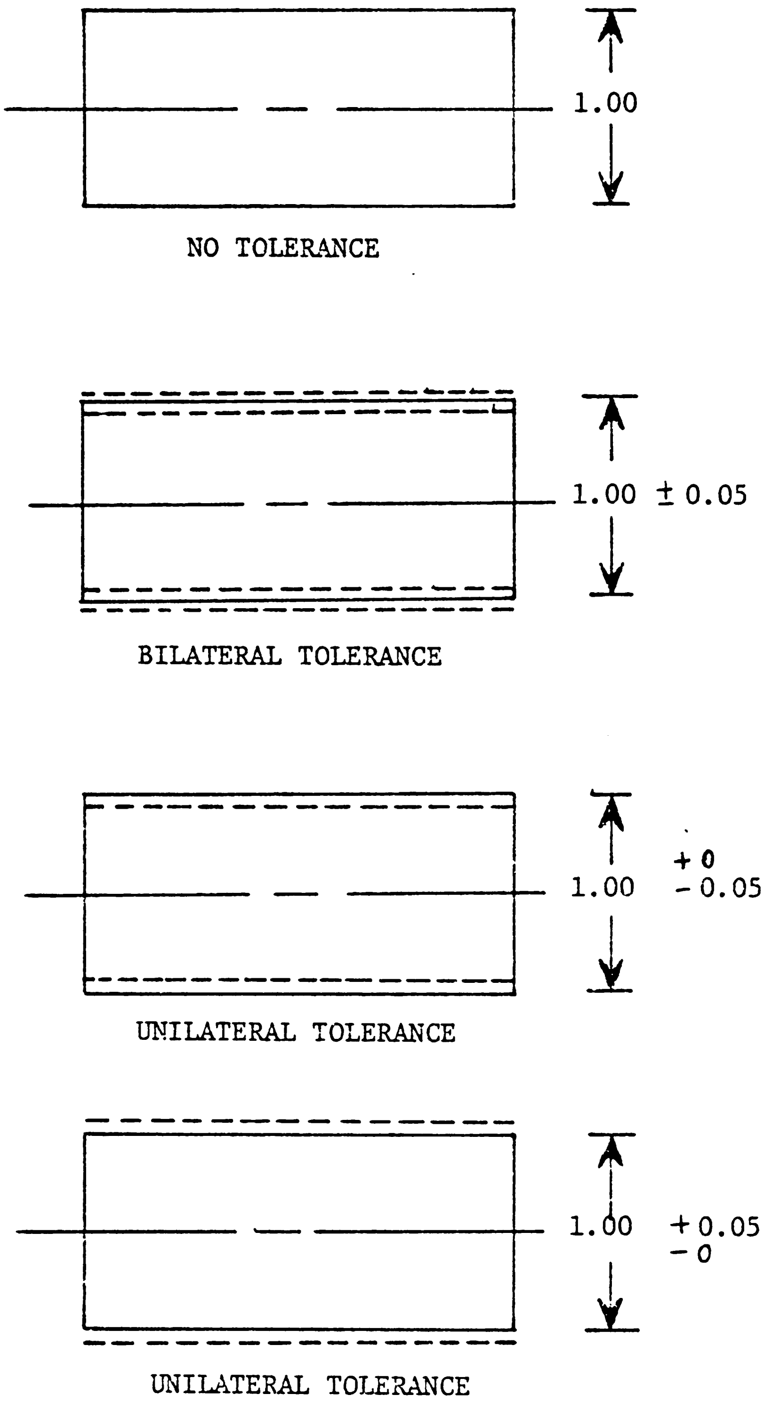
commerically standard, (c) sizes that can be produced with standard tools and inspected with standard gauges, and (d) tolerances from accepted published standards.

2.2.4 Conventional Tolerancing

Since it is impossible to produce the exact dimension specified, a tolerance is also used to show the acceptable variation in a dimension. here are two types of tolerances; bilateral tolerance and unilateral tolerance (Figure 2.6). Bilateral tolerances, such as 1.00 ± 0.05, specify dimensional variation from the basic size, i.e. $0.95 \sim 1.05$. On the other hand, a unilateral tolerance expresses an increase (or decrease) in one direction in relationship to the basic size, e.g. $1.00 ^{+0.00}_{-0.05}$ equals $0.95 \sim 1.0$.

The basic location where most dimension lines originate is the reference location. For example, in Figure 2.3, the center of the part on the top view, and the top surface in the side view are reference locations. For machining, the reference location provides the base from which all other measurements are taken. By stating tolerances from a reference location cumulative errors can be eliminated.

On a component there are working surfaces and non-working surfaces. Working surfaces are surfaces such as bearings,



(Dashed lines show the tolerance limits)

Figure 2.6 <u>Tolerancing - Bilateral and Unilateral</u>

pistons, and gear teeth for which optimum performance may require control of the surface characteristics. Non-working surfaces, such as the exterior walls of an engine block, crank case, or differential housings, seldom require any surface control. For surfaces which require surface control, a control surface symbol can be used. Figure 2.7 shows how the symbol is used.

In the symbol, several surface characteristics are specified. The roughness height is the roughness value as normally related to the surface finish. It is the average amount of irregularity above or below an assumed centerline. It is expressed in microinches (μ in) (0.000001 in.), or, in the metric system, in micrometers (μm) (0.000001 m). Recommended roughness heights can be found in Table 2.1. Lay is another property of a machined surface. It indicates the direction of the predominant pattern of surface irregularities produced by the tool. Lay symbols are listed in Figure 2.8. An example of using control surface symbols is shown in Figure 2.9.

Conventional methods of dimensioning provide only information concerning size and surface condition. A component can be produced without a guarantee of interchangeability. For example in Figure 2.10, both components (B) and (C) satisfy the dimension specified in

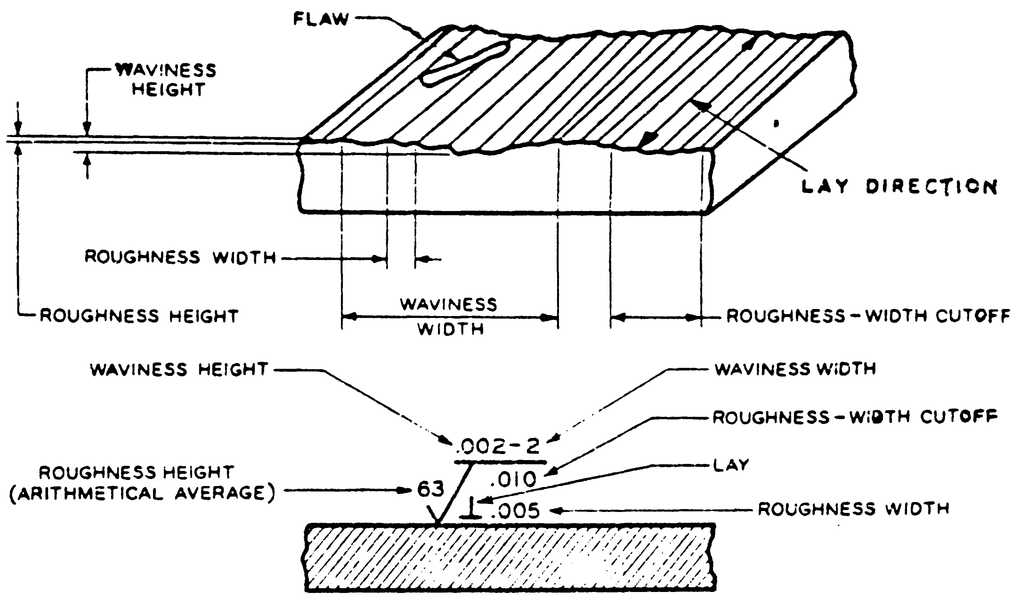


Figure 2.7 <u>Surface Control Symbols</u>

(A) (i.e., the diameter of the components (B) and (C) are 0.501" over the entire length of the component). Obviously both (B) and (C) are not desirable. However, as specified, both (B) and (C) meet specifications.

2.2.5 Geometric Tolerancing

Geometric tolerancing is the method to specify the tolerance of geometric characteristics. Basic geometric characteristics include:

straightness
flatness
roundness
cylindricity
profile
parallelism
perpendicularity
angularity
concentricity
symmetry
runout
true position

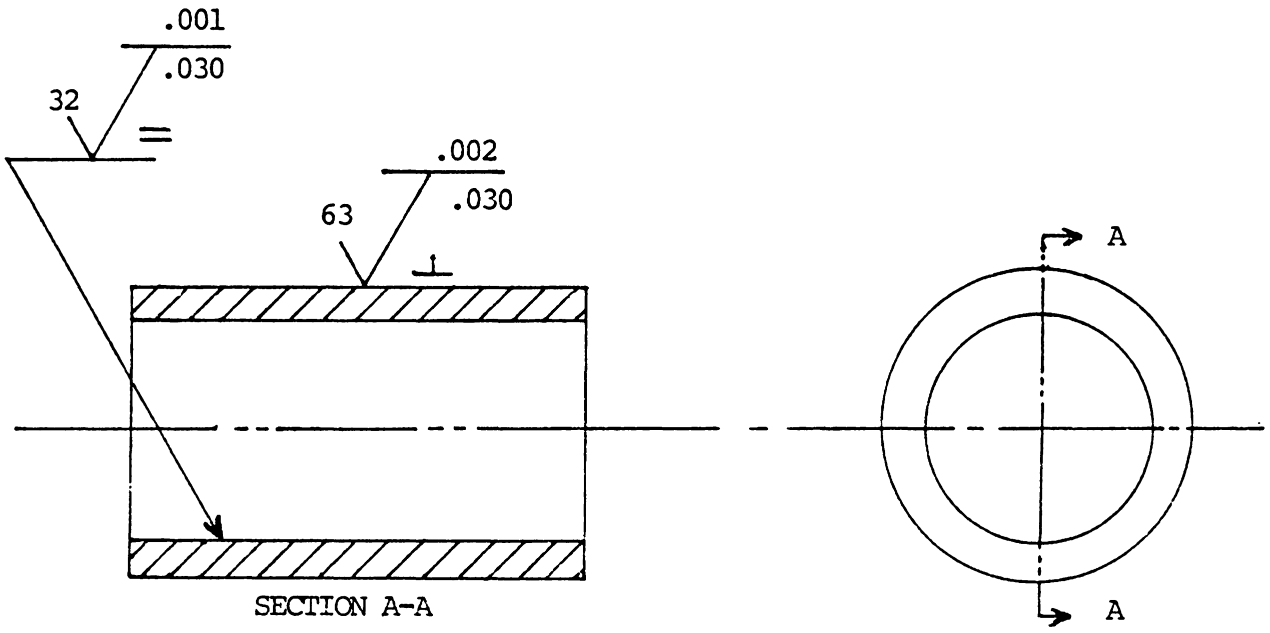
Symbols which represent these features can be found in Table 2.2. In order to specify the geometric tolerances, reference features either planes, lines or surfaces can be

Table 2.1 Recommended Height Values

Roughness value (Microinches)	Type of Surface	Purpose
1000	Extremely rough	Used for clearance surfaces only where good appearance is not required.
500	Rough	Used where vibration, fatigue or stress concentration are not critical and close tolerances are not required.
250	Medium	Most popular for general use where stress requirements and appearance are essential
125	Average smooth	Suitable for mating surfaces of parts held together by bolts and rivets with no motion between them.
63	Better than average finish	For close fits or stressed parts except rotating shafts, axles and parts subject to extreme vibrations.
32	Fine finish	Used where stress concentration is high and for such applications as bearings.
16	Very fine finish	Used where smoothness is of primary importance such as high-speed shaft bearings, heavily-loaded bearings and extreme tension members.
8	Extremely fine finish produced by cylindrical grinding, honing, lapping or butting	Use for such parts as surfaces of cylinder.
4	Super fine finish produced by honing, lapping, buffing or polising	Used on areas where packings and rings must slide across the surface where lubrication is not dependable.

$=$	Parallel to the boundary line of the nominal surface indicated by the symbol.
$\perp$	Perpendicular to the boundary line of the nominal surface indicated by the symbol.
X	Angular in both directions to the boundary line of the nominal surface indicated by the symbol.
M	Multi-directional
C	Approximately circular relative to the center of the nominal surface indicated by the symbol.
R	Approximately radial relative to the corner of the nominal surface indicated by the symbol.

Figure 2.8 Lay Symbols



<u>INTERPRETATION:</u>

Roughness Height (O D)	63μin
Roughness Height (I D)	32μin
Roughness-Width Cutoff (O D and I D)	.030
Waviness Height (O D)	.002
Waviness Height (I D)	.001
Lay (O D)	Circumferential
Lay (I D)	Axial

Figure 2.9 Application and Interpretation of Surface Roughness Symbols

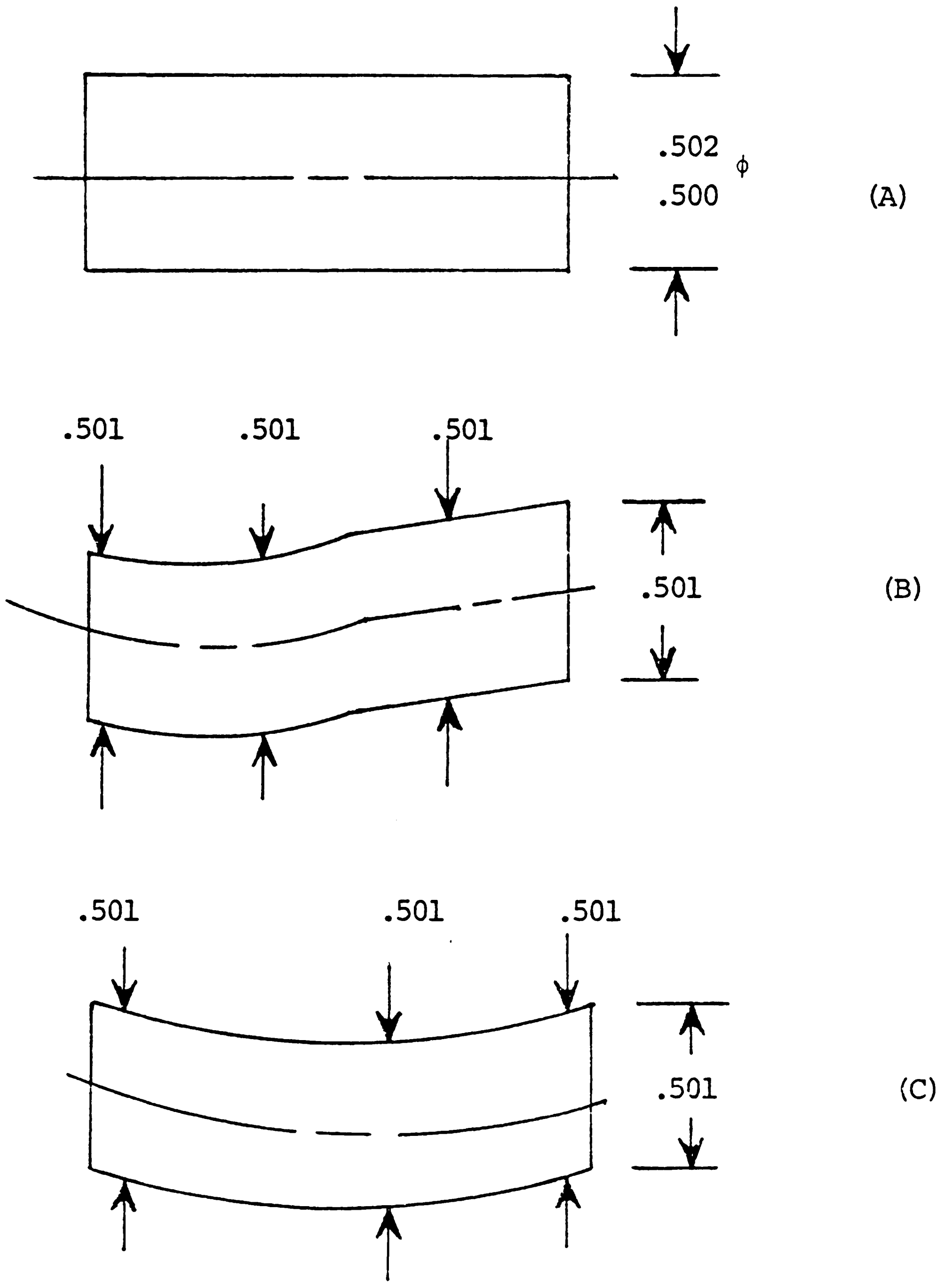


Figure 2.10 An Illustration of Some Addition Part Conditions

established. A datum is a plane, surface, point(s), line, axis or other source of information concerning an object. Datums are assumed to be exact, and from them dimensions similar to the reference location dimensions in the conventional drawing system can be established. Datums however are used only for geometric dimensioning.

Symbolic modifiers are used to clarify implied tolerances. Maximum material condition (MMC) can be used to constrain the tolerance of the produced dimension and the maximum designed dimension. It can be defined as the condition of a part feature where the maximum amount of material is contained. For example, maximum shaft size and minimum hole size can be illustrated as shown in Figure 2.11. Least material condition (LMC) specifies the opposite of the maximum material condition. They can be applied only when both of the following conditions can hold.

1. Two or more features are interrelated with respect to the location or form (e.g. two holes). At least one of the features must refer to size.
2. MMC or LMC must directly reference a size feature.

When MMC or LMC are used to specify the tolerance of a hole or shaft, it implies that the tolerance specified is constrained by the maximum or least material condition as well as some other dimensional features. The tolerance may

Table 2.2 Geometric Tolerancing Symbols

⏤	Straightness	⟂	Perpendicularity
⏥	Flatness	∠	Angularity
○	Poundness	◎	Concentricity
⌭	Cylindricity	⌯	Symmetry
⌒	Profile of a line	↗	Runout
⌓	Profile of a surface	⌖	True Position
∥	Parallelism		

GEOMETRIC CHARACTERISTIC SYMBOLS

Ⓜ	MMC, Maximum Material Condition
Ⓢ	RFS, Regardless of Feature Size
Ⓛ	LMC, Least Material Condition

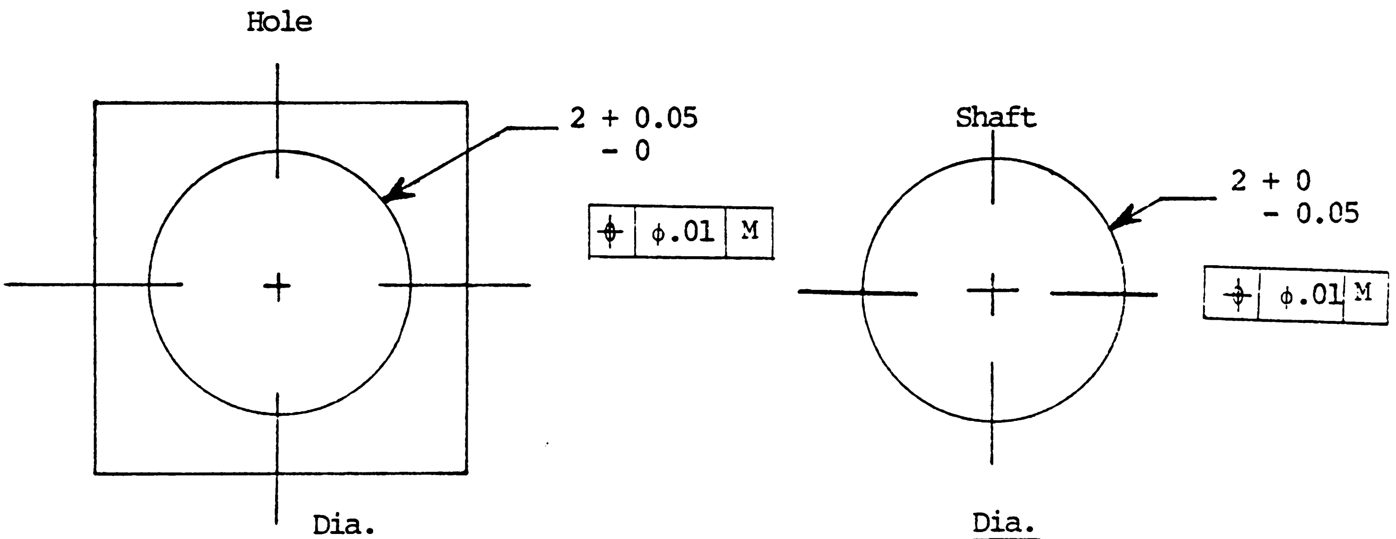
MODIFIERS

$\boxed{-A-}$	Datum A

DATUM IDENTIFICATION

Ⓟ	Projected Tolerance Zone
∅	Diameter

SPECIAL SYMBOLS



	Dia.	Dia.
MMC	2	2
LMC	2.05	1.95

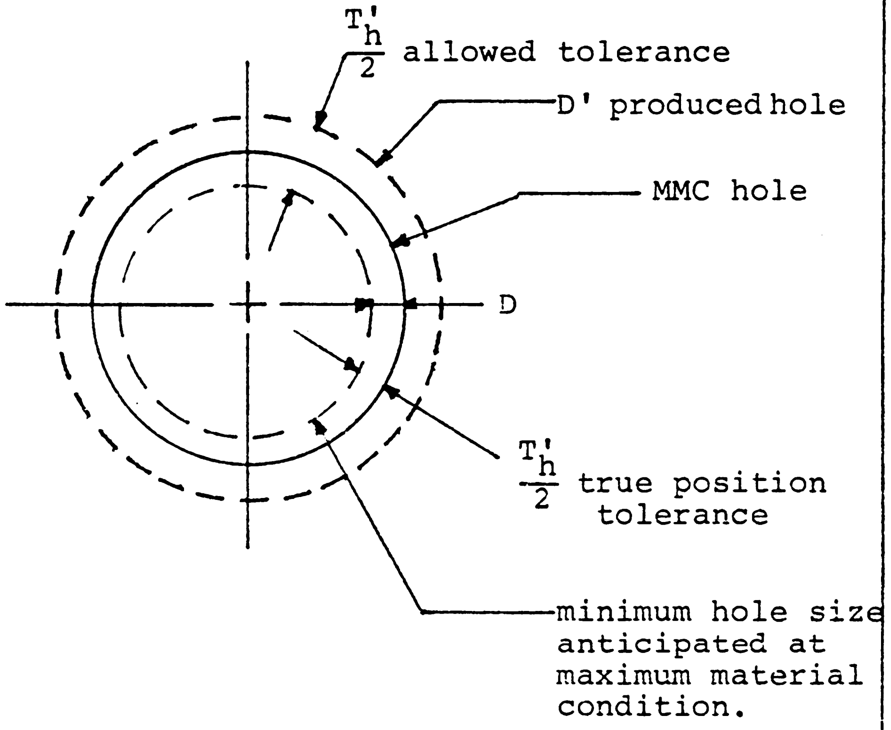
Figure 2.11 Maximum Material Diameter and Least Material Diameter

increase when the actual produced feature size is larger (for hole) or smaller (for a shaft) than the MMC size. Because the increase in the tolerance is compensated by the deviate of size in production, the final combined hole size error and geometric tolerance error will still be larger than the anticipated smallest hole (see Figure 2.12).

The third modifier is Regardless of Feature Size (RFS). It is also the default modifier. When RFS is used, the tolerance does not alter the produced feature. Table 2.3 shows the application of three modifiers applied to the hole and shaft shown in Figure 2.11.

Figure 2.13A -2.13M illustrates the use of form geometry symbols and their meeting. In all of the examples (except true position), RFS is assumed. The first drawing in each group of drawings represents the original drawing. The second drawing illustrates the interpretation of the geometric tolerance specified. All variations on surfaces have been exagerated.

In Figure 2.13A (where straightness is illustrated), straightness defines the maximum deviates on the assumed center line over the entire length of a cylindrical component. It is useful in specifying the fit of shafts and holes. Flatness (Fig. 2.13B) is the maximum deivation allowed on a flat surface. It is important for plane surface


$\frac{T_h'}{2}$ allowed tolerance
D' produced hole
MMC hole
D
$\frac{T_h'}{2}$ true position tolerance
minimum hole size anticipated at maximum material condition.

$$D' - T_h' \geq D - T_h$$

$$T_h' \leq T_h + (D' - D)$$

Figure 2.12 Allowed tolerance under the produced hole size

Table 2.3 Maximum out-of-true-position allowable under thee modifiers

Produced Size	Maximum out of true position allowable					
	HOLE			SHAFT		
	MMC M	LMC L	RFS S	MMC M	LMC L	RFS S
1.95				0.06	0.01	0.01
1.96		out-of-diameter		0.05	0.02	0.01
1.97		tolerance range		0.04	0.03	0.01
1.98				0.03	0.04	0.01
1.99				0.02	0.05	0.01
2.00	0.01	0.06	0.01	0.01	0.06	0.01
2.01	0.02	0.05	0.01			
2.02	0.03	0.04	0.01		out-of-diameter	
2.03	0.04	0.03	0.01		tolerance range	
2.04	0.05	0.02	0.01			
2.05	0.06	0.01	0.01			

Hole: $\phi\ 2\ ^{+0.05}_{-0}$ $\boxed{\oplus}\boxed{\phi}\boxed{0.01}\boxed{M}$

Shaft: $\phi\ 2\ ^{+0.05}_{-0}$ $\boxed{\oplus}\boxed{\phi}\boxed{0.01}\boxed{M}$

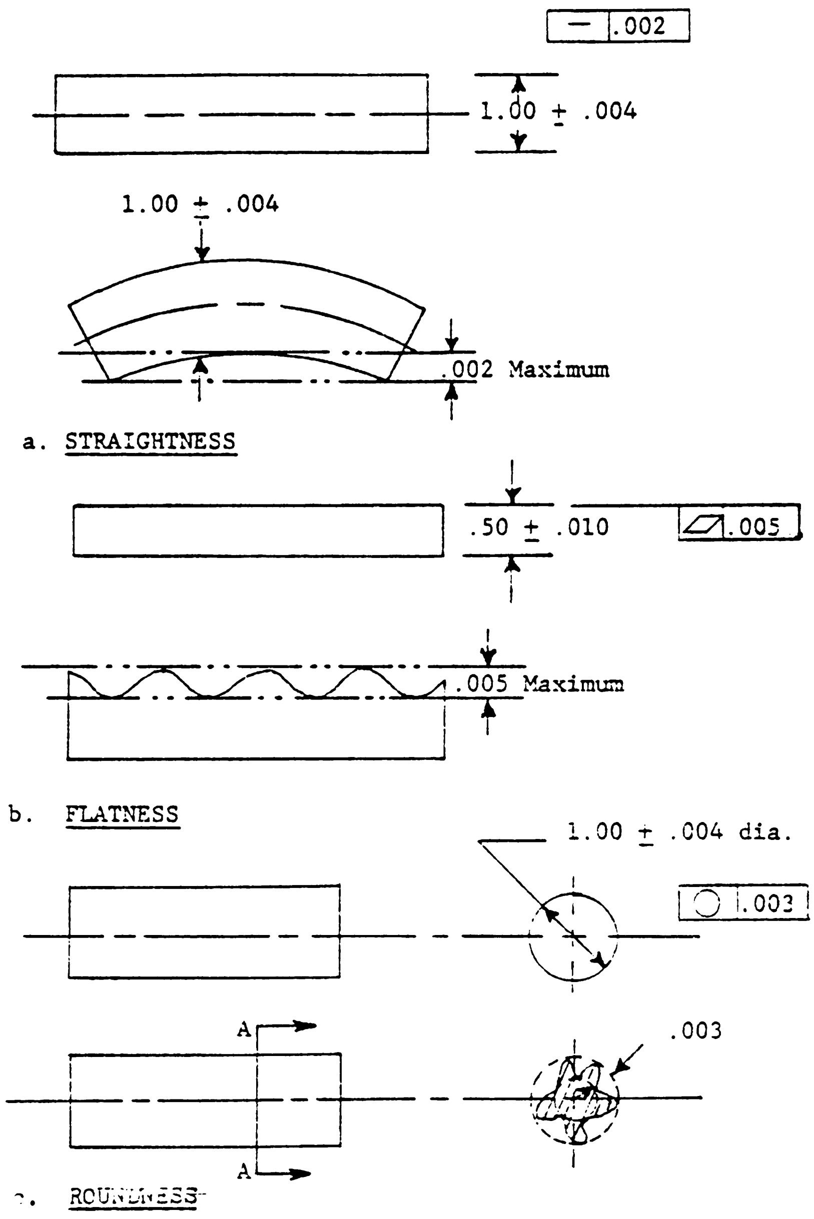
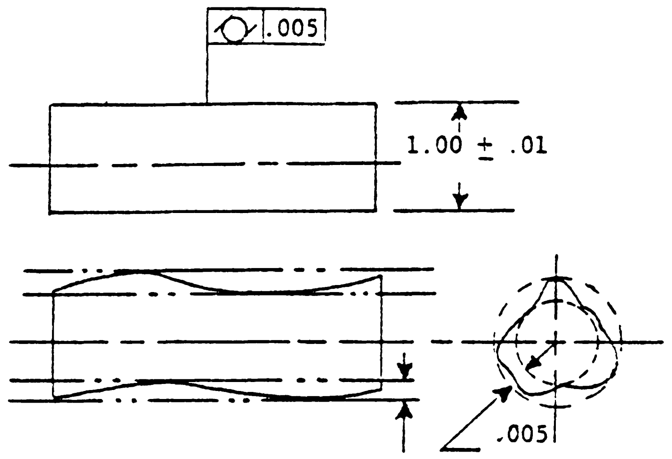
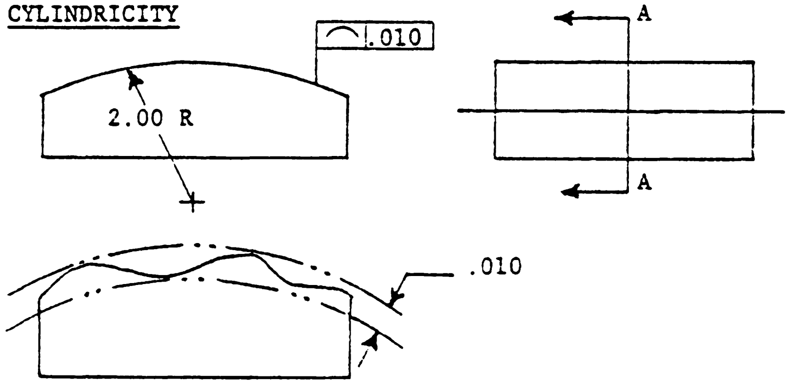


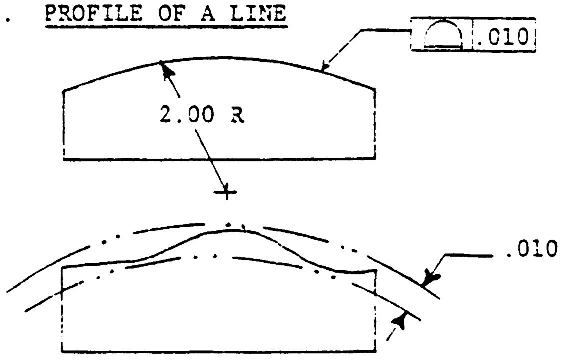
Figure 2.13 Illustration of Form Geometry Symbols



d. CYLINDRICITY

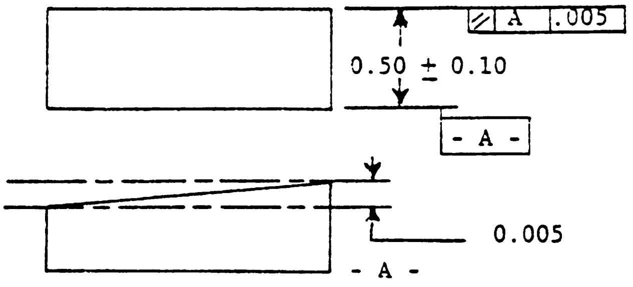


e. PROFILE OF A LINE

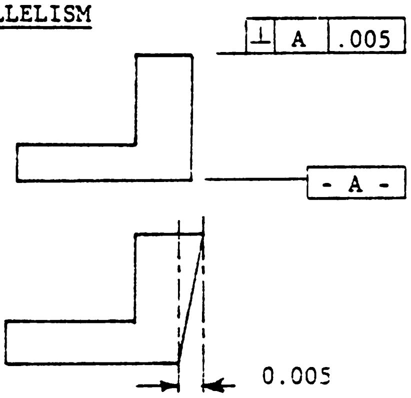


f. PROFILE OF A SURFACE

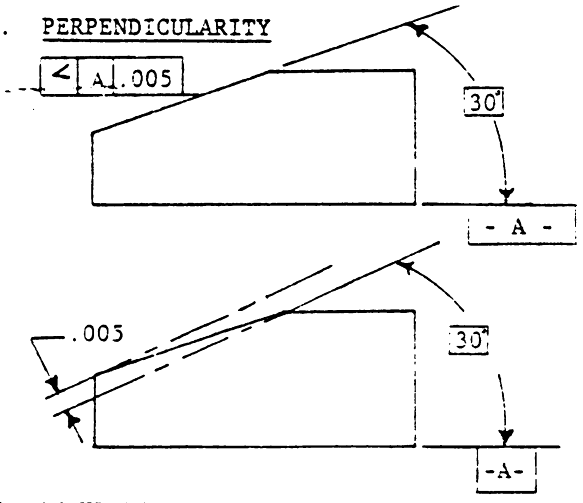
Figure 2.13 (Continued)



g. PARALLELISM

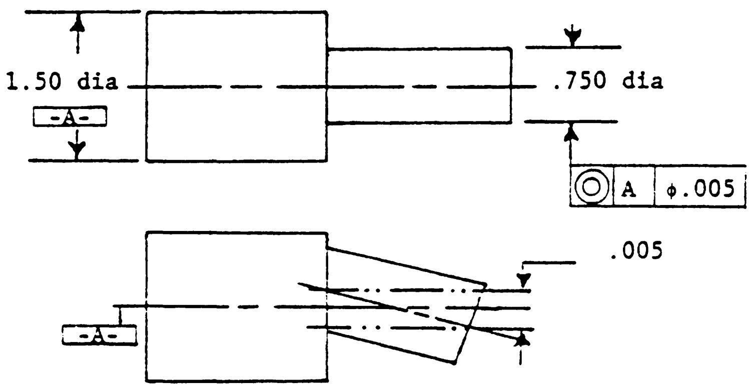


h. PERPENDICULARITY



i. ANGULARITY

Figure 2.13 (continued)



j. CONCENTRICITY

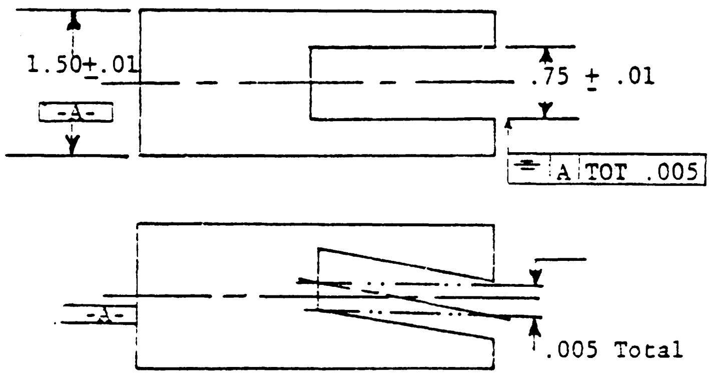
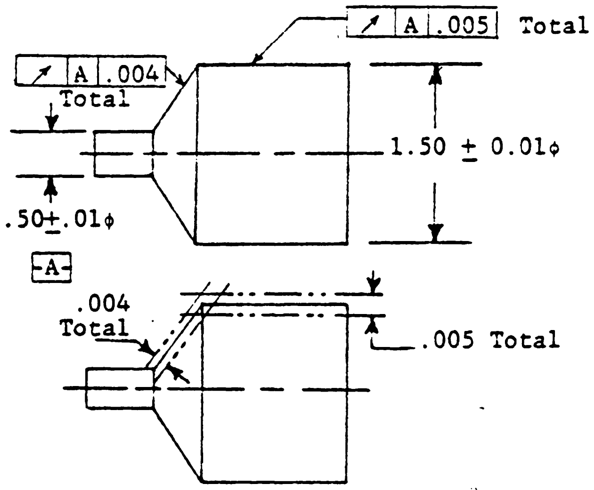
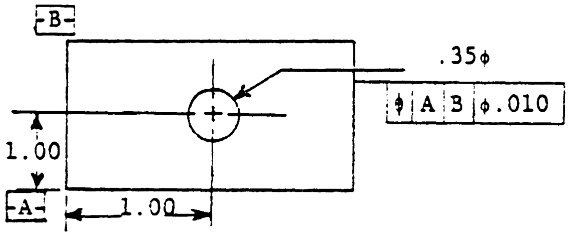


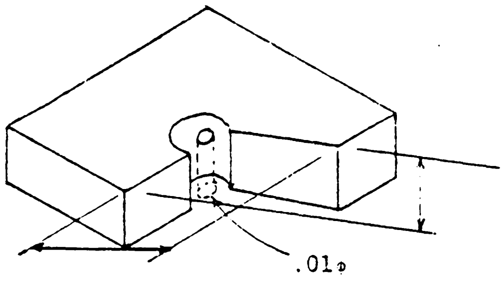
Figure 2.13 (continued)

k. SYMMETRY



l. RUNOUT





m. TRUE POSITION

Figure 2.13 (continued)

fit (e.g. gasket surfaces). Roundness (Figure 2.13C) defines the irregularity of the diameter at any given cross sectional location of a cylindrical component. Cylindricity (Figure 2.13 D) is similar to roundness except it defines the irregularity over the entire length. Again, these symbols are useful in specifying the fit for shafts and holes. Profile of a line (Figure 2.13E) and profile of a surface both describe the deviation on the profile, except profile of a line focuses on any cross sectional location and profile of a surface looks at the entire surface. Any feature of a component can be specified as being parallel to any given datum. Figure 2.13 G shows the use of the parallelism symbol.

Perpendicularity (Figure 2.13H) defines the tolerance of a feature which is 90 degress to a given datum. Angularity (Figure 2.13I) is similar to perpendicularity except the relationship of a feature to a given datum need not be 90 degrees. The axis of a hole or cylindrical object can also be dimensioned with angularity. Concentricity (Figure 2.13J) is used to establish a relationship between the axes of two or more cylindrical parts of a component. Symmetry (Figure 2.13K) is a counterpart of concentricity for planes. Runout (Figure 2.13L) is the composite deviation from the desired form of a rotational part during full rotation (360 degrees) of the part on a datum axis.

True position (Figure 2.13M) expresses the location of the center line with respect to a feature. Conventional tolerancing methods put tolerance area which is greater than the round tolerance area true position (⌖) specifies.

Thus far, basic drafting methods and symbols have been discussed. Using this knowledge of basic engineering geometry, dimensioning and tolerancing symbols, any proper engineering design drawing can be interpreted precisely.

2.3 COMPUTER-AIDED DESIGN

Computer-aided design (CAD) can be most simply described as "using a computer in the design process." A wide range of applications can qualify as CAD (from engineering design analysis to drafting). Some narrower views of CAD have been taken in order to confine it to computer graphics applications in the design process. Some examples of CAD applications include: mechanical part design, IC (Integral circuit) chip layout, architectural design, etc. Because of the nature of this text, the major focus will be placed on mechanical part design (input to process planning). Only drafting and geometric modeling of mechanical parts will be discussed. In this text, the term CAD is equivalent to computer-aided drafting and the related geometric modeling of mechanical parts.

The development of CAD originated from early computer graphic systems. CAD first appeared in the late 1950's when a computer was first used to create graphic objects on a CRT (Cathode Ray Tube). This coincidently was about the same time as NC and APT (Automated Programning Tool) first appeared. A bit later X-Y plotters were used as hard copy output devices. An interesting note is that an X-Y plotter has the same basic structure as an NC drilling machine except a pen is substituted for the tool on the NC spindle. Early CAD systems basically focused on improving the productivity of draftsmen. However, the more recent CAD systems have focused on modeling engineering objects. This approach is not limited to drafting, but also includes APT programming, engineering analysis, etc.

2.3.1 CAD I/O Devices

Figure 2.14 depicts the typical input/output (I/O) devices of a CAD system. Input devices are generally used to transfer information from a human or storage medium to a computer where "CAD functions" are carried out. A keyboard is the standard input device used to transmit alphanumeric data to the system. In some systems, function keypads are also used to make input easier. Joysticks, trackballs, and mouses are also used to manipulate a cursor. They can manipulate the graphic cursor (e.g. crosshair) on a CRT and feedback the location of an object on the CRT to the computer. Using these devices allows an operator to interactively address terminal locations. A light pen can also be used for this type of interactive graphics (locational information is fedback discretely to the computer).

There are two basic approaches for the input of an existing drawing: 1) model the object on a drawing, or 2) digitize the drawing. Digitizing is usually much easier then modeling the object. A digitizer is a device which translates the X, Y locations on a drawing into a digital signal and feeds them to a computer. Finally our last input device is a sketch pad called a Graphics Tablet. A graphics tablet is a special flat surface on which a user draws with a stylus. The location of the stylus is sent to a computer.

The standard output device for CAD is a CRT display. There are two major types of CRT displays: a random-scan-line-drawing display and raster-scan display. Random-scan-line-drawing display uses Direct-View-Storage-Tube (DVST) as well as refresh tubes for displaying pictorial representations. On a DVST, a line written onto the screen will remain visible for some period of time (the tube itself stores the line in its memory). A refresh CRT does not continuously display a line on its tube. Rather, the line disappears

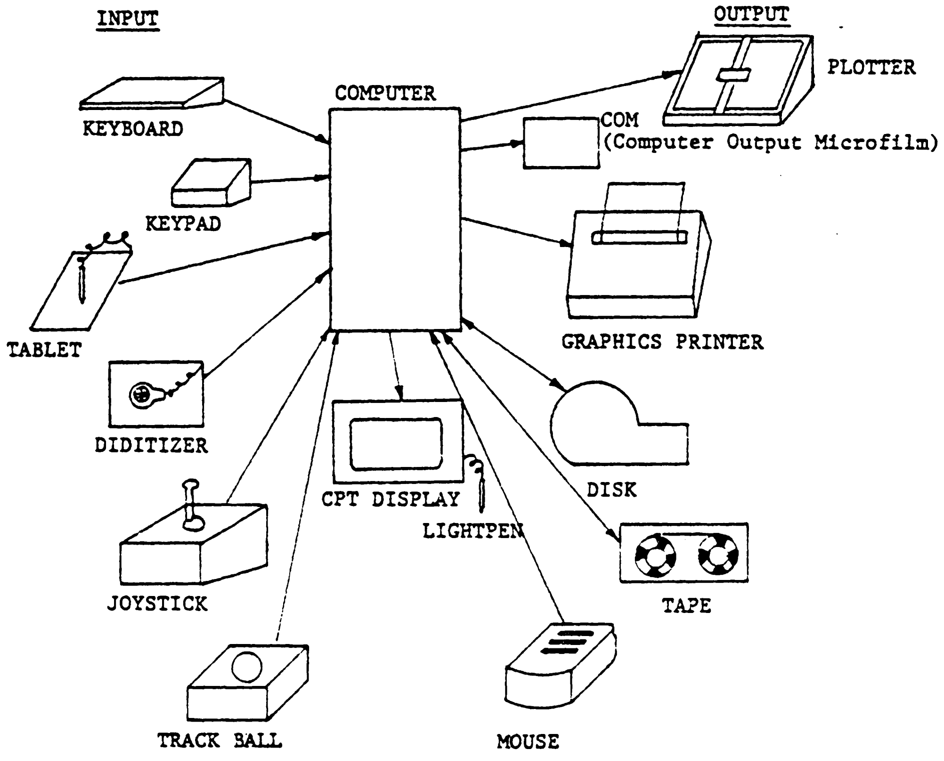


Figure 2.14 Peripherals

right after it is drawn. Therefore, information on the screen must be redrawn very quickly, so that the human eye will have the illusion of continuity of the picture. Although the redraw speed is restricted (lower limit) by the number of lines that can be displayed without the screen appearing to flicker, the partial erasing capability of a refresh CRT display makes it more attractive than DVST.

A raster-scan display differs from the random-scan line-drawing display in the method of scanning. The scanning method of a Raster-scan display is similar to a TV. Frequent scans are conducted from the top line of the screen to the bottom line. A frame buffer made of RAM (Random Access Memory) is designated to correspond to one frame of picture on the screen. Each pixel (addressable dot on the screen) is controlled by a byte or a half byte in the frame buffer. During the scan, a pixel can either be: 1) Turned on/off or 2) change color, or 3) change intensity. Picture images are written into a frame buffer rather than directly onto the screen. Some advantages of the raster-scan display include: 1) a more continuous picture tone can be generated and 2) the cost is usually lower than for random-scanning displays. However the resolution of the display is also lower. Although the method of display is important, the display does not affect the computer graphics technique [Newman 79].

2.3.2 Modeling Objects

A component must be modeled before it can be drawn. In a conventional drafting system, a component is modeled using simple geometry (Figure 2.15). A model can also be configured using an APT-like language to define geometric features. This type of model is stored in a data structure, e.g. the structure in Figure 2.16 [Brand 1973]. The drawing (Figure 2.17) obtained using such a system is basically the same as a draftsman would create.

Modeling a component using simple two-dimensional geometry (i.e. points, lines, circles, curves) is not sufficient for engineering drawing (informatiion concerning their relationships is not represented). Any object can be modelled using a set of primitives. A convenient primitive is a polyhedron (Figure 2.18). A polyhedron is a bounded object with n faces. Each face is a planar polygon (Figure 2.19) which can be modeled by using an ordered list of the vertices (points) of the polygon or by a similar list of its edges(lines). Wire-frame displays can then be generated using the edges of the polyhedron. The hidden-lines or hidden-surfaces cannot; however, be removed from the display without defining faces on the object. It is the face that hides the edges not the other edges.

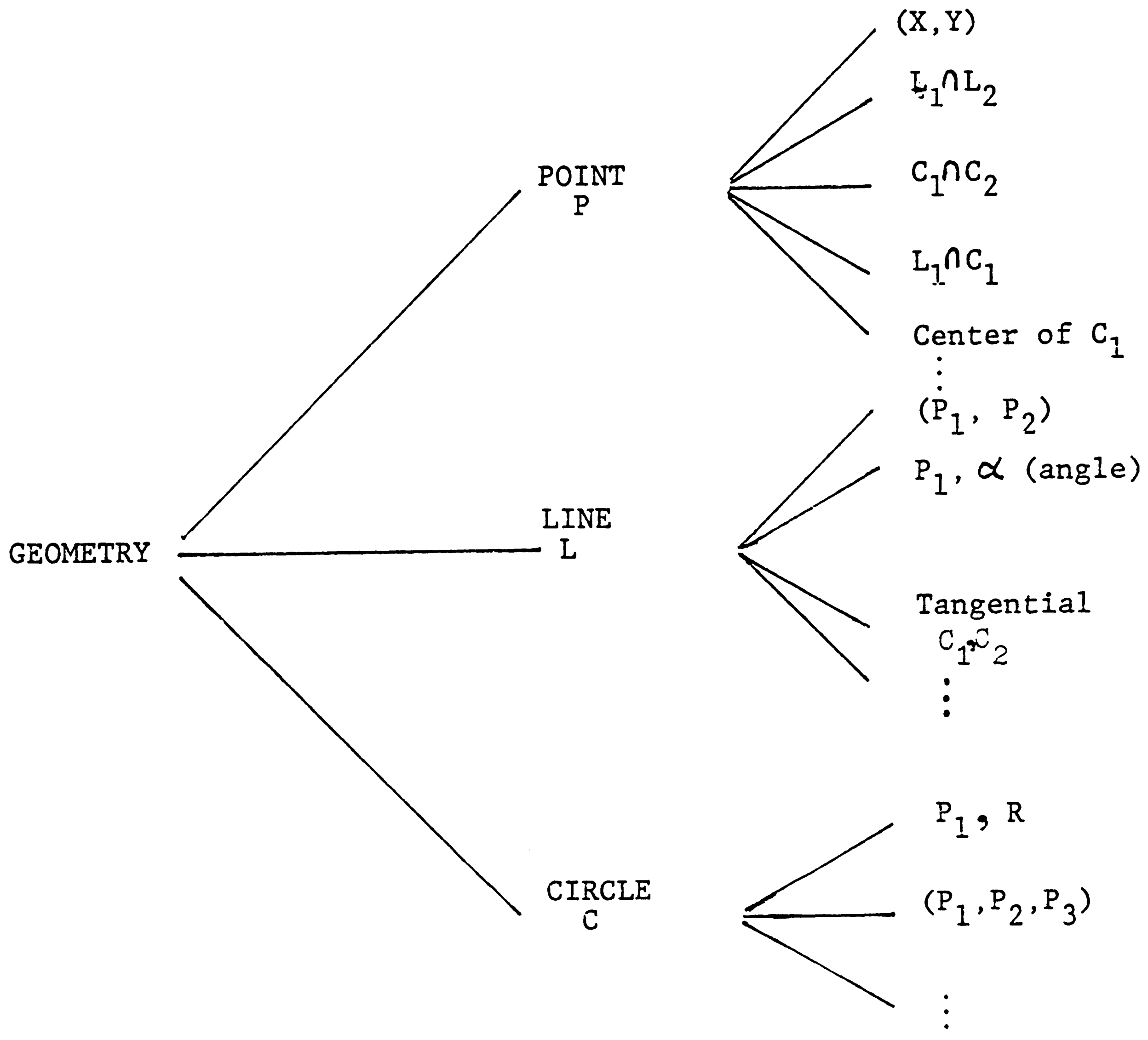


Figure 2.15 A Subject of Geometric Definitions for Drafting

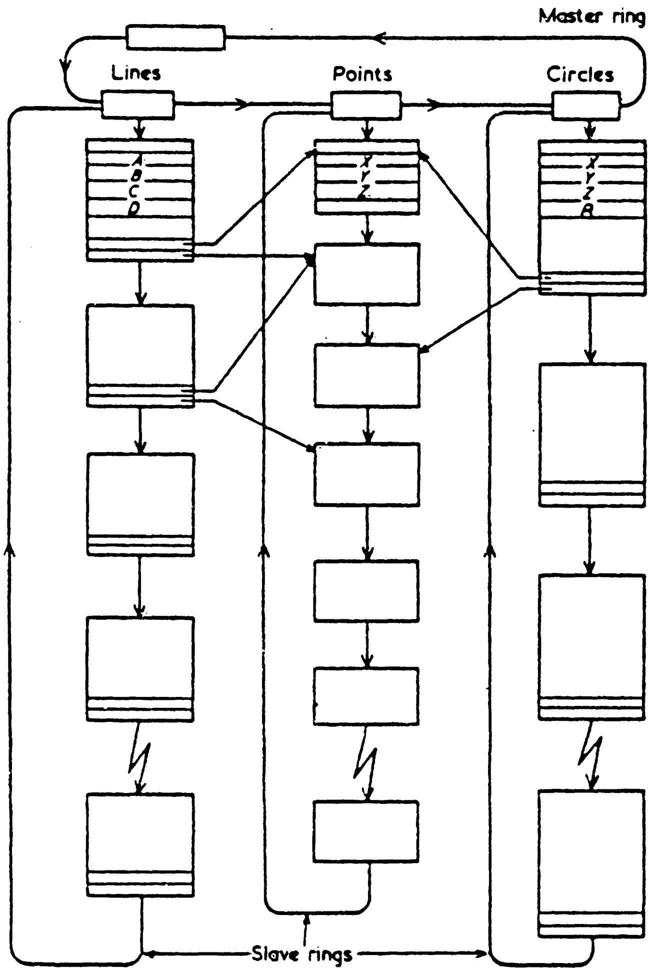


Figure 2.16 A Typical Data Structure for Geometric Entities

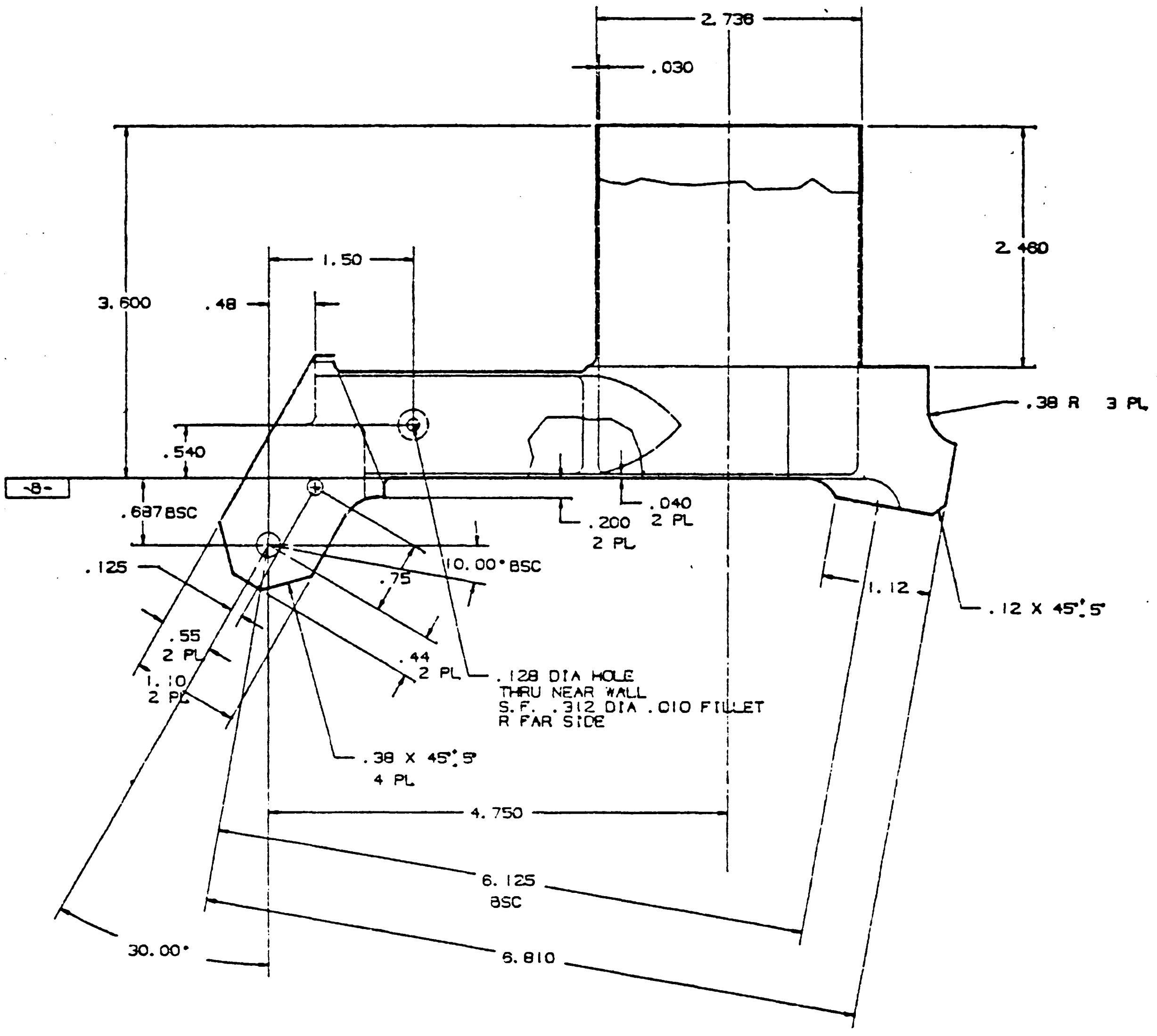


Figure 2.17 A Drawing from a CAD System

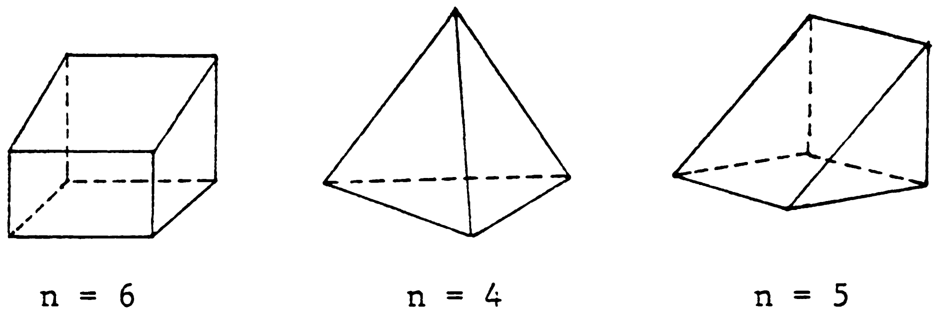
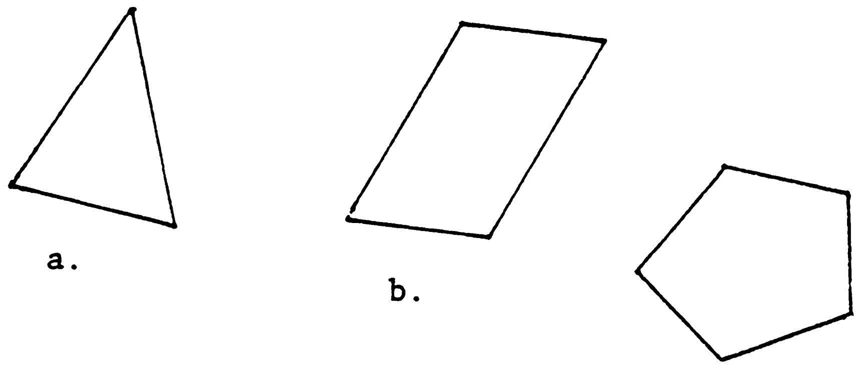
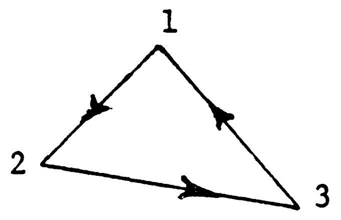


Figure 2.18 <u>Polyhedrons</u>



a. <u>Polygon</u>



$\{V_1, V_2, V_3\}$

$\{e_{12}, e_{23}, e_{31}\}$

Figure 2.19 <u>Polygons and Their Representation</u>

Another problem yet to be reckoned with in CAD systems is detecting whether the inside or outside of a polyhedron is being displayed. The side of a plane facing the interior of a part cannot be seen. However, the side of a plane can be identified by the normal of a face. Any point (x, y, z) on a face can be represented by the equation,

$$ax + by + cz + d = 0$$

where (a, b, c, and d) are parameters of the face. If P represents the face, then P can be characteriaced by its normal and a constant.

$$P = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

We can also map a vertex (from 3-D space to 4-D space) in order to obtain a homogenous representation. This can be written in vector form as

$$V = (x, y, z, w)$$

where w can be any number. The inside or outside then can be represented by

$$V \bullet P \leq 0 \quad \text{Or } V \bullet P > 0$$

2.3.3 Topology

Surface topology represents the basic relationships of the geometries of an object. For example in Figure 2.20, the topology of a tetrahedron is shown using a tree structure. The basic topological relationships will always remain the same for any tetrahedeon even though tetrahedrons can be of different size and may be oriented differently. The analogy of the tetrahedeon can be expanded to any polyhedron. Any polyhedron can be characterized by its:

1. geometry,
2. topology, and
3. auxiliary information.

Both the geometry and topology are necessary in order to manipulate and display the object. Some additional information may also be necessary to display the object or for other purposes (e.g. engineering analysis, manufacturing, etc.). Such additional information is typically stored in an auxiliary information buffer. A complete model for a polyhedron is shown in Figure 2.21.

2.3.4 Curves and Curved Surfaces

Thus far, only planar polygons have been discussed. The representation required for curved surfaces or curves is more complex. There are two ways to represent curves: as

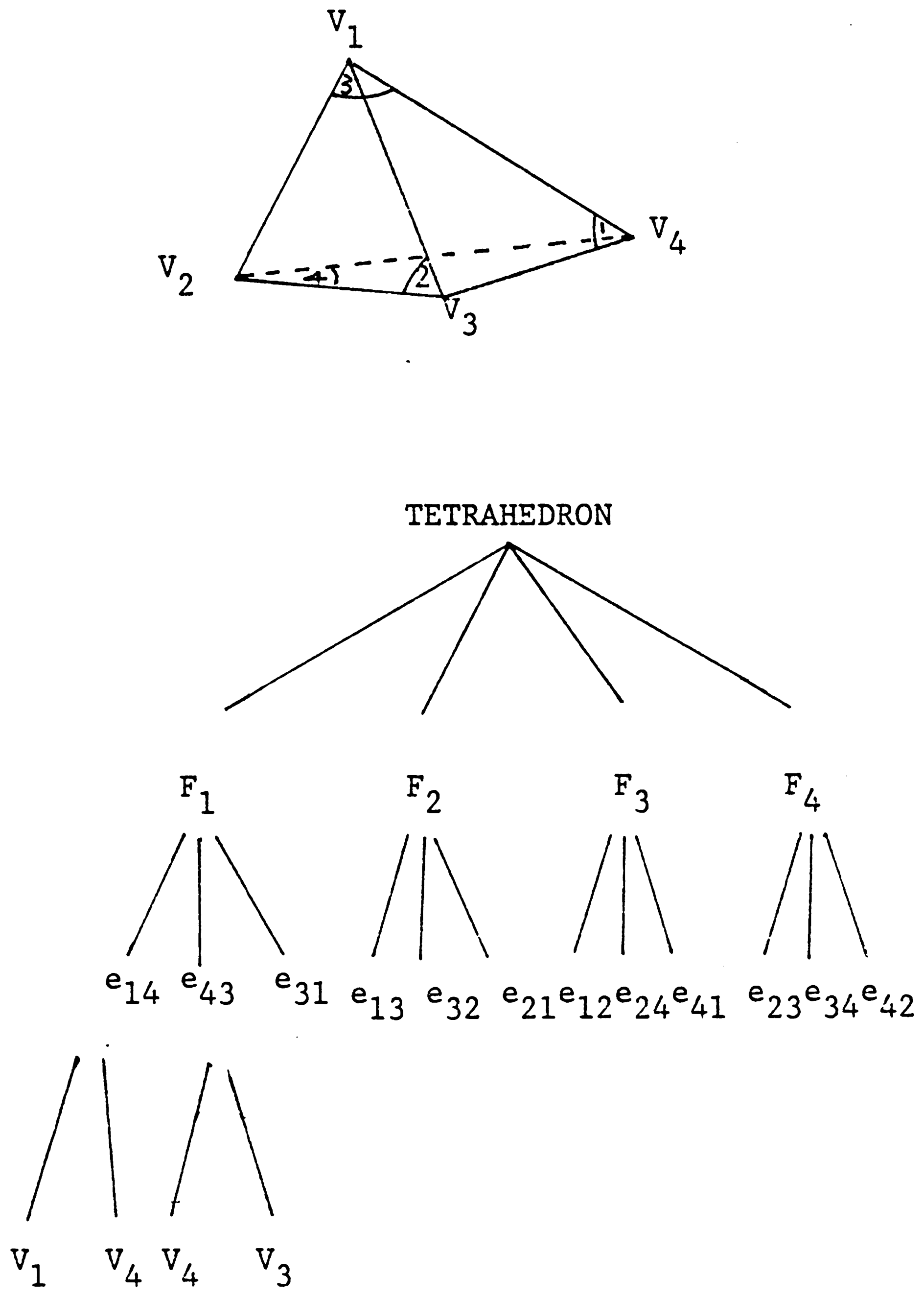
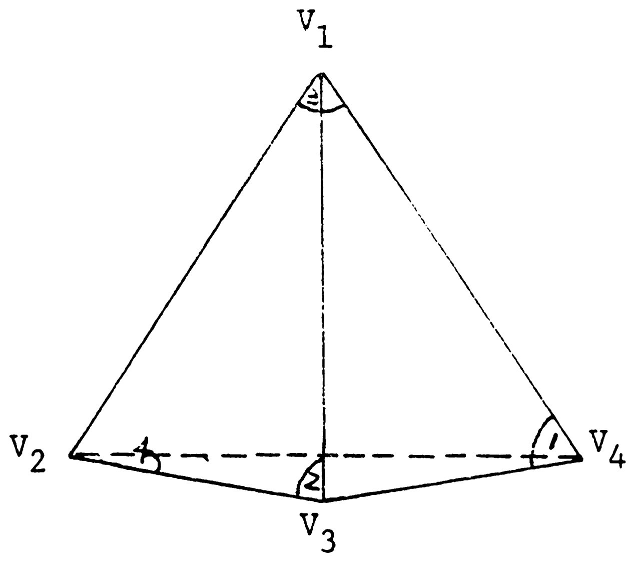

V_1
V_2
V_3
V_4
3
2
1
4
TETRAHEDRON
F_1 F_2 F_3 F_4
e_{14} e_{43} e_{31} e_{13} e_{32} e_{21} e_{12} e_{24} e_{41} e_{23} e_{34} e_{42}
V_1 V_4 V_4 V_3

Figure 2.20 <u>Topology of a Tetrahedron</u>



GEOMETRY		ax+by+cz+d>o outside	
V_1	(x_1,y_1,z_1)	F_1	(a_1,b_1,c_1,d_1)
V_2	(x_2,y_2,z_2)	F_2	(a_2,b_2,c_2,d_2)
V_3	(x_3,y_3,z_3)	F_3	(a_3,b_3,c_3,d_3)
V_4	(x_4,y_4,z_4)	F_4	(a_4,b_4,c_4,d_4)

Topology

Faces		Edges	
F_1	V_1,V_4,V_3	V_1V_2	V_1V_4
F_2	V_1,V_3,V_2	V_2V_3	
F_3	V_1,V_2,V_4	V_3V_4	
F_4	V_2,V_3,V_4	V_1V_3	
		V_2V_4	

AUXILIARY INFORMATION

COLOR
DIMENSIONS
ETC.

Figure 2.21 COMPLETE REPRESENTATION OF A TETRAHEDRON

functions of the variables x, y, and z, or as functions of another parameter (such as t). In the first case, points on a curve can be defined,

$$X = x, \qquad y = f(x), \qquad z = g(x).$$

This representation poses many computational difficulties for computer applications (e.g., an infinite slope may be required at some point on a curve). It is also difficult to plot a smooth curve using finite relationships.

The parametric cubic curve is one in which x, y, and z are represented as a third-order polynomial of some parameter t. Because slope can be replaced with that by tangent vectors, some computational difficulties can be reduced. In this representation, a curve is represented as:

$$\left.\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) &= a_y t^3 + b_y t^2 + C_y t + d_y \\ z(t) &= a_z t^3 + b_z t^2 + C_z t + d_z \end{aligned}\right\} \quad 0 \leqslant t \leqslant 1$$

or in vector form

$$P(t) = T \cdot C$$

where

$$T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} a_x & b_x & C_x & d_x \\ a_y & b_y & C_y & d_y \\ a_z & b_z & C_z & d_z \end{bmatrix}$$

Tangent vectors can be

$$\frac{dx}{dt} = 3\ a_x t^2 + 2b_x t + C_x$$

$$\frac{dy}{dt} = 3\ a_y t^2 + 2b_y t + C_y$$

$$\frac{dz}{dt} = 3a_z t^2 + 2b_z t + c_z$$

which will never infinite.

There are many methods to define parametric cubic curves. The most commonly used methods are Bezier [1972] and B-spline [Gordon 1974] methods. Both methods approximate a curve from n known points (a_x, b_x, c_x, ..., are unknown). We call those points control points P . In the Bezier method, the curve is defined as

$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

and $B_{i,n}$ (t) is called a blending function.

$$B_{i,n}(t) = C(n,i)\ t^i\ (1 - t)^{n-i}$$

and

$$C(n,i) = \frac{n!}{i!(n-i)!}$$

Figure 2.22 shows a curve drawn using the Bezier method. Since the endpoints are connected and the tangent vector of end point P_1 is the tangent of $P_1 P_2$, first order continuity of two Bezier curves can be obtained by making the end point of two curves coincide and the control points adjacent to the end point lie on the same line (Figure 2.23).

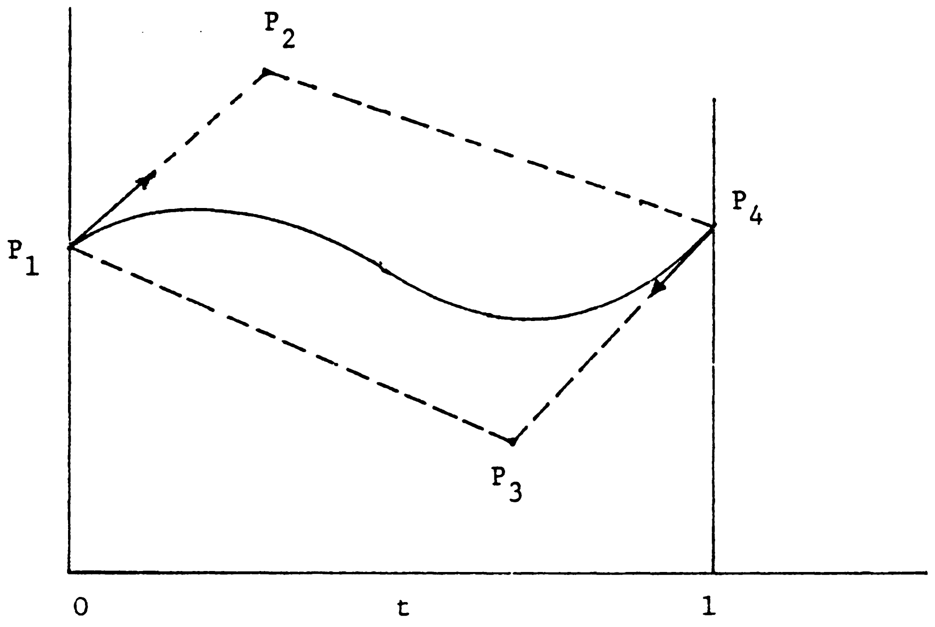
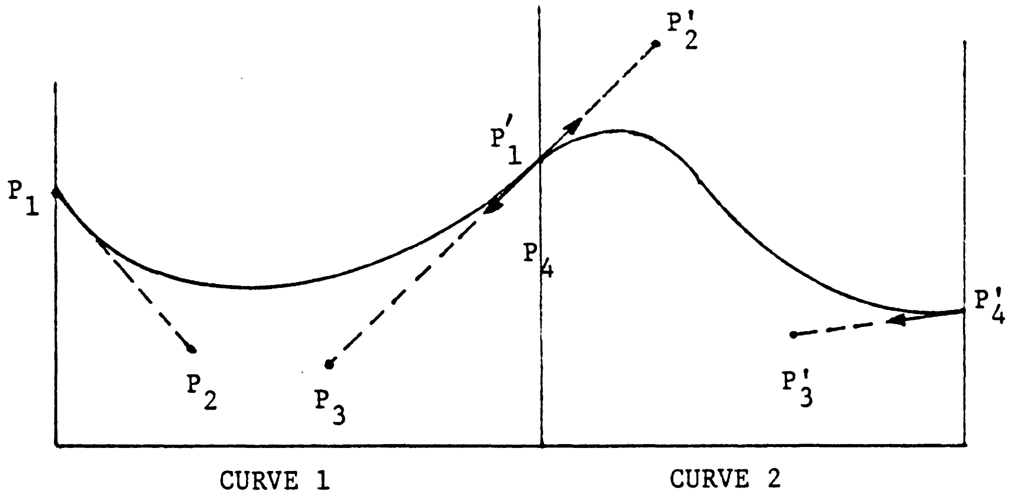
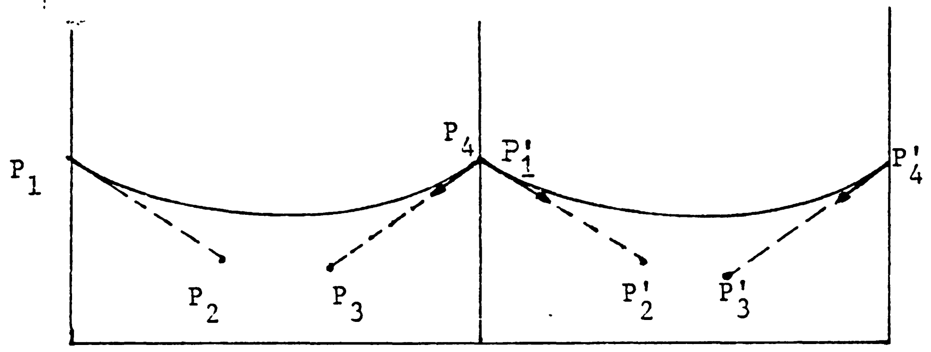


Figure 2.22 Bezier Curve and Its Four Control Points



FIRST ORDER CONTINUITY AT JOINT



ZERO ORDER CONTINUITY AT JOINT

Figure 2.23 Continuity at Joint

The B-spline method also uses a set of blending functions. However with the B-spline method, the location of the curve depends only on a few neighboring control points. A B-spline curve can be defined as

$$P(t) = \sum_{i=0}^{n} P_i N_{i,k}(t) \qquad 0 \leq t \leq n - k + 2$$

where: $N_{i,k}(t)$ is the blending function,

k is the parameter which controls the order of continuity,

and

$$N_{i,1}(t) = \begin{cases} 1 \text{ if } & u_i \leq t < u_{i+1} \\ 0 \text{ otherwise} \end{cases}$$

$$N_{i,k}(t) = \frac{(t-u_i)\ N_{i,k-1}(t)}{u_{i+k-1} - u_i} + \frac{(u_{i+k}-t)\ N_{i+1,k-1}(t)}{u_{i+k} - u_{i+1}}$$

where

$$\begin{aligned} u_i &= 0 \quad \text{if} && i < k \\ &= i - k + 1 && k \leq i \leq n \\ &= n - k + 2 && i > n \end{aligned}$$

and zero/zero = zero

Both Bezier and B-spline curves can be drawn in a seemingly natural way (i.e., the curve is variation-diminishing, axis independent and multi-valued). The B-spline however has local control which enables a user to define a shape without connecting many pieces of curve together. Figure 2.24 shows a comparison of two curves drawn by the Bezier and B-spline methods with the same set of control points.

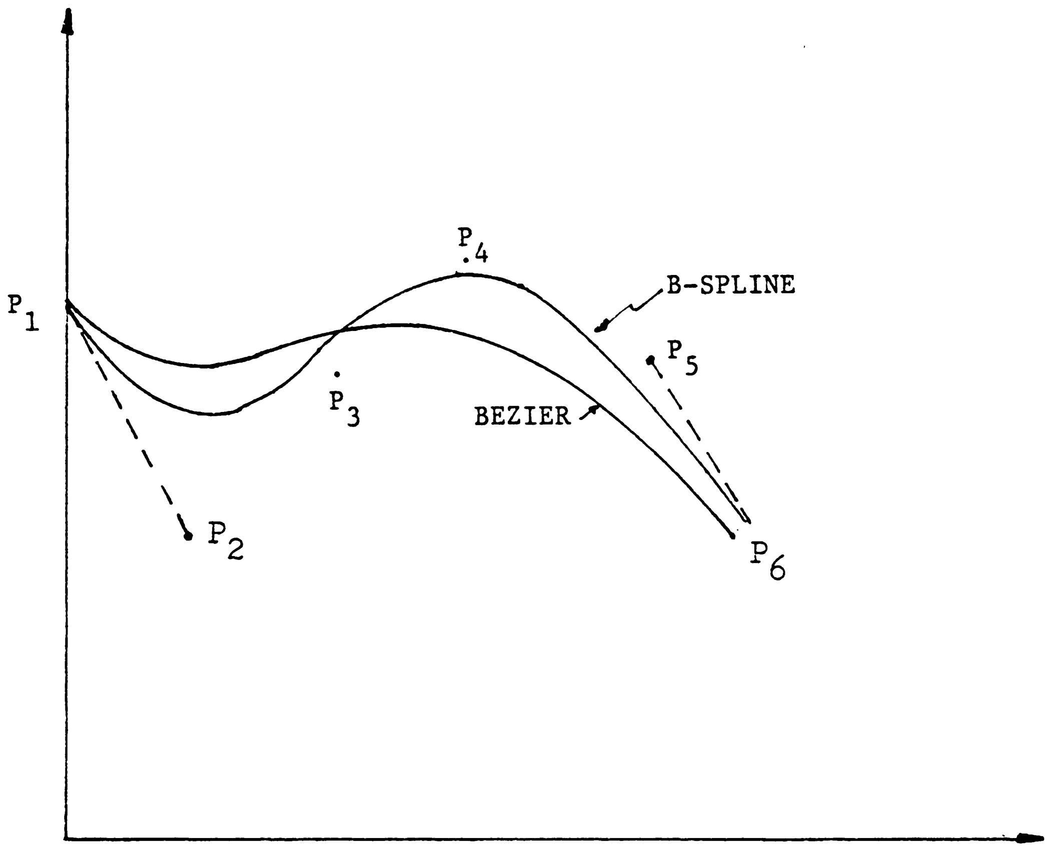


Figure 2.24 Comparison of Bezier and B-Spline Curves

Both the Bezier and B-spline methods can be extended to define curved surfaces. For surfaces, two parameters (t, s) are required. Using a Cartesian product method, a Bezier surface can be written as:

$$P(t,s) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{i,j} B_{i,n}(t) B_{j,m}(S)$$

$B_{i,n}$ (t) and $B_{j,m}$ (s) are similar to blending functions. In a Bezier surface, a total of (n+1) x (m+1) control points, arranged in a mesh are necessary.

A B-spline surface also can be written as

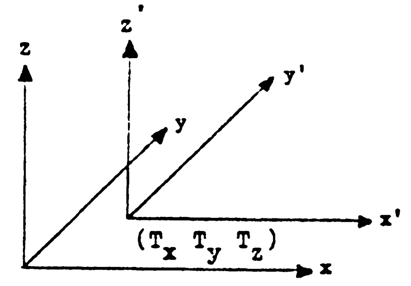
$$P(t,S) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{i,j} N_{i,j}(t) N_{j,1}(S)$$

requiring the same constraints and allowing the same flexibility as the Bezier method.

2.3.5 Geometric Trnasformation

No matter how well an object is modeled, we still must display it in a manner which allows us to easily perceive its detail. Geometric transformations are an important tool for generating images and manipulating primitives. They can be used to define the location of an object relative to other objects. Generally, any image model can be viewed from any point and direction as a 3-D image or translated into a 2-D perspective view.

Basic transformations include translation, scaling, rotation, and mirror imaging. These transformations are repre-

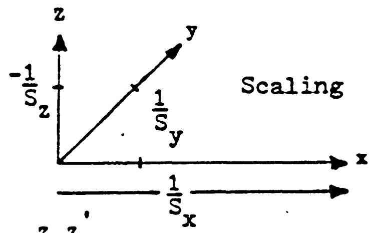


TRANSLATION

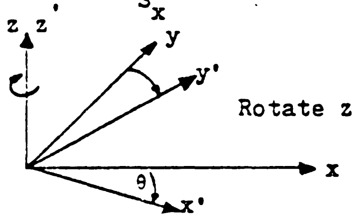
$$T_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -T_x & -T_y & -T_z & 1 \end{bmatrix}$$

$$V' = V\,T_t$$

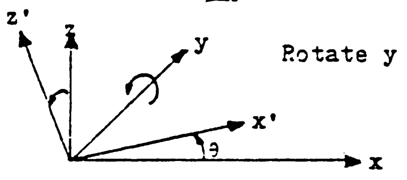
$$V = [x\ y\ z\ 1]$$



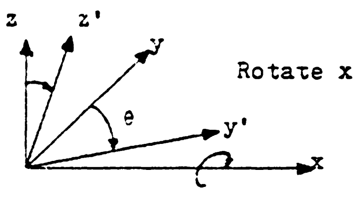
$$T_s = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



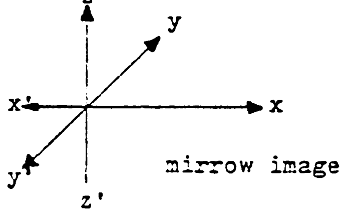
$$T_{r_z} = \begin{bmatrix} \cos\theta & +\sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$T_{r_y} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$T_{r_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$T_m = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2.25 Transformations

sented mathematically in matrix form and pictorially in Figure 2.25. Any point, (V), in the originial coordinate system (x,y,z) can be transformed to some other point, V'; in a new system x',y',z' using the relationship

$$V' = V \cdot T.$$

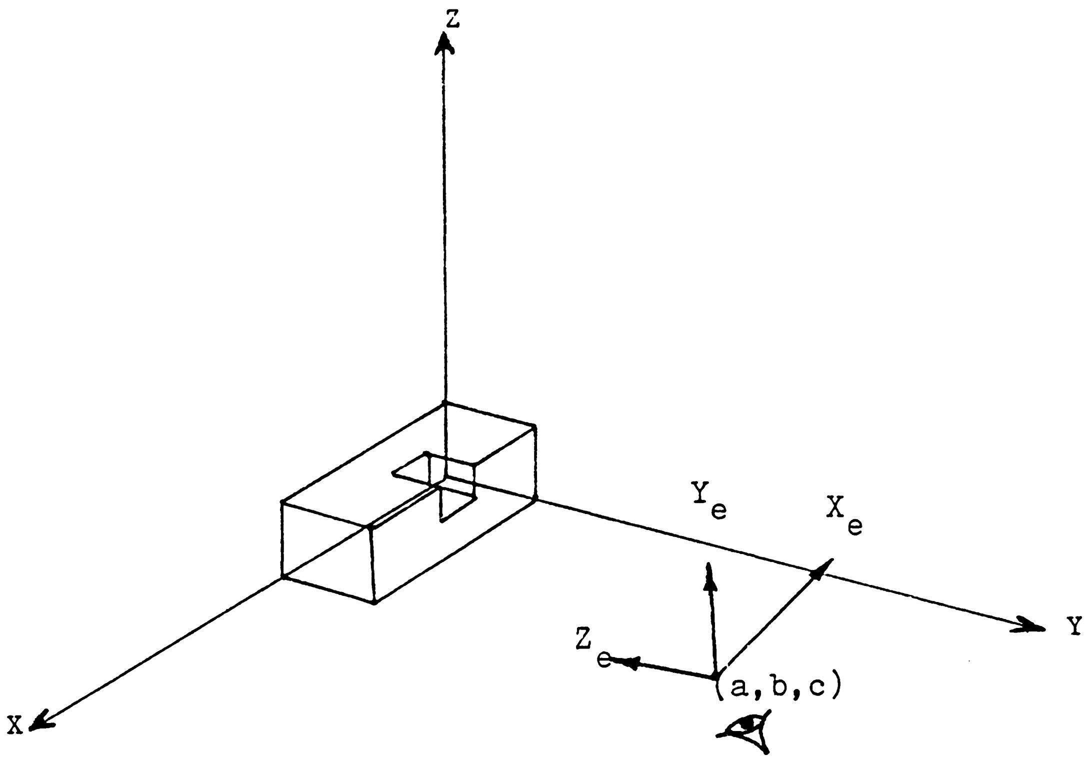
In viewing an object, we need to define two sets of coordindates: 1) the object's coordinates (the originial coordinate definition; and 2) the Viewer's coordinate position. In Figure 2.26A, the object is viewed at a distance Figure 2.26B shows the object viewed from a standard eye location. The object in Figure 2.26C has been transformed using the following sequence,

1. Translate to location (a,b,c)
2. Rotate about x(-90 degrees), and
3. Mirror image on X.

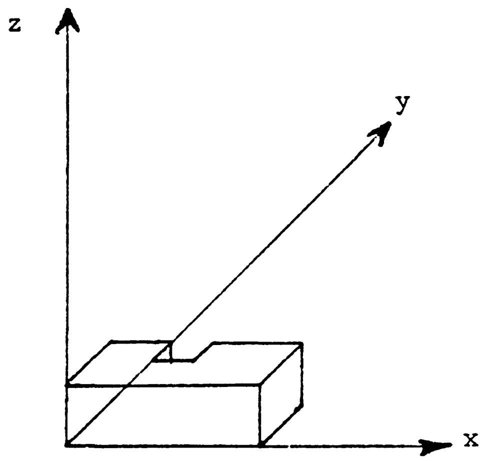
The concentrated transformation matrix can be written as:

let $\theta = -90$ degrees

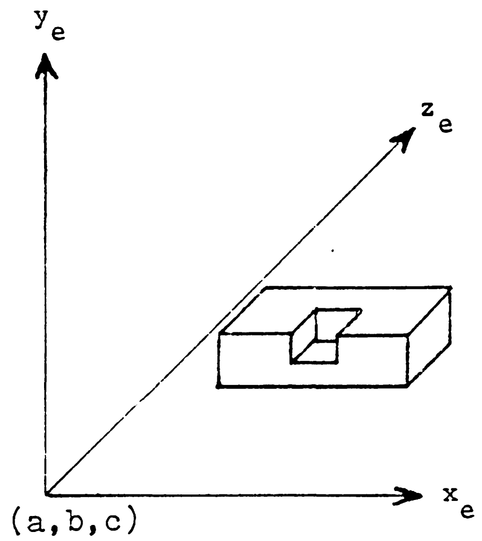
$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a & -b & -c & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



a.



b. ORIGINAL COORDINATE SYSTEM
(RIGHT-HAND CONVENTION)



c. EYE COORDINATE SYSTEM
(LEFT-HAND CONVENTION)

Figure 2.26 TRANSFORM THE VIEW

2.3.6 Perspective Transformation

The next graphical issue to be discussed concerns the development of a methodology to display a 3-D object on a 2-D CRT or plotter. In engineering drawing, there are either multi-view drawings or perspective drawings. The same basic approaches are applied to computer generated drawings. For a multiview (say 3 view) display, each view is a simple 2-D orthographic drawing. The front, top, and side views can be generated by simply discarding y, z, and x coordinates respectively. To construct a perspective drawing, a geometric transformation is required. A perspective drawing can be generated by projecting each point of an object onto a plane (called a display screen (Figure 2.27) which is characterized by d; the distance from the eye. The X_s and Y_s axis of the display screen are parallel to the X_e and Y_e of the eye coordinate system. The transformed projection is shown in Figure 2.28. Point P (x,y,z) in the eye coordinate system can be projected onto the display screen as (x_p, y_p). By using the relationships of two similar triangles, x_p and y_p can be found.

$$\frac{x_p}{d} = \frac{x}{z}, \qquad \frac{y_p}{d} = \frac{y}{z}$$

$$x_p = \frac{x}{z/d}, \qquad y_p = \frac{y}{z/d}$$

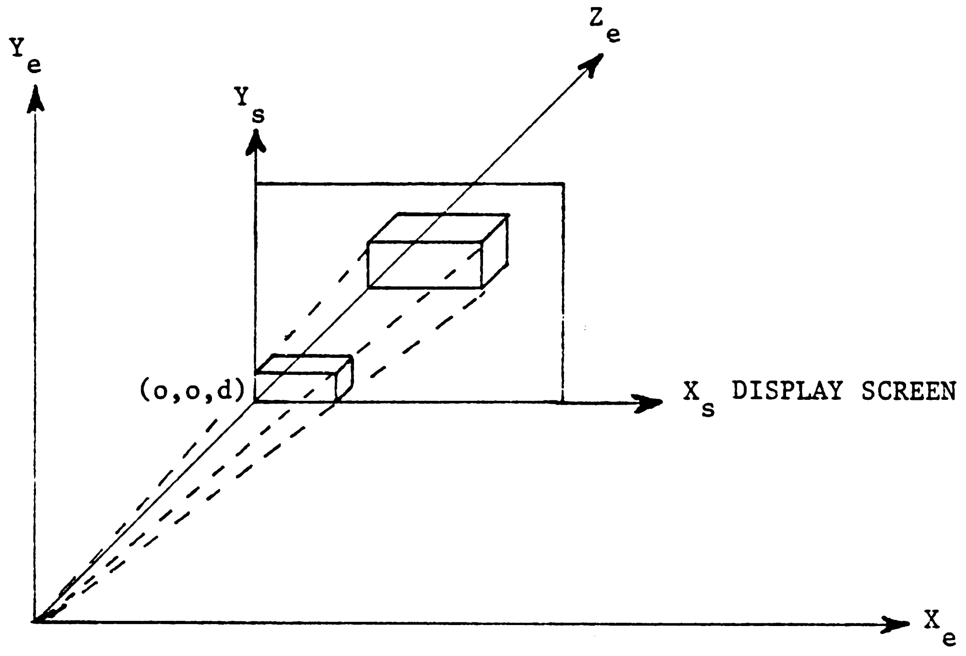


Figure 2.27 <u>The Perspective Projection of an Object on a Display Screen</u>

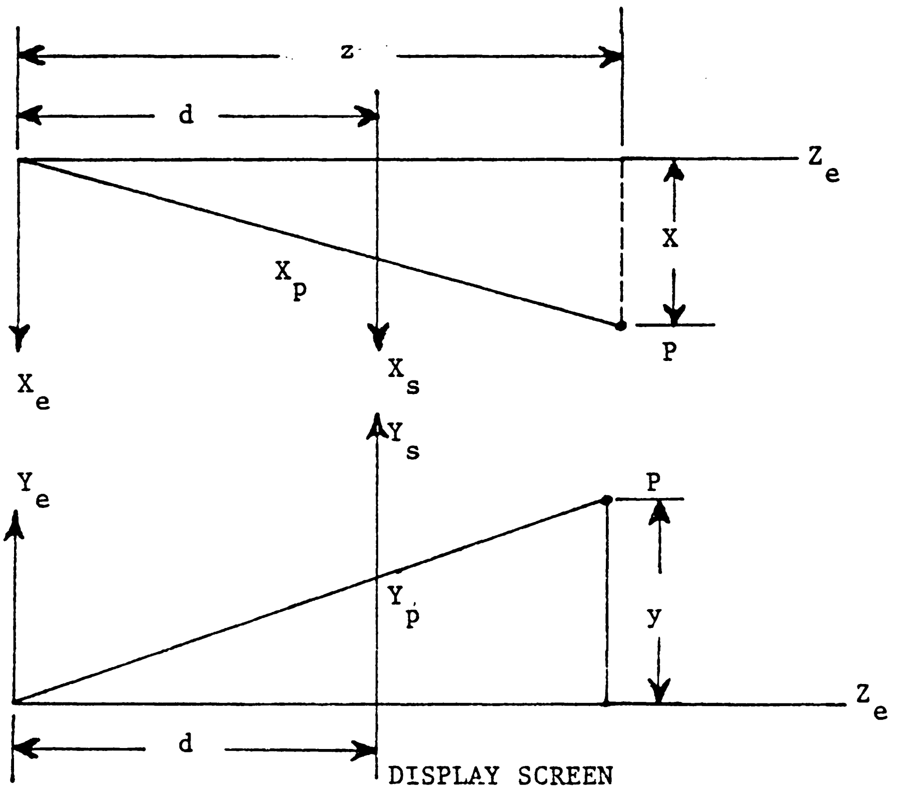


Figure 2.28 <u>Perspective Projection of a Point</u>

All points in the eye coordinate system except those on the eye plane (Z=0) can be transformed. Clipping, a process to discard views outside the screen, is necessary to eliminate invisible lines (eliminate line segments outside the drawing window and eliminate lines hidden by a surface).

2.3.7 Data Structure

Each geometric model of an object in a CAD system is represented by a set of data. The data consists of numerical values, names, codes, and symbols. This data is stored in computer memory in an organized manner. This organization represents the relationship of each data element to other elements. Included in these relationships is the topological relationship of the surface geometry. There are also other relationships such as names and their associated geometries. A more complete survey on the different types of data structures for computer graphics can be found in two technical papers by Williams [1971] and Grays [1967]. In this section, only a general data structure of CAD will be described.

As dicussed in the previous section, an object can be represented by its geometry, topology and ancillary information. This information can be stored in a list structure. (This list structure will be discussed more fully in Chapter

5). By using data pointers, each element geometry (i.e. vertex, edges, faces, etc.) can be connected. A simple data structure is illustrated in Figure 2.29. Although data structures for operational CAD systems are more complicated [Baer 1979, Goldstein 1981], The basic concept is similar. In the figure, a tetrahedron is modeled and stored in a specific data-base. The object (a tetrahedron) has four faces which are pictorially and relationally linked via a link list. A pointer in the "face list" links a face with its edges. Face #1 points to location #1 in the "edge list". Since the faces are comprised of polygons the number of edges for each face must be stored. The first item in the edge list; therefore, represents the number of edges. In the figure, edge 1 is represented by the data: "3,3,4,5", where the first 3 indicate there are three edges - 3, 4, and 5. Each edge has two vertices (edge three has V_1 and V_3). Finally, a vertex may store all the coordinates of vertices.

When objects are represented by polyhedrons with planer polygons, the data structure in Figure 2.29 is sufficient. However, not all components in mechanical design can be modelled using such polyhedrons. Therefore, additional data elements must be stored. Holes are the most common shape in mechanical design and cannot be stored as a simple polyhedron. A hole can be represented as two circles and two vir-

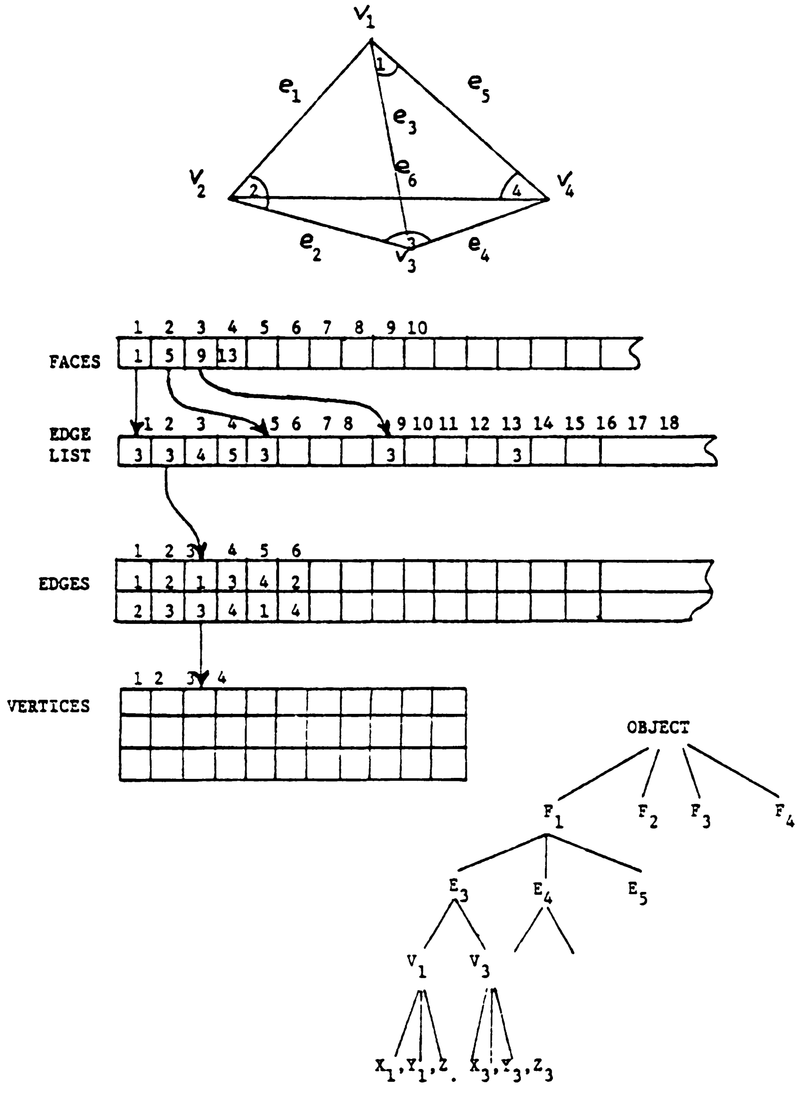


Figure 2.29 A Simple Data Structure
(The Link Within Each List Is Not Shown)

tual edges. The implication of virutal edges implies that they are the projection of a curved surface onto a 2-D display screen. For holes, circles and other curves and curved surfaces, a code is necessary to distinguish them from polyhedron surfaces. In order to avoid redundant computation during some image manipulation, it is sometimes desirable to have "face" equations stored. An additional pointer is therefore necessary to link the parameter list. An example of the data structure for a research/commercial available system is shown in Figure 2.30 [Baer 1979]. BUILD is a system written by Braid at Cambridge University. A commercial system Romulus, marketed by Evans and Sutherland is also based on BUILD.

For manufacturing purposes, technological information, such as dimensions, tolerances, and geometric tolerances are essential. They also must be included in constructing the CAD data structure. Most current systems explicitly (or more appropriately externally) define such information [Gopin 1979], in a manner similar to the way a draftsman would. Dimensioning and tolerancing information is not implicit defined within the model. For explicit dimensioning, another list is added to store the dimensioning information. However, some systems such as PADL [Voelcker 1977] use symbolic dimensioning. In such systems, dimensions are measured directly from the model.

However, geometric tolerances are not available directly from the model, i.e., the model always shows perfect geometric tolerances. The data structure must be expanded to include space for storing tolerance information.

Since most CAD systems use different data structures, it is not possible to move data directly from one system to another. With the increasing need to develop a common data exchange, a joint effort was undertaken by industry and the National Bureau of Standards with the support of U.S. Air Force. The project resulted in a data exchange standard -- IGES (Initial Graphics Exchange Specification] [Nagel 1980). Several existing commercially available CAD systems already have preprosser and postprocessors to read and create IGES files. A survey of some 3-D modeling systems is shown in Table 2.2 [Baer 1979].

2.3.8 Geometric Modeling for Process Planning

In the previous section, the fundamentals of computer-aided design were discussed. In this section, a discussion of 3-D representaticn schemes and their application to process planning is presented.

There are seven different types of graphic representation schemes:

1. wire frame,
2. primitive instancing,

Table 2.4 Some 3-D Modeling Systems (taken from Baer 1979)

Name	AD2000	BDS/GLIDE	BUILD-1	COMPAC	EUCLID(F)	EUCLID(S)	GEOMED	GEOMAP	'HOSAKA'	PADL	TIPS-1
Developer	Hanratty	Eastman et al	Braid	Spur et al	Brun et al	Engeli	Baumgart	Hosaka & Kimura	Hosaka & Kimura	Voelcker et al	Okino et al
Institution	Manufacturing and Consulting Services Inc.	Carnegie- Mellon University	Cambridge University	Technical University of Berlin	LIMSI	Institution	Stanford University	University of Tokyo	University of Tokyo	University of Rochester	Hokkaido University
Country	USA	USA	UK	Germany	France	Switzerland	USA	Japan	Japan	USA	Japan
Host/ Implementation Language	FORTRAN	BLISS	FORTRAN SAL	FORTRAN	FORTRAN FOCAL	SYMBAL	ASSEMBLY	FORTRAN	GIL	FORTRAN	FORTRAN
Machine	Many different Machines	DEC PDP-10	TITAN PDP-7	DEC PDP-10	UNIVAC- 1110	CDC6500	DEC PDP-10	TOSBAC 5600 (GE635)	TOSBAC 5600 (GE635)	PDP-11/45	FACOM machines
Interactive/ Batch	interactive	interactive	interactive	batch	either	batch	interactive	batch	batch	interactive	batch
Purpose/area	commercial: Engin. & NC	research: database for Arch. & Engin.	research: Engin. & NC	commercial: Engin. & NC	commercial: Engin. & Arch.	commercial: Engin. & NC	research: visual shape recognition	research: Engin. & NC	research: Engin. & NC	commercial: Engin. & NC	commercial: Engin. & NC
Shaded Graphics	-	-	Yes	Yes	-	-	-	-	Yes	-	-
Curved Surfaces	splines, toros and others	approximate cylinders & cones	cylinder	cylinder & others	circle & curve segments	(version: Bezier patches)	-	approx. cylinders cones	-	cylinders	cylinder sphere bicubic patch
Shape Operators	intersection (of faces)	union intersection difference	union intersection difference addition	union difference	-	union intersection difference addition	union intersection difference	union intersection difference	union intersection difference	union intersection difference	union difference
Solid definition Method	generating lines and surfaces	Euler operations	primitive solids	generating lines, primitives	face list	face list primitives solids	Euler operations	primitives	intersecting planes	primitive solids	primitive solids

3. spatial occupancy enumeration,
4. cell decomposition,
5. constructive sold geometry (CSG),
6. boundary representation, and
7. sweeping.

Of these, wire frame systems are the most commonly used representation. Most commerically available computer-aided drafting systems are wire frame representations. In such systems, objects are represented by points, lines, curves, circles, etc. (Figure 2.31c). A wire frame is not considered a solid model because it does not provide a complete or unambiguous representation of a part.

Pure primitive instancing is one kind of solid representation (Figure 2.31d). It can be used to represent a family of objects. One application of pure primitive instancing is GT coding [Reguicha 1980].

Spatial Occupancy Enumeration (SOE) records all spatial cells which are occupied by the object (Figure 2.31e). SOE is equivalent to storing the physcial object in sections. In order to describe the object accurately, very small cells must be used. The massive memory required for even small objects makes this representation virutally useless in engineering design.

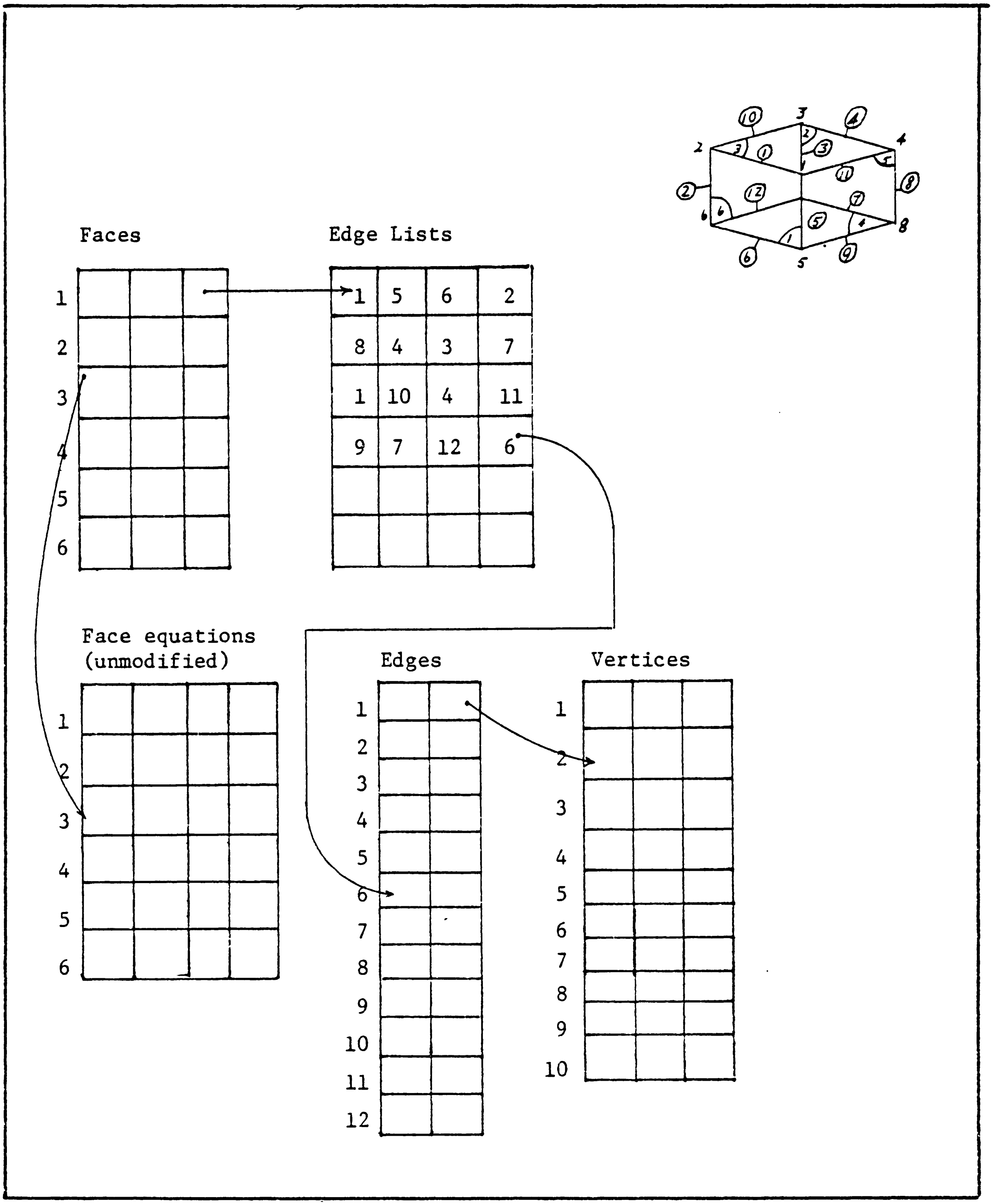
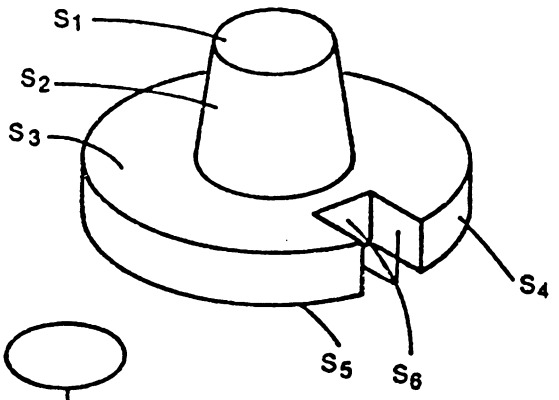
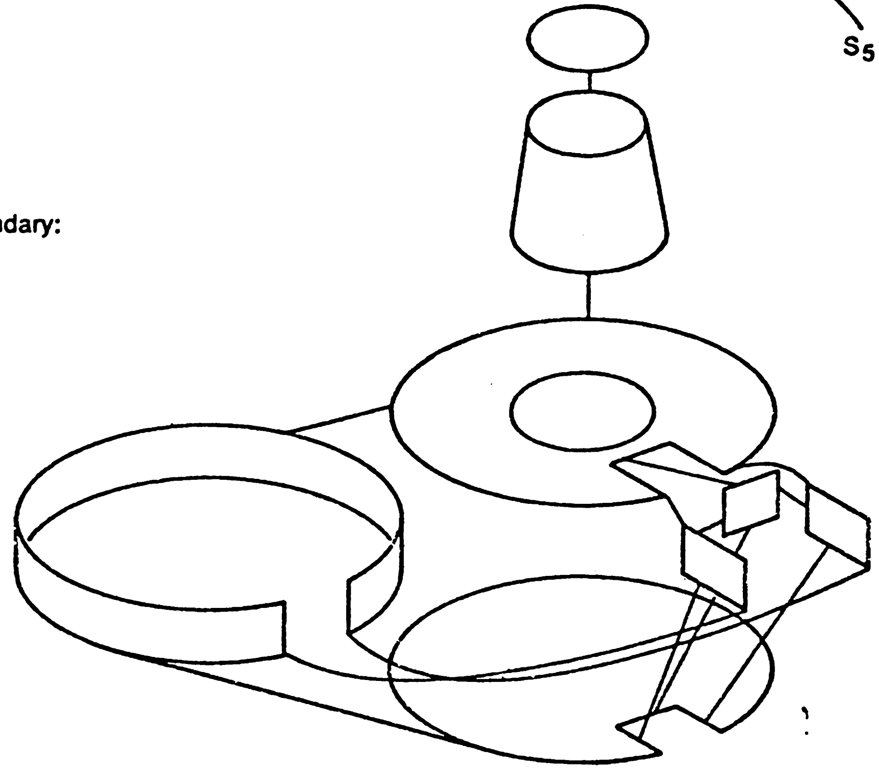


Figure 2.30 A Simple Data Structure



a. CSG: (Δ · dif · Δ) · un · (⬭ · dif · ▭)

b. Boundary:



c. Wire Frame

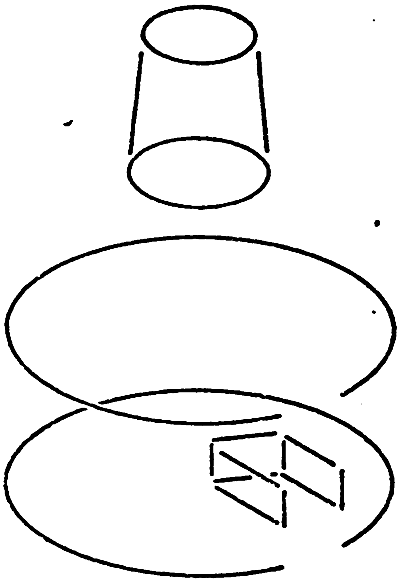
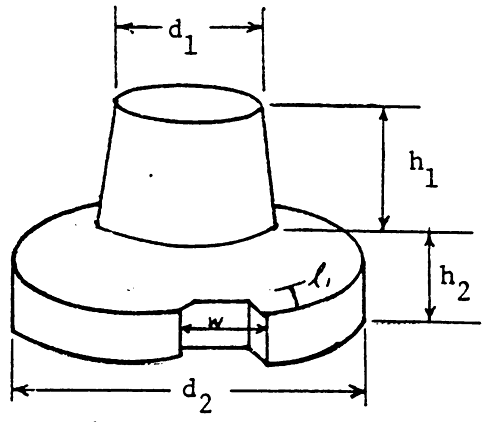
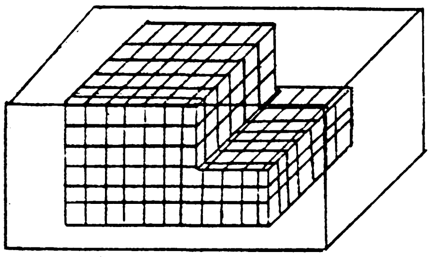


Figure 2.31 <u>3-D Representation Schemes</u>



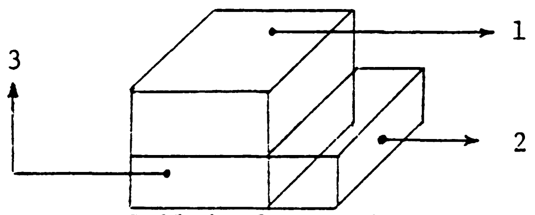
Family A(d_1,h_1,d_2,h_2,ℓ,w)

d. Pure Primitive Instancing



List of Cells Occupied

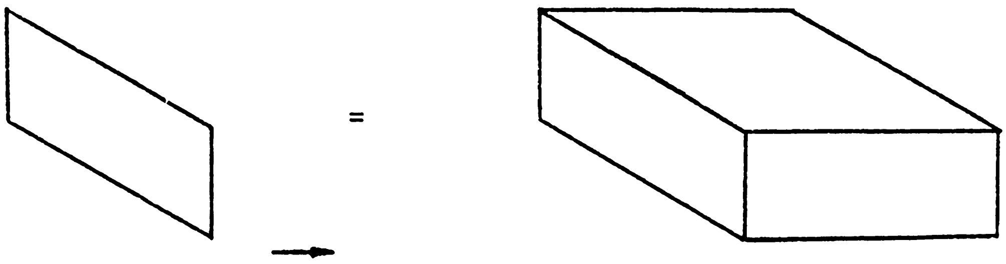
e. Spatial Occupancy Enumeration



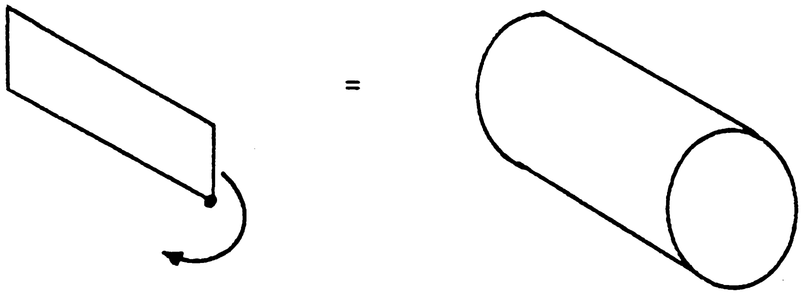
Cell 1, 2, and 3

f. Cell Decomposition

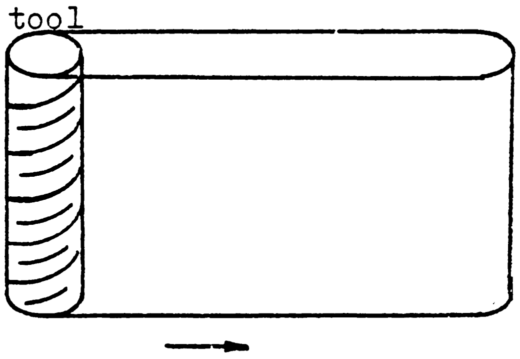
Figure 2.31 (continued)



Translation Sweep



Rotation Sweep



Translation Sweep of a tool

g. Sweeping

Figure 2.31 (continue)

Cell decomposition is a general class of spatial occupancy enumeration. The representation is difficult to create. A solid is decomposed into simple solid cells with no holes, and whose interiors are pairwise disjoint (Figure 2.31F). A solid is the result of "glueing" component cells that satisfy certain "boundary-matching" conditions.

Constructive Solid Geometry (CSG) is a superior system for creating 3-D models. Using primitive shapes (Figure 2.32) as building blocks, CSG employs Boolean set operators to construct an object (Figure 2.31b). Because CSG cannot directly link to a drawing procedure, a transform of CSG to another representation is required. Since the display is generated by another representation (usually a boundary representation), and the reverse process of translating back to CSG is nontrival, direct manipulation of the graphics on the screen is difficult. The input to certain process planning systems (e.g. AUTAP) has been considered using a type of CSG model.

Boundary representations are also used to identify an object. In these systems, objects are represented by their bounding faces. Faces are further broken down and represented by edges and vertices. Boundary models are very difficult to create, but can provide easy graphic interaction.

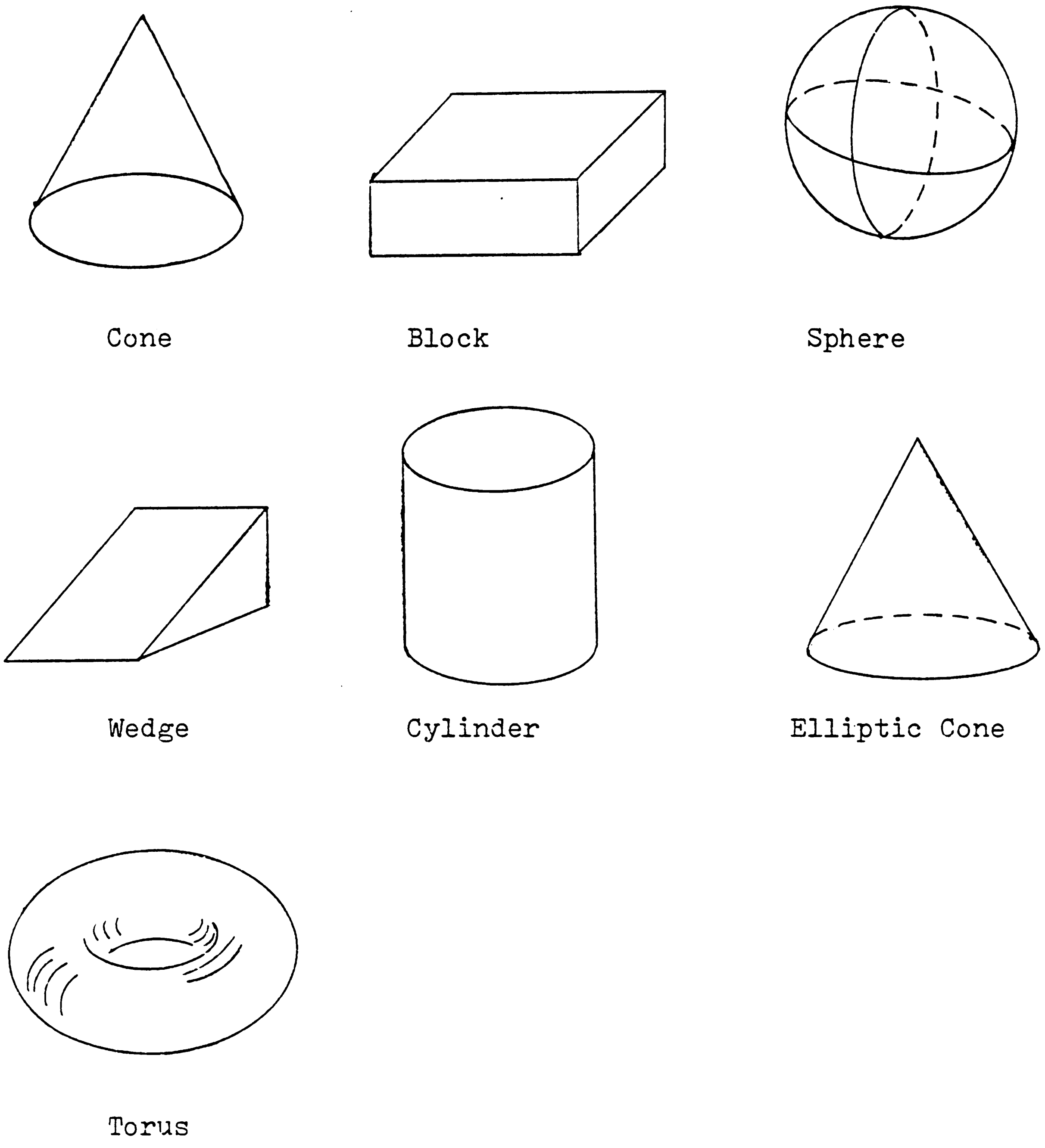


Figure 2.32 An Illustration of Some Primitives

Sweeping is another powerful modeling tool for certain types of geometry. There are two types of sweeping: translation and rotation (Figure 2.31a). Translation sweeping of a rectangle produces a box. Rotational sweeping of the same rectangle is a cylinder. Rotation sweeping can be used to create turned parts. The best use of translation sweeping is in cutter path simulation. Simulation can create intermediate surfaces, which can then be used to determine fixture clamping points and robot gripping points.

With the above taxonomy of graphical representation some general problems of current part representation systems can be identified. These problems include:

1. the lack of embedded tolerancing information
2. the inability to represent special manufacturing features, i.e. drill angles of a hole,
3. poor graphical interaction,
4. a lack of data interchange between representations, and
5. no method currently exists to identify differences in two given models automatically (especially when they have different orientations in space).

Since there is no easy way to identify surfaces requiring machining using any of the previously mentioned geometric representations, the best immediate solution is to use a human-computer interactive system to identify the surface. Wire frame, boundary representation and sweeping are candidate graphic systems for interactive systems. Since sweeping can only model a subset of objects in the object domain, it cannot be used independently. Most currently available commercial CAD systems are based on a wire frame internal representation.

In order to use a CAD system as the front end for computer-aided process planning, the following capabilities are required:

1. lines and faces must be located and identified interatively,
2. easy dimension retrieval,
3. tolerancing information storage and retrieval,
4. built-in special tags, i.e. drill angle for hole, and
5. capability of displaying multiple objects.

The front end interface should be designed so that it is independent of any representation or specific system. Therefore, any system can be modified to support the previously mentioned five requirements in a single work, but alphabetics must be stored in an array.

2.4 GROUP TECHNOLOGY CODING

Both drafting and geometric modeling are detailed representations of an engineering design. They provide detailed information concerning the component to be made, and are essential in conveying the design for manufacturing. However, like with many decision making processes, too much information may make the decision more difficult. For example when one reads a magazine or journal, they seldom begin by reading each sentence on successive pages. Instead they peruse the Table of Contents first to locate needed technical information. By doing so, candidate articles can be located more quickly. However the title may not convey all of the necessary information. Therefore the reader would typically scan the abstract. The abstract is a summary which represents the article without great detail. Reading the abstract of an article normally provides us with the insight to continue the reading. For a long while, parts lists corresponding to a Table of Contents have been around, and although it is impractical to write abstracts for CAD model, a similar concept can be applied using coding.

Group technology (GT) as described in Chapter 1 is an appropriate tool for this purpose. Coding, a GT technique, can be used to model a component without all of the detail. When constructing a coding system for a component's representation, there are several factors to be considered. They include:

1. the population of components; i.e. rotational, prismatic, tub, sheet metal, etc.,
2. the detail the code should represent,
3. the code structure; chain, hierarchical or hybrid, and
4. digital repesentation, i.e. binary, octal, decimal, alphanumeric, hexadecimal, etc.

The population of components contributes to the variety of shapes. For example, the population in the United States includes virtually all races that exist on earth. In a sense, it is necessary to distinguish race, hair color, eye color, etc. However, in a nation like China or Japan, it is not worthwhile to record skin color, hair color, etc. because these items are virtually invariant. In component coding, it is also true that only those features which vary need to be included in the code. When designing or using a coding scheme, two properties must hold true: The code must be: 1) unambiguous, and 2) complete.

We can define coding as a function H, which maps components from a population space P into a coded space C (Figure 2.33). An unambiguous code can be defined as:

$$i \in P \ \exists \text{ only one } j \in C \Rightarrow j = H(i)$$

completeness can be defined as:

$$\forall i \in P \quad \exists\ j \in C \Rightarrow j = H(i)$$

The above two properties suggest that each component in a population has its own unique code. However, normally it is desireable to have several components in the population share one code.

The code need be concise. If a 100 digit code and a 10 digit code can both represent components in P, completely and unambiguously, then the 10 digit code is more desireable. However, when more detail is required a longer code is normally necessary. For example, the basic Opitz code (shown in Figure 1.19) uses five digits to describe the shape. Five digits can represent 10^5 combinations. With this set, it is not possible to show a large amount of detail of a component. Some codes are significantly longer, eg., the KK-3 of Japan [Japan 1980] has 21 digits, contain multiple digits for single features; e.g., MICLASS of TNO [Houtzeel 1979] has 12 a digit code. For some computer-aided process planning systems (e.g., APPAS [Wysk 1977]), a detailed surface code is used instead of a code for the entire component. The decision on how much detail the code should represent depends solely on the application. The selection of a code structure again depends on the application. Table

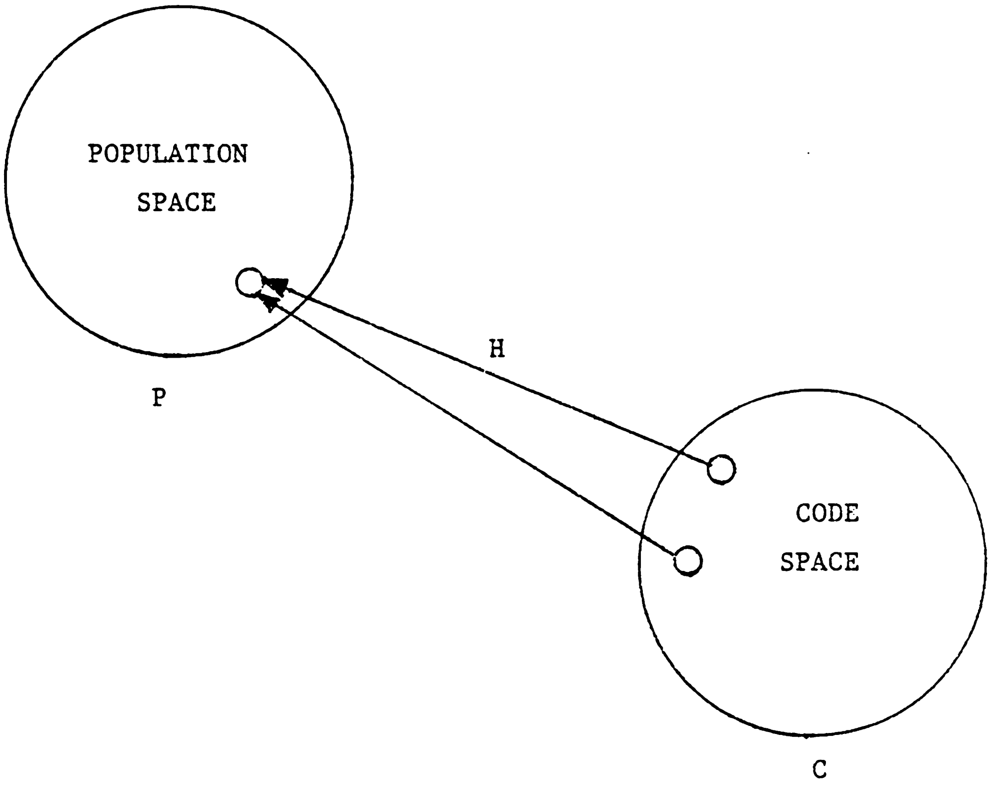


Figure 2.33 <u>MAPPING FROM POPULATION SPACE TO CODE SPACE</u>

1.2 of Chapter one provides a comprehensive selection criterion.

The last consideration in coding system construction is the code digits. Several positional alternatives can be selected (from binary to alphanumeric). However this selection yields different percision for the different schemes. For example, a N digit code with different sytems yields following combinations of code.

Binary	2	(0,1)
Octal	8	(0,1,...,8)
Decimal	10	(0,1,...,9)
Hexdecimal	16	(0,1,...,9, A,...,F)
Alphanumeric	(26 + 10)	(0,1,...,9, A,...,Z)

Although alphanumeric systems are the most compact (the same amount of information can be represented by fewer digits), the difficulty of handling both numerical and alphabetical characters make alphanumerics less attractive. In many computer languages, a numeric can be stored in a single word, but alphabetics must be stored in an array.

2.4.1 Code Contents

When using a code to represent an engineering design, it is important to represent the basic features of the design. As a human can be represented by his/her height, weight, color, hair, eye color, and sex, an engineering component can be represented by its basic shape, secondary features, size, tolerance, critical dimensions, material, etc. For process planning, it is desireable to have codes which can distinguish unique production families (as wasdiscussed in Chapter 1).

Fortunately most components of similar shape can be produced by the same set of processes. However the opposite is not true as frequently. Shape elements normally dictate the manufacturing process. Secondary features such as auxiliary holes, gear teeth, threads, chamfers, etc. are also important and can dictate a different set of process plans. From only the shape, there is no way of telling how large or small a component is. The production methods for a 2 cubic centimeter model airplane engine block is definitely different than that for the engine block of a 6 liter V-8 engine. Although similar processes may be applied to both engine blocks, the machines, tools, and material handling methods will probably be very different. Size, tolerance and surface finish can affect required processes. The price of a precision component (as was shown in Figure 1.3) increases as the tolerance is tightened. This is usually the result of a precision component requiring several more processes than a component of standard tolerance specification. For example, a milled workpiece will not require an addition finishing process if the specified surface finish is 125 microinch or above on a flat surface. However if the surface finish requirement is specified as 4 microinches, a careful finishing cut on the last milling pass followed by grinding, polishing and lapping is necessary.

The workpiece material is also a factor that must be considered. Tools for machining aluminum are not appropriate for machining alloy steel. Feeds and speeds used for machining also depend on the material.

Since a coding system transforms the above properties and requirements into a code, this information should somehow be tied into a process planning system. Some of the systems which have been successfully implemented (process planning systems) will be illustrated in the following section.

2.4.2 Opitz System

The Opitz coding system [Opitz 1970] is probably the best known coding system. It was developed by Professor H. Opitz of the Aachen Tech University in West Germany. The Opitz code uses a mixed code structure. However except for the first digit, it more closely resembles a chain structure (Figure 1.19).

The Opitz code consists of a geometric code and a supplementary code. The geometric code can represent parts of the following variety: rotational, flat, long and cubic. A dimension ratio is further used in classifying the geometry, i.e., length/diameter ratio is used to classify rotational components, length/width and length/height ratios are used to classify nonrotational components. The Opitz geometric code uses five decimal digits, each representing component class, basic shape, rotational surface machining, plane surface machining, auxiliary holes, gear teeth, and forming. Primary, secondary and auxiliary shapes can be represented using the five geometric digits.

A supplemental code containing four digits is usually appended to the Opitz code. The first digit represents the major dimension (either diameter or edge length). The approximate component size can then be determined by using the dimension ratio specified in the geometry. The dimension range is specified from .8 to 80. Dimensions of less than .8 and greater than 80 are represented by a 0 or 9 code respectively. The material type, raw material shape and accuracy are represented by digits 2, 3 and 4.

The Opitz code is concise and easy to use. It has been adopted by many companies as the coding subsystem. Several CAM-I CAPP systems currently use an Opitz based coding system.

2.4.3 The CODE System

CODE is a coding and classification system developed by Manufacturing Data Systems, Inc. (MDSI) [Haan 1977]. CODE is an eight digit code similar to the Opitz system. CODE however has a mixed code structure. Each digit of the code is represented by a hexidecimal value (most systems use decimal numbers). Using hexdecimal numbers, allows more information to be represented with the same number of digits. The structure of the code is shown in Figure 2.35. CODE contains form and dimensional information, but does not include material or accuracy information.

CODE can be used to classify both rotational and non-rotational components. Because more digits in CODE are assigned to auxiliary shapes, better size information can be captured. Instead of classifying the ratio of dimensions, CODE directly classifies major dimensions. Since CODE uses two digits to classify dimensions, 16 sizes can be classified.

	1	2	3	4	5	6	7	8
							DIMENSIONAL	
1	CONCENTRIC OTHER THAN PROFILED	OUTSIDE DIA OR SECTION	CENTER HOLES	OTHER HOLES	GROOVES THREADS	SLOTS AND PROTRUSION	OUTSIDE DIA	LENGTH
							DIMENSIONAL	
2	CONCENTRIC PROFILED	SPECIFIC PROFILE SUCH AS TEETH AND GROOVES		CENTER HOLE	PROFILE SHAPE	NO. OF TEETH SPLINES OR GROOVES	OUTSIDE DIA	LENGTH
						DIMENSIONAL		
3	BENT ROD OR TUBE	BEND ANGLE	DIRECTION OF ENDS	TYPE OF ENDS	TYPICAL SECTION	OUTSIDE DIA	WIDTH	LENGTH
						DIMENSIONAL		
4	BENT OR SEEMINGLY BENT OTHER THAN ROD OR TUBE	NUMBER OF BENDS	OTHER BENDS	BEND ANGLE OR ARC	PROTRUSION CUT-OFFS CUT-OUTS	THICKNESS	WIDTH	LENGTH
						DIMENSIONAL		
5	FLAT	NUMBER OF SIDES	HOLE TYPES	CUT-OFF CUT-OUT	THICKNESS PROTRUSION SLOT, GROOVE HOLES	THICKNESS	WIDTH	LENGTH

Figure 2.34 Summary of CODE Major Divisions

One distinct advantage of CODE is the ready to run software support system offered by MDSI. CODE is not only a coding and classification system, but also a data base system. Information related to the part, such as design data, process plan, etc. can be stored and retrieved by CODE.

2.4.4 KK-3 System

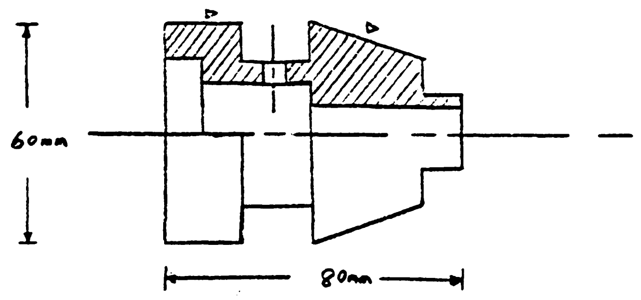
The KK-3 coding system is a general purpose classification and coding system for machining parts. KK-3 was developed by the Japan Society for the Promotion of Machine Industry (JSPMI) [Japan 1980]. Parts to be classified are mainly metal cutting and grinding components. KK-3 was first presented in 1976, and uses a 21 digit decimal system. The code structure for rotational components is shown in Figure 2.35. Because KK-3 has a much longer code than Opitz and CODE, more information can be represented. The KK-3 code includes two digits for component name (functional name) classification. The first digit classifies the general function, such as gears, shafts, drive and moving parts, fixing parts, etc. The second digit describes more detailed functions. For example in gears, there are spur gears, bevel gears, worm gears, etc. With 2 digits, KK-3 can classify only 100 functional names for rotational and non-rotational components. However at times, this can be as confusing as complete. KK-3 also classifies materials using two code digits. The first digit classifies material type, and the second digit classifies shape of the raw material. Dimensions and dimension ratios are also classified. Some redundancy can be found, i.e. both length/diameter and length/diameter ratios are classified for rotation components. Shape detail and process type is classified in KK-3 using thirteen digits of code (much more detail than either the Opitz or CODE systems). An example of coding a component using KK-3 is illustrated in Figure 2.36. A modified version of KK-1 (an earlier version of KK-3) has been reported used for computer-aided process planning (Japan 1980).

2.4.5 The MICLASS System

The MICLASS system was developed by TNO of Holland, and is currently maintained in the US by OIR [Houtzeel 1976 a, b and c]. It is a chain structured code of 12 digits. The code is designed to be universal; therefore, it includes both design and manufacturing information. Information, such as, main shape, shape elements, position of the elements, main dimension, ratio of the dimensions, auxiliary dimension, form tolerance, and the machinability of the material are included (Figure 2.37). An additional six digits

DIGIT	ITEMS (ROTATIONAL COMPONENTS)		
1	PARTS NAME	GENERAL CLASSIFICATION	
2		DETAIL CLASSIFICATION	
3	MATERIALS	GENERAL CLASSIFICATION	
4		DETAIL CLASSIFICATION	
5	CHIEF DIMENSIONS	LENGTH	
6		DIAMETER	
7	PRIMARY SHAPES & RATIO OF MAJOR DIMENSIONS		
8	SHAPE DETAILS & KINDS OF PROCESSES	EXTERNAL SURFACE	EXTERNAL SURFACE & OUTER PRIMARY SHAPE
9			CONCENTRIC SCREW THREADED PARTS
10			FUNCTIONAL CUT-OFF PARTS
11			EXTRAORDINARY SHAPED PARTS
12			FORMING
13			CYLINDRICAL SURFACE
14		INTERNAL SURFACE	INTERNAL PRIMARY SHAPE
15			INTERNAL CURVED SURFACE
16			INTERNAL FLAT SURFACE & CYLINDRICAL SURFACE
17		END SURFACE	
18		NON-CONCEN-TRIC HOLES	REGULARLY LOCATED HOLES
19			SPECIAL HOLES
20		NON-CUTTING PROCESS	
21	ACCURACY		

Figure 2.35 STRUCTURE OF KK-3 (ROTATIONAL COMPONENTS)



Code digit	Item	Component condition	Code
1	} name	control valve	0
2		(others)	9
3	} Material	Copper bar	7
4			0
5	Dimension length	80mm	2
6	Dimension diameter	60mm	2
7	Primary shape & ratio of chief dimension	L/D 1.3	2
8	External surface	with fuctional tap-ered surface	3
9	Concentric screw	Non	0
10	functional cut-off	Non	0
11	Extraordinary shaped	Non	0
12	Forming	Non	0
13	Cylindrical surface ≥ 3	Non	0
14	Internal primary	Piercing hole with dia. variation, NO cut-off	2
15	Internal curved surface	Non	0
16	Internal flar surface	Non	0
17	End surface	Flat	0
18	Regularly located hole	Holes located on cir-comfere tial line	3
19	Spacial hole	Non	0
20	Non-cutting process	Non	0
21	Accuracy	Grinding process on external surface	4

Code for the component: 097022230000020003004

Figure 2.36 Example of KK-3 Coding System

of code are also available for user specified information, i.e. part function, lot size, major machining operation, etc. These supplemental digits provide flexibility for the system expansion. MICLASS was also one of the earliest interactive coding systems (Figures 2.38 illustrates an interactive coding session). MICLASS has been adapted by many U.S. industries. Several application programs based on MICLASS are currently available, such as MITURN - a process planning system for turned parts, MIAPP - a variant process planning, and MIPLAN, another variant process planning system.

2.4.6 <u>DCLASS Systems</u>

The DCLASS system was developed by Dr. Del Allen at Brigham Young University [Allen 1980]. DCLASS was intended to be a decision making and classification system (thus the name DCLASS). DCLASS is a tree structured system (Figure 2.36) which can generate codes for components, materials, processes, machines and tools. For components, an eight digit code is used:

digit 1 - 3:	basic shape
digit 4:	form feature
digit 5:	size
digit 6:	precision
digit 7,8:	material

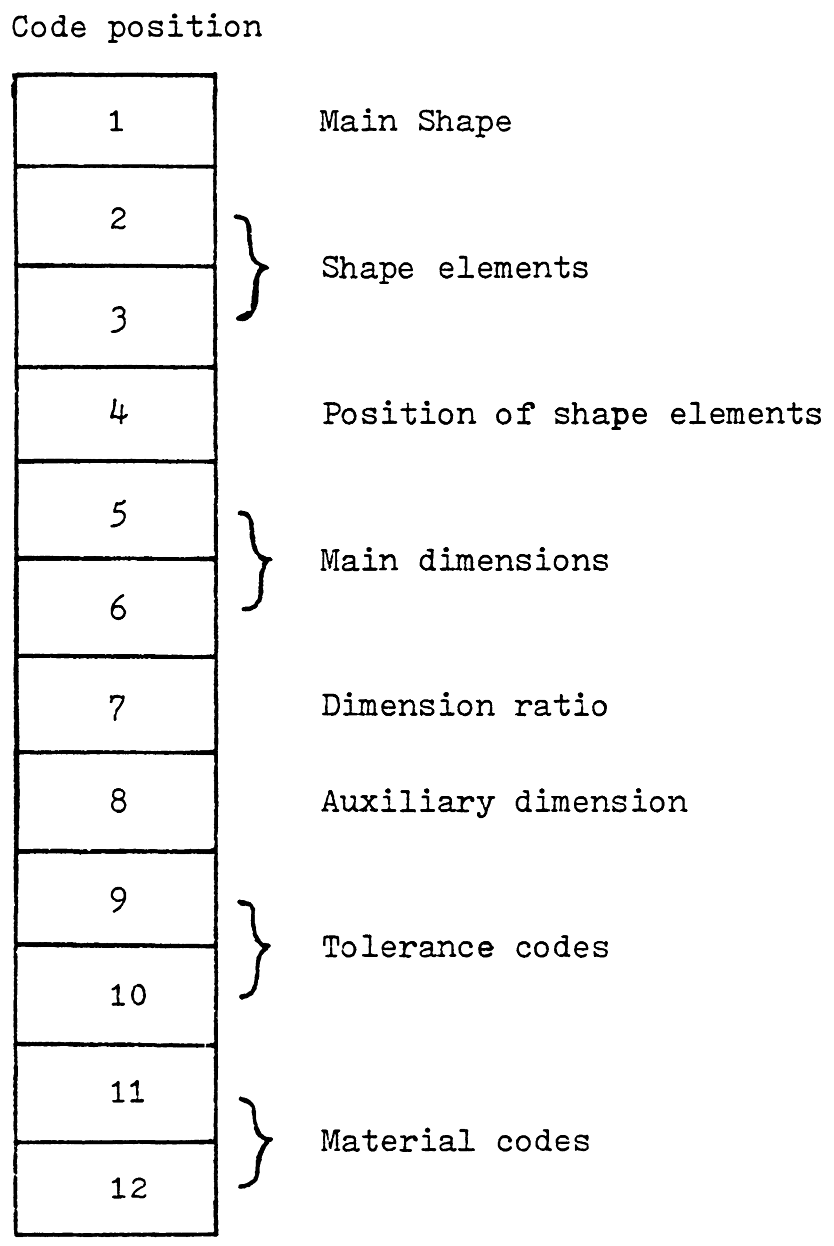
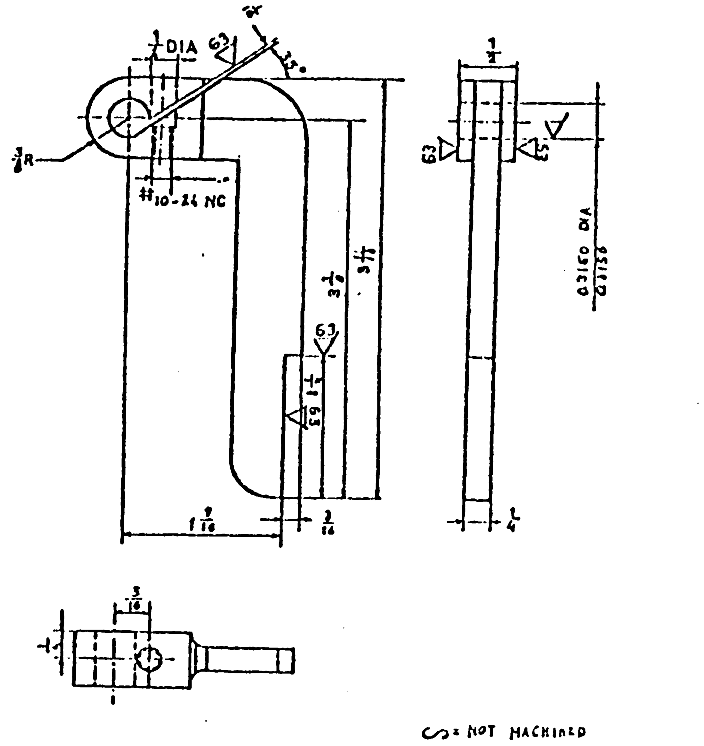


Figure 2.37 MICLASS Code Structure

```
VERSION -A- JAN..1975
3 MAINDIMENSIONS (WHEN ROT. PART D,L AND O)?3.6875 2 .5
BOXFORM?NO
PART FROM BAR- OR SHEETMAT. (NOT BENT)?NO
WELDED, CAST OR ASSEMBLED?YES
 AT LEAST IN 1 CROSS-SECTION SYM.?YES
 MACHINED FACE(S)?YES
 MACHINED SLOT(S)?YES
   MACHINED SLOTS OR FACES CURVED OR SLANTING TO A -B-C?YES
 MACHINED SLOTS OR FACES CURVED?NO
 -ALL MACH-ONLY DIRECTIONS PERP. OR 1 DIR. IN 2 SENSES?NO
 1 DIRECTION NOT PERP.?YES
FORM-OR THREADING TOLERANCE?NO
DIAM. ROUGHNESS LESS THAN 33 RU (MICRO-INCHES)?NO
  NUMBER OF CLOSE TOLERANCE DIAMETERS (NONE=0)
  THAN ON DIFFERENT LINES THE SIZE OF THE DIAMETER,
  SPACE AND THEN TOLERANCEFIELD OR IT-CLASS?1
?.3156 .0006
  SMALLEST POSITIONING TOL. FIELD?.0313
  SMALLEST LENGTHTOL. FIELD?.0313
MATERIALNAME?AL380
CLASS.NR.= 7795 3421 1164
*************************
DRAWINGNUMBER MAX 10 CHAR?18
NOMENCLATURE MAX 15 CHAR?LEVER
CONTINUE(1);STOP(2);SECOND PART AGAIN(3)?2
PROGRAM STOP AT 4690
```



DRAWING TITLE:	TOLERANCES	MATERIAL:
LEVER	Fractional ± 1/64	AL 380
DRAWING NO:	Decimal ±.003	ALL OVER EXCEPT AS NOTED
18		

Figure 2.38 Coding Session of MICLASS (taken from Houtzeel 1976)

In DCLASS, each branch represent a condition, and a code can be found at the terminal (junction) of each branch. Multiple passes of the decision tree allow a complete code to be found. Figure 2.39 illustrates the process by showing a portion of the decision tree for components. One pass on this tree will generate the first three digits of the code. Code construction is established using certain roots (N nodes, and E nodes). At N nodes all branches can be true, and at E nodes only one branch can be true.

The DCLASS system is not only a coding system but also a decision support system. A generative process planning system using DCLASS was reported by Allen [Allen 1979, 1980].

2.4.7 COFORM

CORFORM (COding FOR Machining) was developed at Purdue University by Rose [1977]. CORFORM is neither a commercial system nor a widely used system. COFORM is the coded input used by a generative process planning system (APPAS [Wysk 1977]) and is therefore different than other systems. Some of its specifics are worth discussing because of its uniqueness.

COFORM rather than dealing with the entire part geometry, limits itself to the description of individual machined sur-

8 DIGIT PART FAMILY CODE

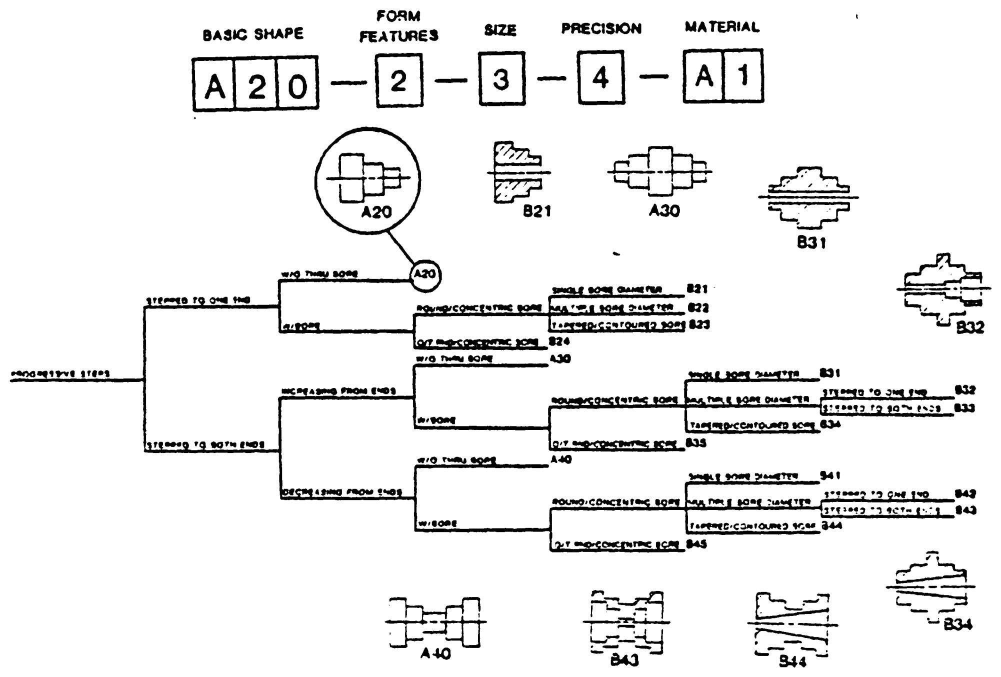


Figure 2.39 DCLASS Structure

faces (holes, slots, and general surfaces). The code describes each surface in terms of the attributes needed to select the appropriate machining process(es) and the related parameters (feed, speed, and depth of cut). Figure 2.40 contains a list of the attributes required to describe holes. The entire component can be described by decomposing it into several surfaces. However, COFORM does not contain a description of non-machined features. Therefore COFORM can not be used for a CAD system even though it is quite detailed.

COFORM does not code each feature. Instead, it records the data values for some features, i.e. diameter, length, tolerances, etc. It is not considered a GT coding and classification system, but, rather a component description system using mixed codes and data values.

2.4.8 Closing Remarks

Process planning is necessary to plan the detailed processes for the production of a design. The input to a process planner or a process planning system is always a design. In this chapter different types of design representations have been discussed, from the most conventional design (drafting) to the modern CAD systems. An important design representation method which has been widely

ATRIB(1) - Surface or hole identification number, ID number is userassigned and can be between 1 and 99999

ATRIB(2) - The type of surface of program entry being coded

1 = round hole
2 = plane or other surface
3 = slot
4 = symmetric turned surface
5 = gear
6 = spline
7 = unround hole
8 = other
9 = end of information

ATRIB(3) - ATRIB(3) is dependent on ATRIB(2). The following description is for (ATRIB(2)=1)

1 = simple, cylindrical hole
2 = tapered or conic hole
3 = threaded cylindrical hole
4 = threaded tapered hole
5 = cylindrical hole with keyway
6 = non-cylindrical hole with keyway
7 = threaded hole with a keyway
8 = a hole used to locate a surface
9 = a hole used to locate another hole

ATRIB(4) - The number of diameters or locating surfaces dependent on the current hole

ATRIB(5) - aterial type

1 = grey cast iron
2 = malleable iron
3 = steel
4 = steel forging

ATRIB(6) - Material hardness (BHN)

ATRIB(7) - Dimensionality of the hole-diameter

ATRIB(8) - Tolerance on diameter-plus

ATRIB(9) - Tolerance on diameter-minus

ATRIB(10) - Surface finish - (side) - CLA

ATRIB(11) - Length of diameter

ATRIB(12) - Tolerance on lengthm $\pm$ X

ATRIB(13) - Direction cosine of surface

ATRIB(14) - Straightness

ATRIB(15) - Roundness

ATRIB(16) - Parallelism

Figure 2.40 <u>Description of the Attributes used in COFORM</u> (to be continue)

ATRIB(17) - Perpendicularity

ATRIB(18) - Angularity

ATRIB(19) - Concentricity

ATRIB(20) - Symmetry

ATRIB(21) - True position

ATRIB(22) - Profile of a surface

ATRIB(23) - Profile of a line

ATRIB(24) - Cored hole indicator

ATRIB(25) - Maximum stock removal when hole is cored

ATRIB(26) - Initial surface codition indicator

ATRIB(27) - Chamfer code

0 = no chamfer
1 = chamfer
2 = concave radius (inner fillet)
3 = convex radius (outer fillet)

ATRIB(28) - Angle of chamfer or dimension of radius
If ATRIB(28)=1 $35^\circ \leq$ angle $\leq 80^\circ$

ATRIB(29) - Length of chamfer or radius (maximum)

ATRIB(30) - Tolerance of chamfer length

ATRIB(31) - Threads/inch (0 for no tapping operation)

ATRIB(32) - Thread tolerance

ATRIB(33) - Thread series

1 = unified standard or American standard
2 = others

ATRIB(34) - Bottom hole indication

ATRIB(35) - Bottom hole value

ATRIB(36) - Surface finish (bottom) - CLA

ATRIB(37) - Interrupter cut indicator

ATRIB(38) - Reference surface number

ATRIB(39) - Access surface indicator

ATRIB(40) - The number of holes to be produced by one tool.
If ATRIB(40) is negating minimum production time for each hole is to be obtained.

Figure 2.40 (continue)

used in the computer-aided process planning -- GT coding, was also discussed.

It is necessary to understand the data one is working on because the efficiency and accuracy of a system is greatly affected by the way information is represented and interpreted. Only after one understands the data he has can be make intelligent decisions. In the next chapter the knowledge needed to make these decisions in process planning will be discussed.

Chapter III

PROCESS ENGINEERING

Process planning is the activity which determines the appropriate procedures to transform a raw material into a final product. The final product is specified in the engineering design (either a drawing on paper or a CAD model). Selecting the manufacturing processes to transform the raw material into finished specification is based on matching the design requirements with process capabilities. Process capability is the data base of knowledge for each process. It includes:

1. the shapes and size a process can produce,
2. the dimensions and geometric tolerances which can be obtained, by various processes,
3. the surface finish attainable,
4. material removing rate,
5. relative cost, and
6. other cutting characteristics/constraints.

Process capability does not necessarily imply that all process selection is based on the above information. However the more information considered in selecting a process, the more complete the result will be. For conventional process planning (where human planners are used) all process capability information comes either in the form of experience or handbood lists and guides. A computer-aided process planning system functions based on this process capability information (see Section 1.3). A pure variant planning system is a pure retrieval system, and is analogous to planning based on experience. In a variant planning system, standard plans are stored based on component shape. These plans are then retrieved based on the similarity of a coded part. Only some vary basic information such as computing machining time is calculated in the system. A generative system, however, makes processing, tooling and other decisions via software program logic. In order to select an appropriate process(es), the process capabilities must be stored in the system.

The basic mechanisms of process selection will be discussed in the next chapter. Because the data structure of the process capabilities is planning method dependent, a discussion of process selection mechanisms and process capability information representation will be presented in this chapter.

3.1 EXPERIENCE BASED PLANNING

It is always true that the accumulation of experience is knowledge. Most of our knowledge comes from either our own experience or the experience passed on to us by others. This is also true in the context of manufacturing processes. If we go back to the example of the farmer and blacksmith in Chapter 2, we can make a point of how this experience works in a manufacturing environment. When the farmer orally specified or drew a picture of the hoe he wanted, the blacksmith must have had a manufacturing method (process plan) in mind; otherwise, he would not have been able to make the hoe. The process plan the blacksmith created was neither a written one, nor a complete one. However, in his mind he knew he had some scrap iron in the back of his shop which he could use for the hoe. After he fetched the material, heated it in his furnace until it reached red heat (hot working) and then completed the forging cyle. Had the farmer asked, "How do you know to convert your scrap to my hoe?", the answer most assuredly would have been something like, "I've been in the trade for thirty years, experience told me."

Even in this space age, many process planning activities still rely on the experience of process planners. Where do process planners obtain their experience? From earlier training as a machinist (most typical), from books, or discussions with colleagues. This kind of information can be passed from person to person and generation to generation. However, there are some problems associated with this planning base.

1. Experience requires a significant period to accumulate.
2. Experience only represents approximate, and not exact, knowledge.
3. Experience is not applicable to new processes or new systems.

Because of the above problems, we need to seek other ways to represent our process capability knowledge base so that it might be preserved and so that it might be installed as a decision support system on a computer.

3.1.1 Machinist's Handbooks

One way to store process capability information is to print it in handbooks. This has long been a standard manufacturing practice. Process capability information is usually presented in tables, figures, or listed as guidelines. Large manufacturers typically prepare their own handbooks for internal use. Therefore, the knowledge is kept, but has traditionally been "proprietary". Handbooks can serve both as a reference and as a guide for process selection. Figure

3.1 represents some typical information of process capability (surface finish ranges). The surface finish ranges chart shows the limiting extremes of several processes. For example, a flat surface of 8 microinch surface finish can be machined by grinding, polishing and lapping. It can rarely be achieved by milling, but a surface finish of 8 microinches is still possible. Other information such as process accuracy can be found in similar tables or charts (see for example Table 3.1). In Table 3.1, an accuracy class 10 for drilling is considered highly accurate, but for reaming, this is only considered moderately accurate.

Some process capability information is listed as guidelines, so process planners can follow some general rules. For example, the following guidelines can be applied to produce holes.

1. Dia. =< .5"
   - A. True position > 0.010
     1. Tolerance > 0.010

        Drill the hole
     2. Tolerance =< 0.010

        Drill and ream
   - B. True position =< 0.01
     1. True position =< 0.01

        Drill then finish bore or ream
     2. Tolerance =< 0.002

Surface-finish ranges

Here are typical ranges of surface-roughness-height values obtainable by various production methods. Because of the many variables in processing, this information is for general guidance only, and the values shown below should not be considered as fixed limits.
Reprinted from *General Motors Drafting Standards*, August, 1971. ■

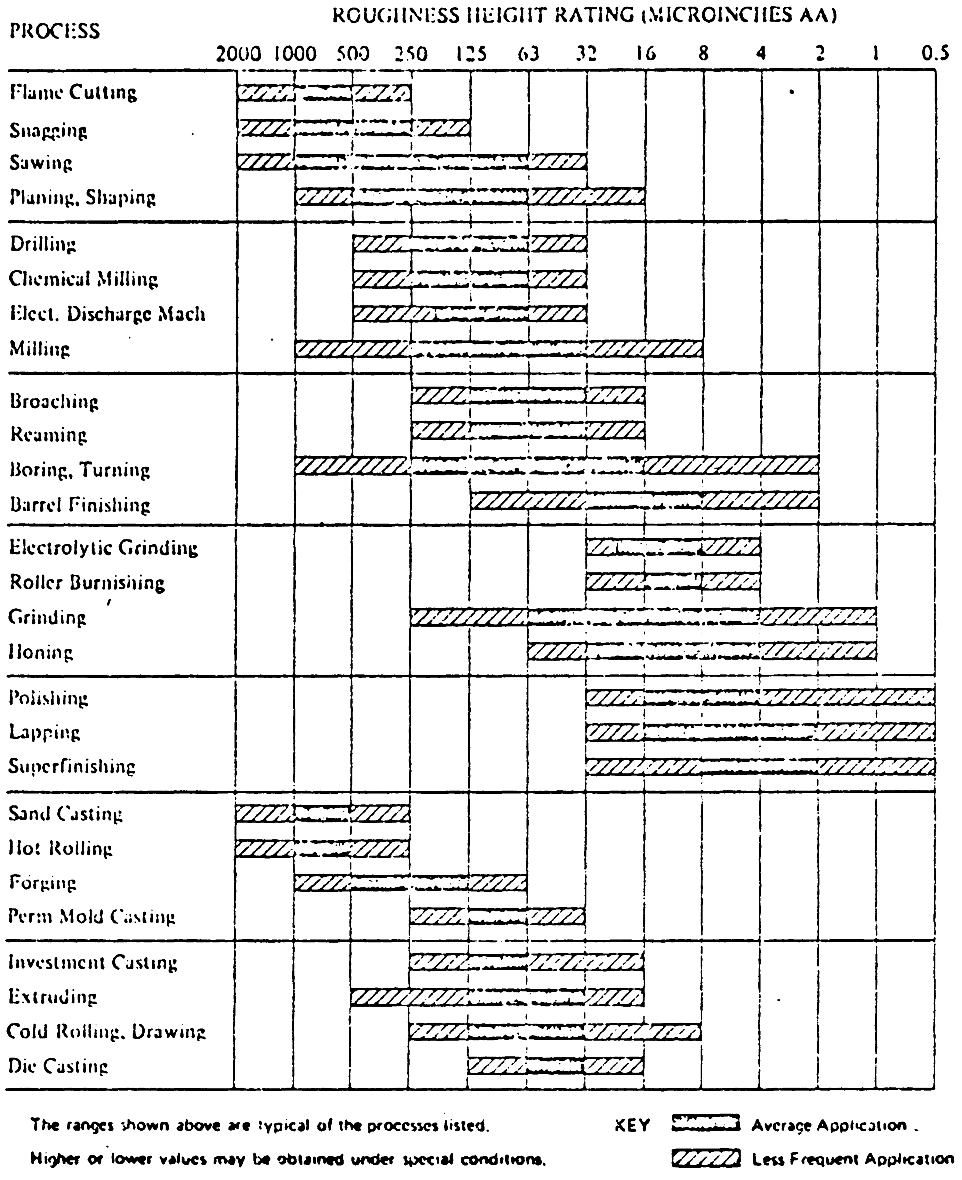


Figure 3.1 Surface-finish ranges

TABLE 3.1

Principles of Machining by Cutting, Abrasion and Erosion

(taken from Kaczmarek 1977)

Machining method	Classes (according to ISO)																									
	of accuracy												of surface quality													
	1 to 3	4	5	6	7	8	9	10	11	12	13	14 to 16	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I. CHIP REMOVING PROCESSES																										
Turning				●	●	o	o	o	x	x	—	—	—	—	—	x	x	o	o	●						
Boring				●	●	o	o	o	x	x	—	—	—	—	—	x	x	o	o	●						
Drilling								●	o	x	x	—	—	—	x	x	o	●								
Reaming			●	●	o	o	x	x	—	—							—	x	o	o	●	●				
Peripheral milling					●	●	o	o	x	x	—	—	—	—	x	x	o	o	●	●						
Face milling					●	●	o	o	x	x	—	—	—	—	x	x	o	o	●	●						
Planing and shaping						●	o	o	o	x	x	—	--	—	x	x	o	o	●	●						
Broaching			●	●	o	o	x	x	—	—	—	—				—	x	o	o	●	●					
II. ABRASION PROCESSES																										
A. *Using abrasive tools*																										
Centre-type cylindrical grinding			●	o	o	x	x	—	—								—	x	x	o	●	●				
Centreless cylindrical grinding			●	o	x	x	x	—	—								—	x	x	o	o	●				

TABLE 3.1 (continued)

Internal grinding			●	o	o	x	x	—	—								—	x	x	o	●	●				
Surface grinding			●	o	o	x	x	—	—	—							—	x	x	o	o	●				
Abrasive belt grinding				●	o	o	x	x	—	—						—	x	x	o	o	●					
Surface honing		●	●	o	o	x	—	—										—	—	x	x	o	●			
Shaft and internal honing	●	●	●	o	o	x	—											—	—	x	x	o	o	●		
Superfinish	●	●	o	o	x	x	—												—	—	x	o	o	●	●	
B. *Using loose abrasive*																										
Lapping		●	●	o	o	x	x	—									—	—	x	o	o	●	●			
Mechanical polishing	●	●	o	o	x	x	—	—										—	—	x	x	o	o	●	●	●
Vibratory and barrel finishing					●	●	●	o	o	x	x	—			—	—	x	x	o	●						
Abrasive-blast treatment				●	●	●	o	o	x	x	—	—			—	—	x	o	o	●	●					
Ultrasonic machining				●	●	o	o	x	x	—						—	x	o	o	●	●					

Accuracy	Surface quality
— rough	— rough
x fairly accurate	x fairly smooth
o accurate	o smooth
● highly accurate	● very smooth

Drill, semi-finish bore then finish bore

2. 0.5" < Dia =< 1.00

etc.

3.2 DECISION TABLES AND DECISION TREES

Decision tables and decision trees are methods of describing or specifying the various actions (decision) associated with combinations of (input) conditions. Both of these have been used for a long time to help systematize decision making. With the advent of the digital computer, an increased use of these tools has occurred. Both of these methods are described using the following example. The activity deals with a weekend decision which can be described as follows:

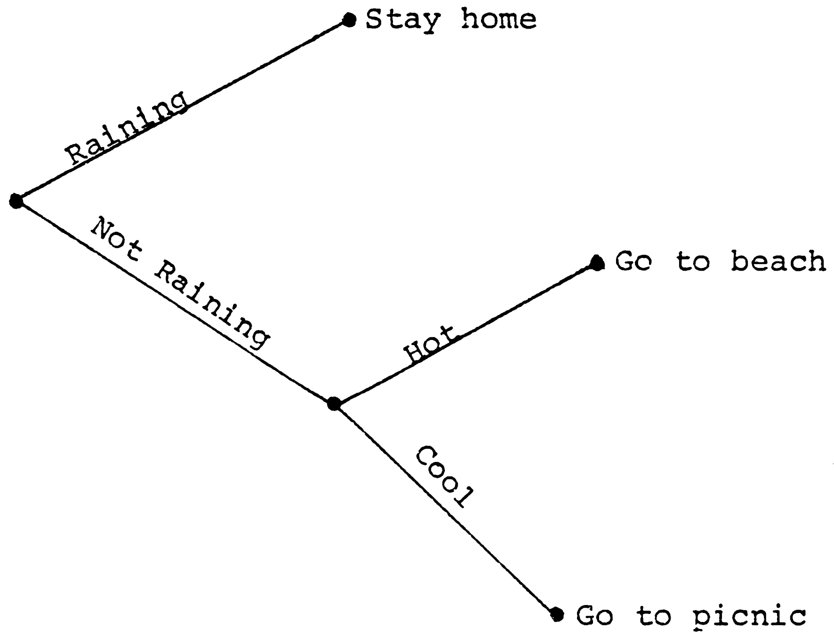
"If it is raining, then I will go to the arcade and play video games";
"If it is not raining and hot, then I will go to the beach";
"If it is not raining and cool, then I will go on a picnic with my friend".

The above plan can also be represented as either a decision table or decision tree (see Figure 3.2).

In our decision table, T implies true to a specified condition, F implies false and a blank entry implies that we do not care. Only all conditions in a decision table column are met, then the marked action is taken.

Raining	T	F	F
Hot		T	F
Stay Home	X		
Go to beach		X	
Go to picnic			X

a. Decision table



b. Decision tree

Figure 3.2 Decision Table and Decision Tree Representation

In a decision tree, the conditions are specified on the branches of the tree. When all of the branches leading to an action hold true, then the action at the terminal point is taken. This system can be used to represent the decision for the above personal recreation plan, as well as a knowledge base for process planning.

Both decision tables and decision trees are tools to assist in decision making. Since decision rules must cover all possible situations, they must be thought out before such a tool can be used for process planning. For a given set of decision rules, one can list the conditions and actions verbally as shown in the example, and then translate the actions into decision table or decision tree form. Whatever can be represented by a decision table can certainly be represented by a decision tree. The primary difference is the ease and elegance of presentation and programming if a computer is used.

3.2.1 Decision Tables

A decision table is partitioned (conditions and decisions) by vertical and horizontal lines (Figure 3.3). The portion of the table above the horizontal line specifies the condition, while the portion below that line indicates the action. The left portion of the vertical line contains the stub (or tag) and right portion, the entries. Decision rules are identified by columns in the entry part of the decision table. Based on the rule representation, decision tables can be classified as:

1. Limited-entry decision tables.

   The condition stub/tag specifies exactly what the conditions are (the value of the input variable(s)). Condition entries can only be T, F or do not care. (See Figure 3.2)

2. Extended-entry decision tables.

   The condtion stub/tag specifies the identification of the condition, but not the value. Values are specified in the condition entries (See Figure 3.4).

3. Mixed-entry decision Tables.

   Sequenced as well as unsequenced actions can be identified using mixed-entry decision tables. A portion of a decision may be made then refined to compute the decision using mixed-entry systems. For sequenced actions a sequence number is assigned to each applicable action entry (Figure 3.5). Unsequenced actions do not require a sequence number. Therefore, only an "X" is used in the table (Blank action entries implies do not perform).

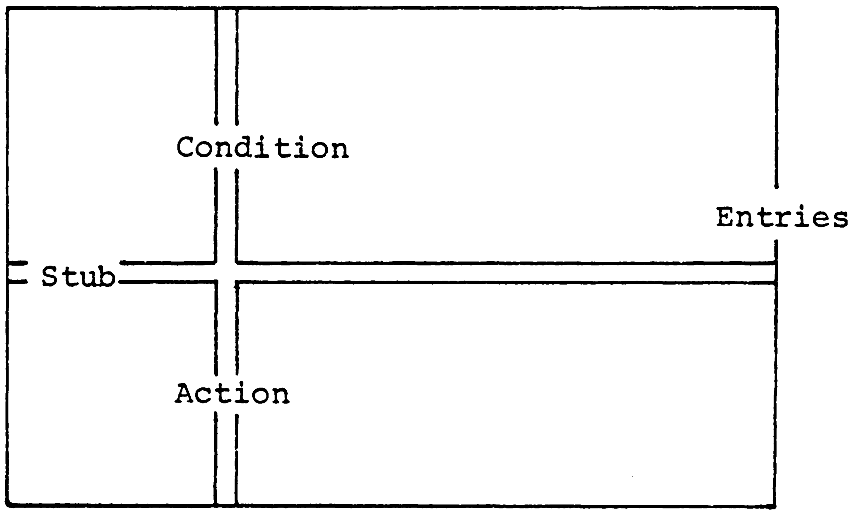


Figure 3.3 Decision Table Patitions

Temperature		≥ 80	< 80
Weather	Raining	Not Raining	Not Raining
Stay home	X		
Go to beach		X	
Go to picnic			X

Figure 3.4 Extended-entry Decision Table

True position > 0.01	T	T	
T. P. < 0.1			
Tolerance $\geq$ 0.1	T		
Tolerance $\leq$ 0.1		T	
Drill	X	1	
Ream		2	
Bore			

(a partial table)

Figure 3.5 Sequenced Action

		A	B	C
Condition	1	T	T	T
	2			
	3	T	T	
	4		F	
Action	1	X	X	
	2			X
	3			X

Figure 3.6 Redundant Rules

When constructing a decision table, the following factors must be considered: completeness, accuracy, redundancy, inconsistency, loops, and size. Completeness is somewhat obvous and is required for every specification. This rule must always be satisfied. An incomplete decision table will lead to uncertain actions. Decision table accuracy is always important. A decision table must represent the original rules which were specified. Redundancy occurs when two rules have the same action(s) and their condition sets overlap. For example in Figure 3.6, rules A, B are redundant. When conditions 1 and 3 are true and condition 4 is false, both rules will select action 1. However, when conditions 1 and 3 are true, rule A will conclude action 1 and rule C will conclude actions 2 and 3. The conflict is called an inconsistency, and an inconsistent decision table produces erroneous decisions.

When an action is used to change conditions, and recurrent calls to the table are used, an endless loop can result. The process will never terminate with a conclusion from the table. For example, the initial condition of a machined hole has a diameter of 2", surface finish 60μ inch, and dimensional tolerance 0.005". Using the decision table given in Figure 3.7 and a backward planning method, the following processes can be found. For this example, Rule C must first be satisfied; therefore, a boring process is selected as the "final operation", and the surface finish is changed to 60 μ inch. Rule B is then satisifed and a remaining process is selected. Since the tolerance does not change, ruler B is always satisfied, and the process never ends.

The size of the decision table is also important. If a decision table requires several pages of print, it is difficult for a human to read. The same also holds true for a computer augmented decision table. A "large" decision table in a computer will not only require excessive memory, but also reduce the efficiency of the decision making. In order to reduce the table size, the following steps can be used.

a. Merge those rules with a common action and only one different condition (Figure 3.8). The merged rule has a "do not care" in that different condition entry.

b. If the table is still too large, it can be split, decision table parsing can be used to connect the separate tables (Figure 3.9).

A more detailed discussion on decision table techniques can be found in Metzner and Barnes [1977] and Montalbano [1974].

	A	B	C	D
T.P. > .01	T			
Tolerance > .01	T	F		
S.F. ≥ 60		T	F	
ia = 0	F	F	F	T
Drill	X			
Ream		X		
Bore			X	
Dia = 0	X			
S.F. = 60			X	
Terminate				X

S.F.: 40

Dia: 2

Tol: .005

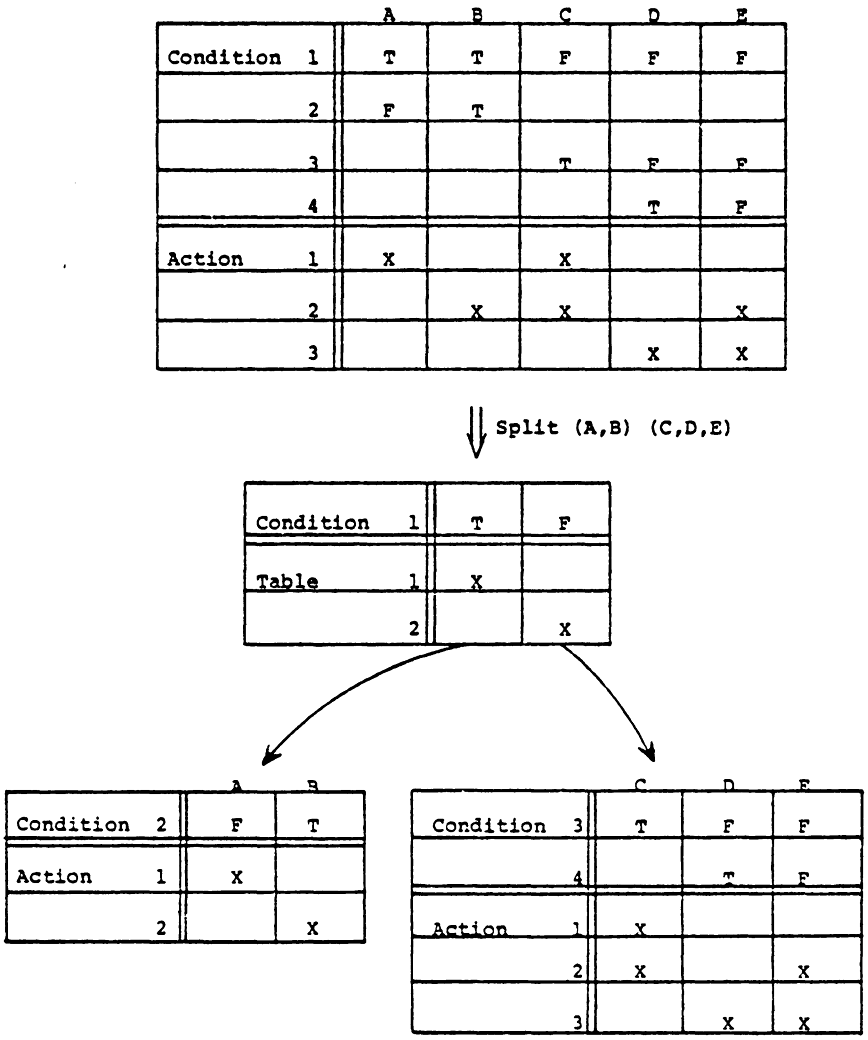
Figure 3.7 Endless Loop

		A	B	C
Condition	1	T		F
	2		F	
	3	F	T	F
	4	F	T	F
Action	1	X		X
	2		X	
	3		X	

$\Downarrow$ Merge A,C

		A'	B
Condition	1		
	2		F
	3	F	T
	4	F	T
Action	1	X	
	2		X
	3		X

Figure 3.8 <u>Rule Merge</u>



		A	B	C	D	E
Condition	1	T	T	F	F	F
	2	F	T			
	3			T	F	F
	4				T	F
Action	1	X		X		
	2		X	X		X
	3				X	X

Condition	1	T	F
Table	1	X	
	2		X

		A	B
Condition	2	F	T
Action	1	X	
	2		X

		C	D	E
Condition	3	T	F	F
	4		T	F
Action	1	X		
	2	X		X
	3		X	X

Figure 3.9 Table Splitting and Parsing

Several existing process planning systems use decision table techniques. Since most process knowledge can be represented by "If .... THEN .... process, AND ...." rules, this information can be implemented using decision tables. Figure 3.10 shows a partial decision table for hole making process selection.

3.2.2 Decision Trees

A decision tree is a graph with a single root and branches eminating from the root. Each branch communicates a value. In probability, decision trees are used to represent the outcome of events or the probability of a state transition. Figure 3.11 illustrates a decision tree which represents the transition probabilities from state A to other states in a three state Markov process. The value on the branches are the one stage transition probabilities. When used in decision making, branches usually carry values or expressions (Figure 3.12). Each branch represents an "IF" statement, and branches in serial represent a logical "AND". Therefore, a path from the root to the terminal can represent a rule similar to that in the decision table. The action is listed at the junction of each terminal branch.

Decision trees are composed of: a root, nodes, and branches (Figure 3.13). The root is the source of the tree,

dia $\le$ .5	T	T	T	T	F	F	F	F	F	F	F	F
.5 < dia $\le$ 1.0	F	F	F	F	T	T	T	T	T	F	F	F
1.0 < dia $\le$ 2.0	F	F	F	F	F	F	F	F	F			
10 < dia	F	F	F	F	F	F	F	F	F			
T.P.$\le$ .002	F	F			F	F	F	F	T	F	F	
.002 < T.P.$\le$ .01	F	F			F	F	T	T	F	F		
.01 < T.P.	T	T	F	F	T	T	F	F	F	T		
Tol $\le$ .02	F		F	T	F		F	T	T	F	F	T
.002 < Tol $\le$ .01	F			F	F		T	F	F	F	T	F
.01 < Tol	T	F		F	T	F	F	F	F	T	F	F
Drill	1	1	1	1	1	1	1	1	1	1	1	1
Ream		2										
Semi-finish bore				2				2	2			2
Finish bore			2	3		2	2	3	3		2	3
Rapid travel			3						4		3	4

T.P.: True position

Figure 3.10 Decision Table For Hole Making Process

	To		
	A	B	C
A	.6	.3	.1
B	.2	.5	.3
C	.1	.2	.7

Markov Chain Transition Matrix

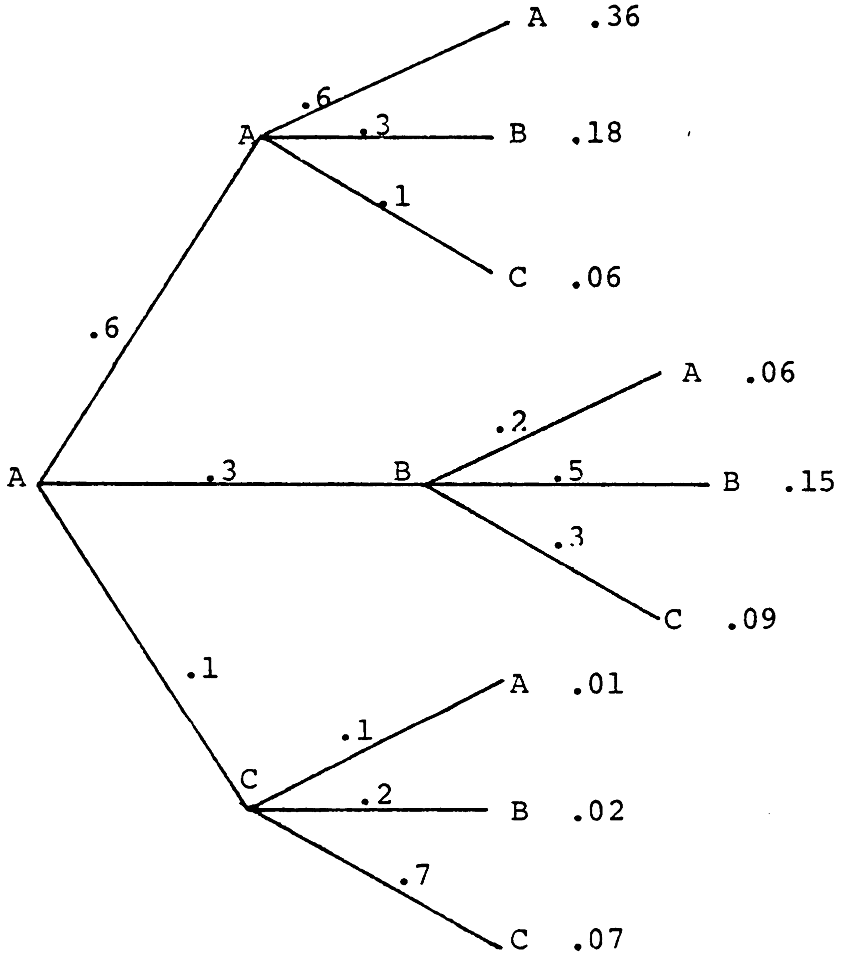


Figure 3.11 <u>A Decision Tree to Show the Probabilities</u>

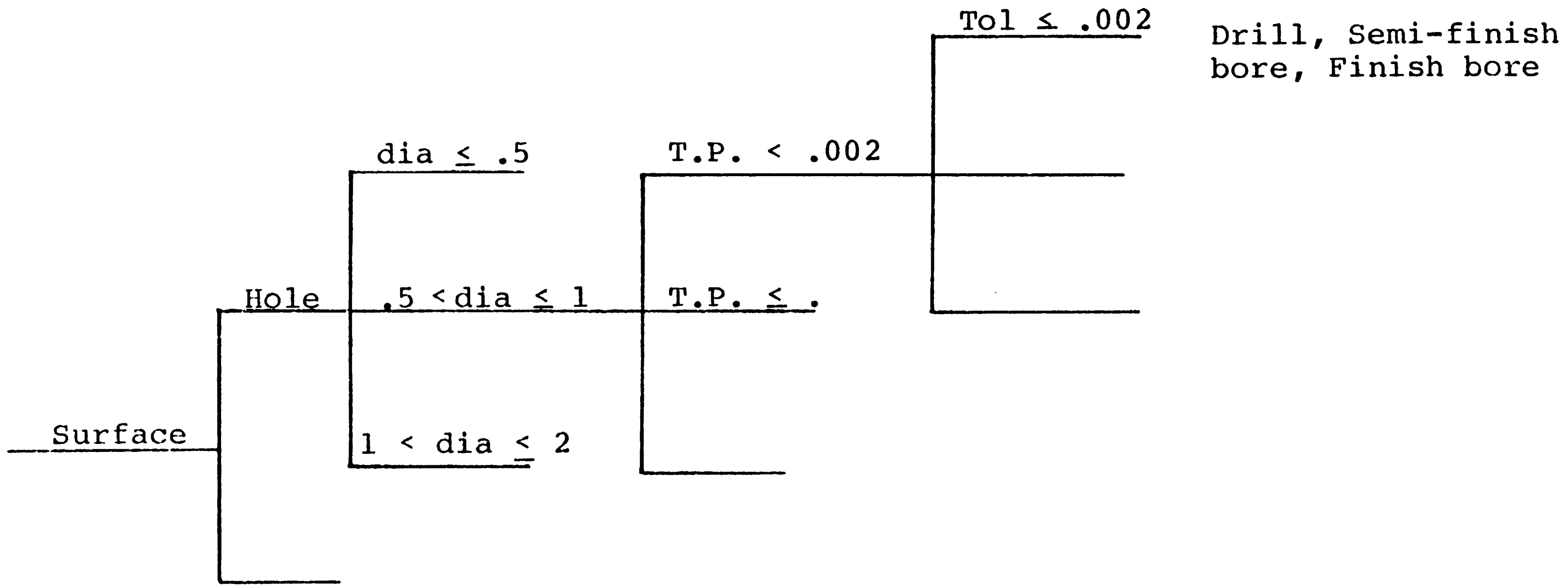


Figure 3.12 Decision Tree

and each tree can have only a single root. However there are nodes which have properties similar to roots, except nodes are preceeded by other branches. Both roots and nodes have branches eminating from them. Branches can only have two logic values -- True or False. When a branch is True in a specific situation, it can be traversed to the next node. There are two types of nodes, mutually excrusive and non-mutually excursive [Allen 1981]. A mutually excrusive node allows at most one of its successive branches to be true. But a non-mutually excrusive node allows all the successive branches to be traversed concurrently. By using both types of nodes, we can construct a decision tree to select actions. A decision tree representation of process knowledge can be found in Figure 3.13.

There are few other methods for decision making except decision tables and decision trees. We will show some of these methods in the next chapter. Next we will discuss the knowledge which is built into the decision logic.

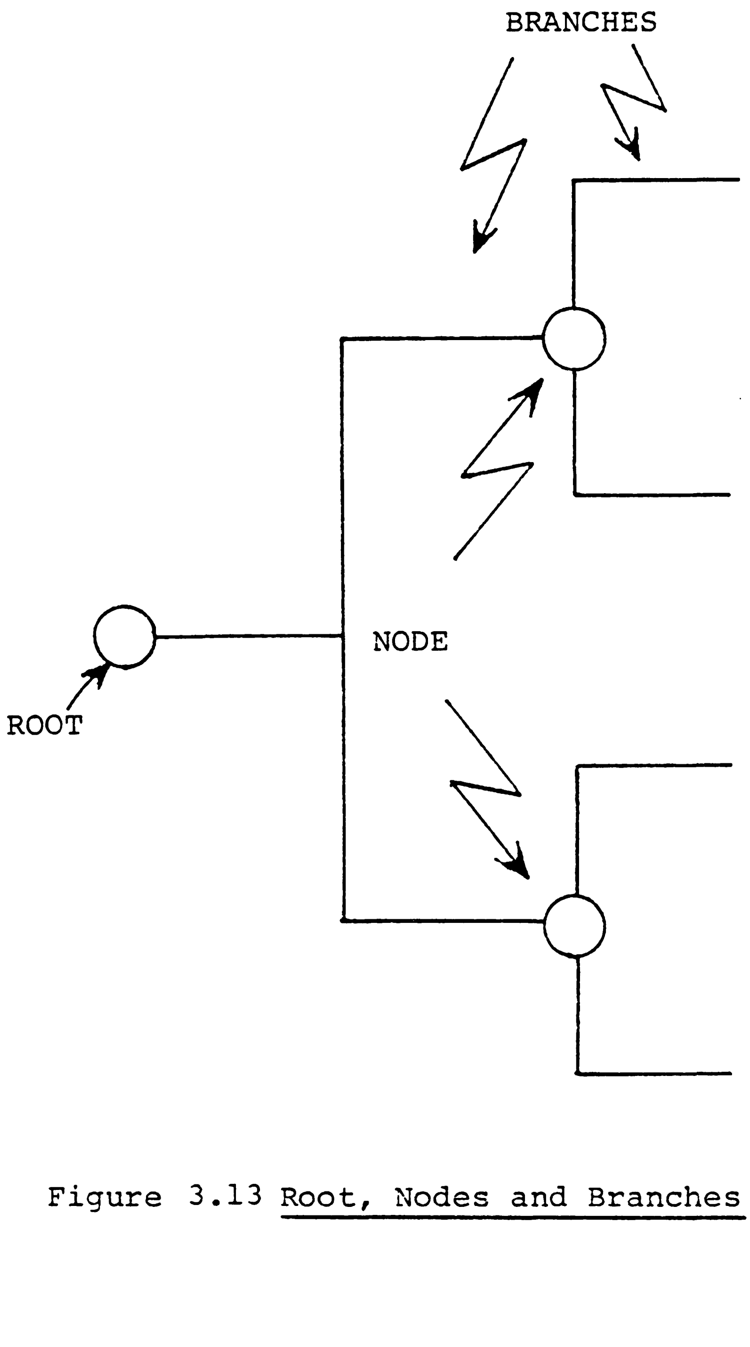


Figure 3.13 Root, Nodes and Branches

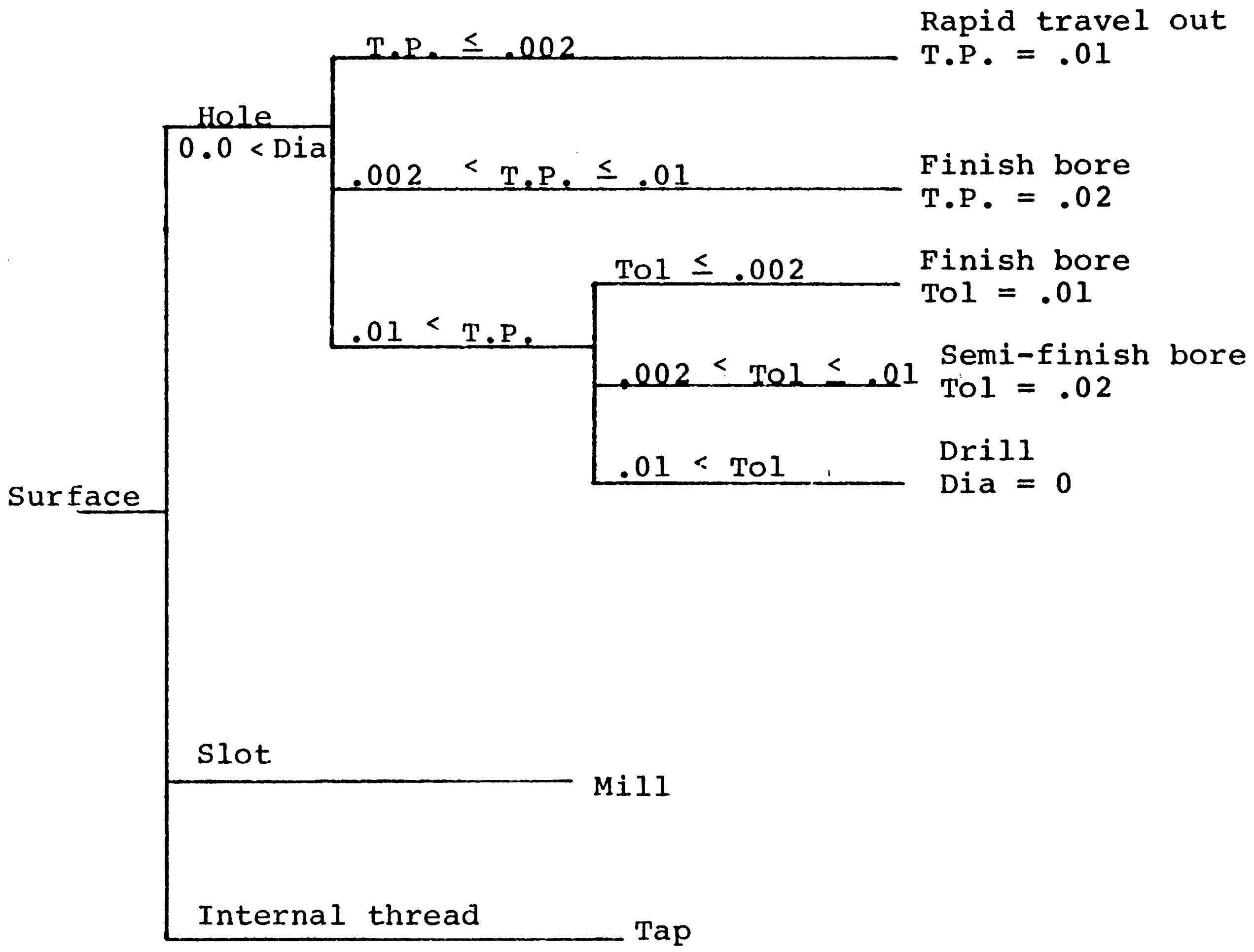


Figure 3.14 Decision Tree for Process Selection

3.3 PROCESS CAPABILITY ANALYSIS

Now that different process knowledge representation methods have been discussed, we can return our focus to the "knowledge" that will be represented in our system. This knowledge is the process capabilities that we begin discussing earlier in the chapter. Again, this knowledge base will be limited to chip-metal removal processes.

Before any decision can be made (whether it be experience, table or tree based), the information required to make the decision must be completed. This information includes a process-by-process breakdown of the following elements:

1. Shape(s) that a process can generate,
2. Size limitation (boundaries of the tooling, machine tools, fixtures),
3. Tolernaces (both dimensional and geometric),
4. Surface finish (as a limiting value or functional expression),
5. Cutting force, and
6. Power consumption.

The first four elements are capabilities and the last two are limitations. The limitations form the constraints that the machinery is bounded by during processing. The shape elements imply/define the basic geometry produceable by a process (See Table 3.2). With little training, virtually everyone can develop a set of shapes which can be produced by a process. However this feature is perhaps the most difficult task to achieve using a computer, i.e. reasoning the shape capability in its natural form (geometry and topology), especially for sophisticated components. A feasible alternative is to represent shape by a code. Human judgement can be used to identify machined surfaces and assign codes to them. The matching of shape capability and machined surface may not be automatic but significantly simplified.

For internal machining processes (holes, etc.) the size capability is constrained by the available tool size. In external machining, size is constrained by the available machine table size (machine cube). The other capabilities and limitations can be expressed mathematically. These expressions are straight-forward to program on a computer if the "exact" equations and constraints can be found.

3.3.1 Process Boundaries

Process capabilities can be represented by process boundaries. A process boundary is interpreted as the limiting size, tolerances and surface finish for a process. It is expressed as the best (or worst) case result of a process. Such a result can be obtained by careful control of the cutting condition and process parameters (feed, speed, depth of

Table 3.2 Shape Capability

Process	Shape Capability
Turning	External surfaces which can be generated by rotating a line or curve around an axis
Boring	Hole
Drilling	Hole
Reaming	Hole
Face Milling	Flat surface
Peripheral Milling	Flat surface, slot,
Shaping & Planing	Flat surface
End Milling	Hole, flat surface, curved surface, slot
Grinding	Flat surface, hole, external cylindrical surface
Honing	Hole
Taping	Internal Thread

cut). Figure 3.15 contains a typical process boundary table.

Size boundaries are determined by available tool sizes or machine table sizes. This limitation is purely a function of the production system. In a small custom machine shop, the largest spindle for their lathes may be 10" in diameter. Therefore, the upper boundary for turning is a 10" diameter. However in a shipyard, the largest lathe may be able to accomodate a 3' diameter component. The upper boundary for turning would be 3' instead of 10". Other process boundaries are also system dependent.

For instance, dimensional tolerance is affected by many variables, e.g. tool diameter, machine tool accuracy, vibration, etc. Several sources [Trucks 1974, Eary 1962] have developed tables, charts or functions which show a tolerance dependency as a function of surface size and perhaps other variables such as material or operational parameters. Scarr [1967] suggested that the tolerance for all hole making processes can be expressed in a general form.

Tol = A(D) ** N + B .... (3.1)

where

Tol:	tolerance
A:	coefficient of the process
N:	exponent describing the process

Boundary	Hole Producing Processes	Plane Producing Processes
Smallest tool size	S_h	S_s
Largest tool size	L_h	L_s
Negative tolerance	$A_1(DIA)^{n_1} + B_1$	$A_1(DIA)^{n_1} + B_1$
Positive tolerance	$A_2(DIA)^{n_2} + B_2$	$A_2(DIA)^{n_2} + B_2$
Straightness (holes)	$A_3(Len/DIA)^{n_3} + B_3$	$A_3 \frac{\text{Depth of cut} \cdot \text{length}}{DIA} + B_3$
Parallelism	$A(\frac{Len}{DIA})^{n_5} + B$	$A_5(\frac{Len}{DIA})^{n_5} + B_5$
Roundness (holes) Angularity (planes)	R_h A_4	A_S A_4
Depth Limit	D_h $A_6 \cdot Dia$	$A_6(DIA) + B_6$
True position	A_7	
Surface Finish	A_8	A_8

Figure 3.15 Process Boundary Table (taken from Wysk 1977)

B: constant describing the best tolerance attainable by the process

D: hole diamter

Likewise this general form can be used to represent other processes. The values of the coefficients A, B, and N in the equation can be obtained by experimentation. Again, it is worth noting that, A, B, and N are system dependent i.e., no universal parameters can be found.

Straightness and parallelism tolerances for holes define the axial tolerance. The major cause of error on straightness and parallelism is tool deflection. A tool can most simply be modeled as a cantilever beam (Figure 3.16). The deflection of the beam is:

$$\delta = \frac{P\,l^3}{3EI} \qquad (3.2)$$

where

E: modulus of elasticity

I: moment of inertia

l: beam length

P: force

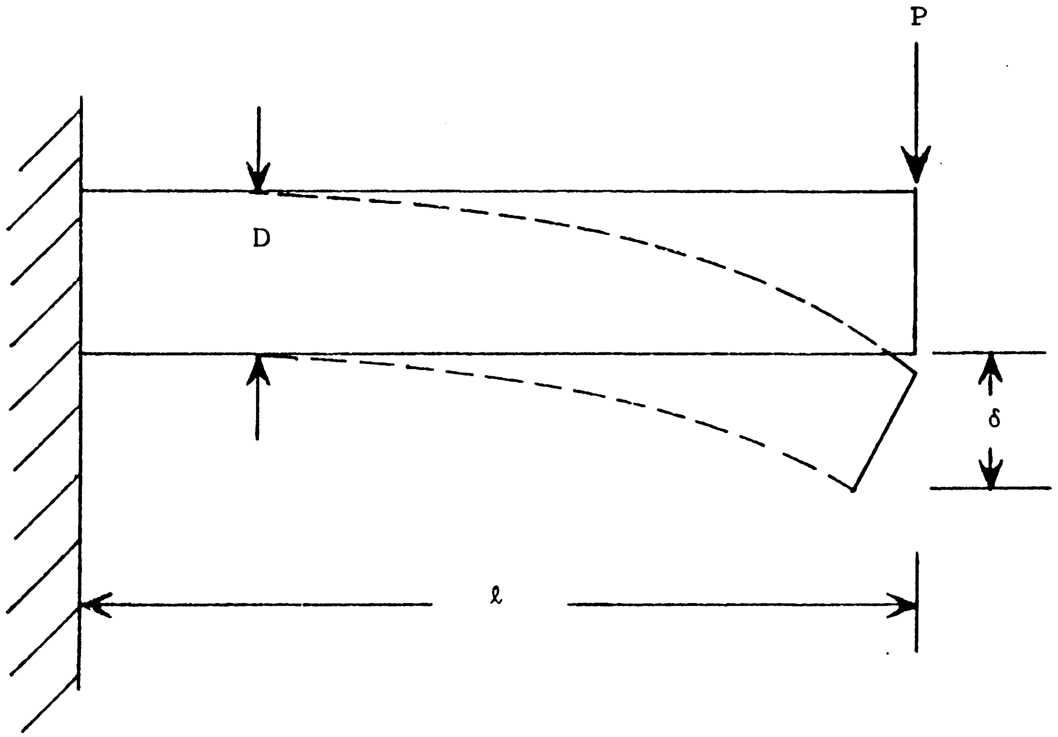


Figure 3.16 Cantilever Bean

$$I = \frac{\pi D^4}{64} \quad \text{for cylindrical beam or tool} \tag{3.3}$$

$$\delta = \frac{64\ Pl^3}{3E\ D^4} = \frac{C\ P}{D} \left(\frac{l}{D}\right)^3 \tag{3.4}$$

Roughly the error caused by deflection can be expressed as a coefficient multiplied by the length/diameter ratio raised to some exponential power. Since no machine is perfect, a constant is necessary to represent the effect of other errors. The final form of straightness and parallelism tolerances can be written as,

$$\text{Tol} = A\ (l/d) ** N + B \tag{3.5}$$

where A, N and B are experimentally determined values.

Flatness error, in surface production, is generally caused by deflection of the tool and machine inaccuracy. machining accuracy is a function of the machine tool repeatability (backlash, etc.), deflection or distortion, and machine spindle/tool alignment. If the machine is rigid, the tool can again be assumed to be a cantilever beam. Given this assumption, the flatness error can be represented as shown in Figure 3.17. In this sytem, flatness error, f, is calculated as

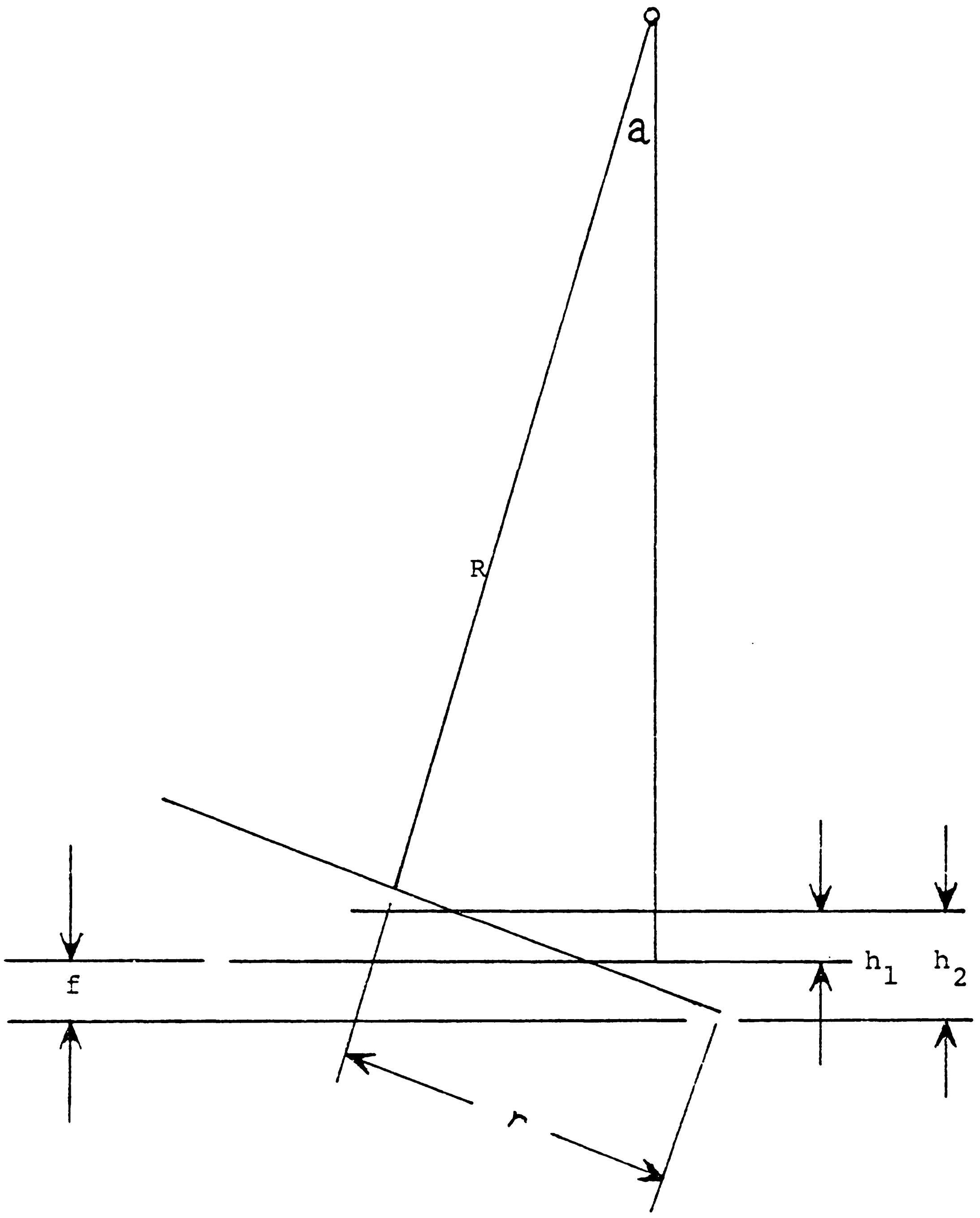


Figure 3.17 Flatness Error Produced by Tool Deflection

$$h = r \sin a \tag{3.6}$$

$$h = R \sin a \tan a \tag{3.7}$$

$$f = h_1 - h_z \tag{3.8}$$

for a small a, sin a ≅ tan a ≅ a

$$f = r\,a - Ra^2 = a(r - Ra) \tag{3.9}$$

For this cantilever system with a discrete bend

$$a \cong F_t \cdot K \cdot R^3 \tag{3.10}$$

and

$$f = F_t r \cdot k \cdot R^{-3} - R\,F_t^2\,k_1^2\,R^{-6} \quad \text{or}$$

$$f = \frac{F_t k_1 (r - \frac{F_t k_1}{R^2})}{R^3} \tag{3.11}$$

The constant, k_1, is determined by the material and its hardness (in this case, k_1 is the force required to remove a cubic measure of a material of this type). This equation can be expanded further to fit the lower limit of flatness error by making a number of assumptions, such as minimum cutting feed, minimum cutting width and depth, etc. This equation, however, becomes quite cumbersome and may (or may not) become inaccurate because of all of the approximations made throughout.

Because of the complexity of the derivation, the dependence on the cutting force variables (width and depth cut, feed, number of cutter teeth), the length and diameter of the cutter, as well as today's shop practices, a general equation of form

$$\text{Flatness} = A\left(\frac{\text{depth of cut}\cdot\text{cutter length}\cdot\text{cutter dia}}{\text{width of cut}}\right)^{n} + B \quad (3.12)$$

was selected to quantify flatness error.

Roundness error of a hole making process is a function of the machine rigidity, material, tool geometry, etc. Little quantifiable information is available. However experimental results can be used to find the boundaries for each process. The same is true for angularity and true position. Constant values or function values can be used to model these characteristics.

Surface finish can be analytically expressed as a function of tool geometry and feed. The theoritical surface finish height for different tool geometries is shown in Figure 3.18. In the figure, h is the maximum height irregularity, and f is the feed. The arithmetic mean value of surface finish, R_a, is defined as the sum of the areas above and below a line which equally divide these two areas (Figure 3.19). For a pointed tool,

$$R_a = \frac{h}{4} = \frac{f}{4(\tan C_s + \cot C_s)} \quad (3.13)$$

For a rounded tool with small feed

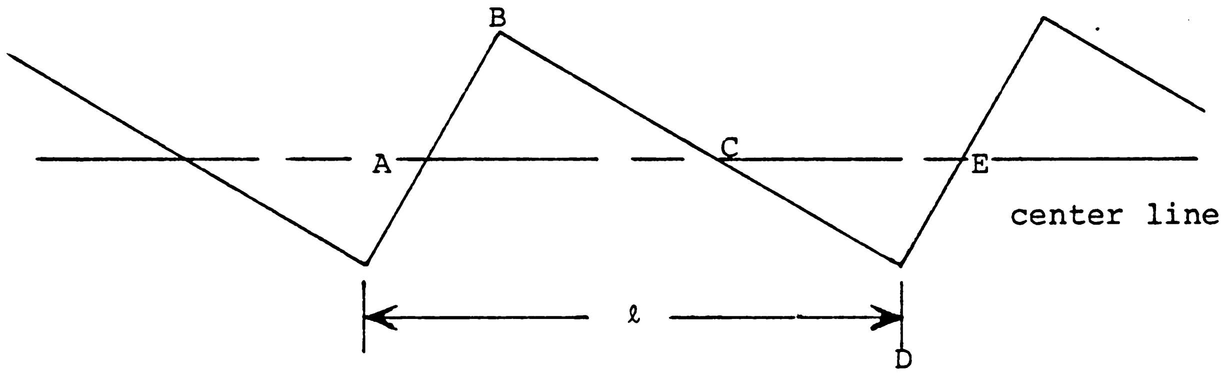
$$R_a = \frac{0.0321 \ f^2}{r} \qquad (3.14)$$

For a rounded tool with a large feed, the above equation also provides a reasonable approximation.

However the above equations only estimate the surface finish under perfect cutting conditions, no consideration was given to other factors which affect cutting. Cook [1964] showed that surface finish is also affected by cutting speed as well as depth of cut. Actually the surface finish value may be more than twice the theoretical value. The surface finish boundary however should be the best attainable surface finish.

3.3.2 Machining Force and Power Requirements

Machining force and power is not a limiting value in process selection, but becomes an important consideration in selecting process parameters (feed, speed and depth of cut). Force and power are functions of process parameters. Using the same tool, machine and workpiece material, in general, the greater the volume of work material removed per unit time, the greater the power required. A reduced feed, speed



area ABC = area CDE

$$R_a = \frac{|\text{area ABC}| + |\text{area CDE}|}{\ell}$$

Figure 3.18 Arithmetic Mean Surface Finish

or depth of cut can reduce both cutting force required and the power consummed. Since force and power are constrained by the machine output, it is necessary to know the power requirements of a cutting process as a function of the process parameters.

In orthogonal cutting, the resultant force, F_r, applied to the chip by the tool lies in a plane normal to the tool cutting edge (Figure 3.20). F_c is the major cutting force and F_t is the thrust force. The cutting force can be expressed roughly as a product of the specific cutting resistance, k_s, and the cross-sectional area of undeformed chip.

$$F_c = b\, a_p\, k \tag{3.15}$$

$$F_t = W\, b\, a_p\, k_s \tag{3.16}$$

where

A : depth of cut

b : width of cut, in turning b = feed

w : coefficient empirically determined by the tool geometry.

The cutting power consumption, p_m, can be calculated as the product of the cutting speed, v, and the cutting force, F_c. Thus

$$P_m = F_c V \tag{3.17}$$

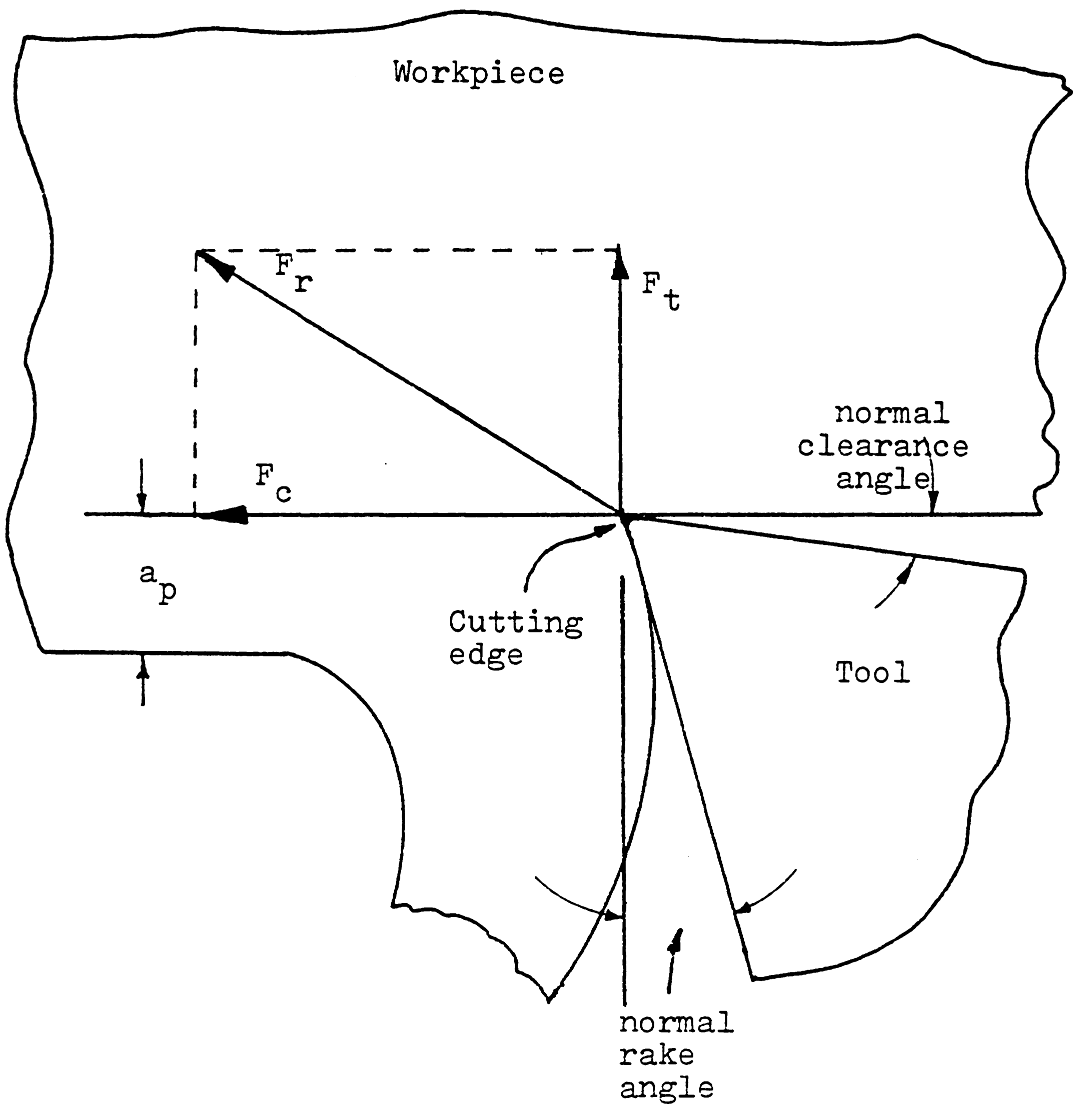


Figure 3.19 <u>Cutting Force Geometry</u>

Table 3.3 Summary of Equations for Cutting Operations
(taken from Coelho 1980)

OPERATION	Machining time t_m	Tool Life t	Cutting Force F_c	Power P_m	Surface Finish R_a
Turning	$\frac{l_w}{f \cdot n_w}$	$K_T \cdot v^{\alpha_T} \cdot f^{\beta_T} \cdot a_p^{\gamma_T}$	$K_F \cdot f^{\beta_F} \cdot a_p^{\gamma_F}$	$\frac{F_c \cdot v}{6120\ \eta_m}$	$\frac{32 \cdot f^2}{r_\epsilon}$
Boring					
Facing	$\frac{D_m}{2 \cdot f \cdot n_w}$				
Parting					
Shaping & Planing	$\frac{b_w}{f \cdot n_r}$				
Drilling	$\frac{l_w}{f \cdot n_t}$	$K_T \cdot v^{\alpha_T} \cdot f^{\beta_T} \cdot a_p^{\gamma_T} \cdot D_t^{\delta_T}$	$K_F \cdot f^{\beta_F} \cdot a_p^{\gamma_F} \cdot D_t^{\delta_F}$ (2)	$\frac{M \cdot n_s}{9.74x10^5\ \eta_m}$	$\frac{64 \cdot 2f^2}{D_t}$
Reaming					
Slab Milling[(a)]	$\frac{l_w + k\sqrt{a(D_t - a)}}{f \cdot n_t}$ (1)	$K_T \cdot v^{\alpha_T} \cdot a_f^{\beta_T} \cdot a^{\gamma_T} \cdot D_t^{\delta_T} \cdot b_w^{\epsilon_T} \cdot z^{\xi_T} \cdot \lambda_s^{\eta_T}$	$K_F \cdot f^{\beta_F} \cdot a_e^{\gamma_F} \cdot D_t^{\delta_F} \cdot b_w \cdot z$		$\frac{64\ 2f^2}{D_t + e}$
Side & Face Milling[(b)]			$K_F \cdot v^{\alpha_F} \cdot a_f^{\beta_F} \cdot a_p^{\gamma_F} \cdot d_t^{\delta_F} \cdot b_w^{\epsilon_F} \cdot z^{\xi_F}$	$\frac{F_c \cdot v}{6120\ \eta_m}$	$K_R \cdot a_f^{1.4}$
Face Milling	$\frac{l_w + D_t}{f \cdot n_t}$				
Broaching	$\frac{l_t}{v}$	$K_T \cdot v^{\alpha_T} \cdot a_f^{\beta_T}$	$K_F \cdot a_f^{\beta_F} \cdot D_m \cdot z_c$ (3)		. . .

(1) $k = \begin{cases} 1, & \text{case (a)} \\ 2, & \text{case (b)} \end{cases}$

(2) same formula holds for torque M

(3) valid for broaching of round holes only

(4) valid for ideal conditions only; however, empirical formulas are available for specific cases in turning operations

(5) eccentricity e will be equal to zero in ideal conditions

$n_w = \frac{v}{\pi D_w}$ $\qquad n_t = \frac{v}{\pi D_t}$

† from Armorego 1968, Boothroyd 1975, and Kaczmarek 1976.

However the above equations for force and power do not consider compound affects of f and a_p to force and power. A more general expression can be found in Kaczmarek [1976]. Equations for cutting force, power, surface finish, tool life and machine time are summarized in Table 3.3 [Coelho 1980]. The equations for F_c contain an empirically determined coefficient, K_F , to account for all possible factors which affect cutting force. K_F must be determined for all tool, work material combinations process types, tool wear conditions, workpiece hardness, tool geometry, and speed.

Nomenclature for Table 3.3

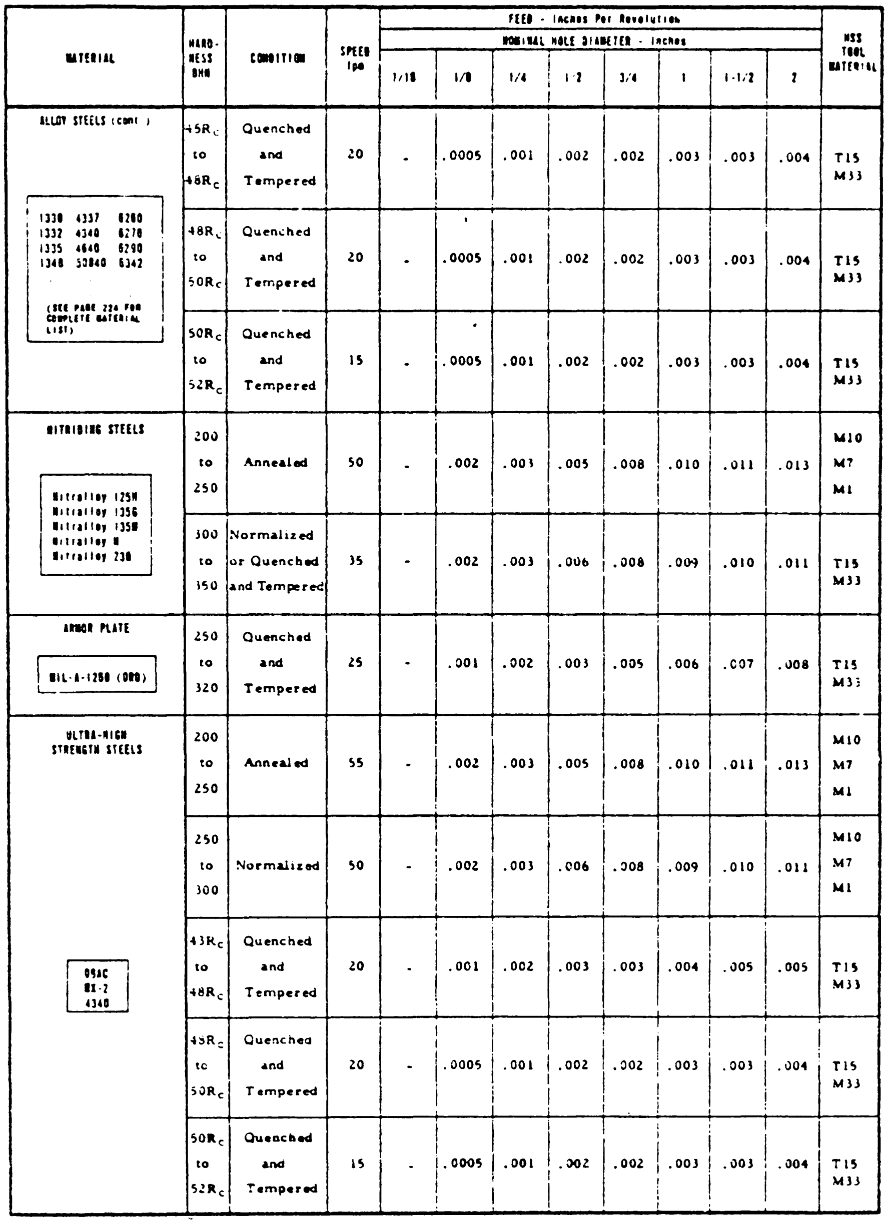
a, a_e, a_f, a_p	: depth of cut
b_w	: width of workpiece
D_m	: diameter of the machined surface
D_t	: diameter of the tool
f	: feed
K_F	: constant for cutting force, tool life and surface roughness, empirical equations, respectively.
l_t	: length of tool or broach
l_w	: length of surface to be machined
n_r	: frequency of reciprocation (strokes/min)
n_t	: tool spindle speed (rpm)
n_w	: rotational frequency of the workpiece (rpm)
r_ε	: tool nose radius
v	: cutting speed
z	: number of teeth on the cutting tool
z_c	: number of teeth cutting simultaneously in a tool
$\alpha_T, \beta_T, \gamma_T, \varepsilon_T, \xi_T, Y_T, n_T, \delta_T$	: cutting speed, feed, depth of cut, tool diameter, machined surface width, number of teeth in the cutting tool and tool cutting edge inclination (L^o) exponents for cutting force, surface

		roughness and tool life equations, respectively.
n_m	:	overall efficiency of the machine tool motor and drive systems
e	:	tool cutting edge inclination

3.3.3 Process Parameters

So far we have discussed process capabilities, cutting force, and power. However, surface finish, force, and power constraints are directly affected by the process parameters -- feed, speed and depth of cut. Therefore process selection becomes an iterative procedure -- first a process is selected and then the machining parameters adjusted to accommodate the system constraints. The selection of the machining parameters however effects the time and cost required to produce a component. These parameters are not arbitrary nor are they constant for different operations.

Process parameters are the basic control variables for a machining process. The earliest study on the economical selection of process parameters was conducted by F. W. Taylor in 1906. As a result of his effort and later the efforts of many others, machining data handbooks [Metcut 1971] and machinability data systems [Parson 1971] have been developed to recommend process parameters for efficient machining. These handbooks contain recommended feeds and speeds for different tooling, work material, tool diameter, depth cut, etc. combinations. The parameters recommended are "good", but not necessary the best or the most appropriate. An example is shown in Figure 3.22. In the figure, feeds and speeds for a 30 to 60 minute tool life for HSS tools (for carbide insert, 1 to 2 hours) are recommended.



MATERIAL	HARD-NESS BHN	CONDITION	SPEED fpm	FEED - Inches Per Revolution								HSS TOOL MATERIAL
				NOMINAL HOLE DIAMETER - Inches								
				1/16	1/8	1/4	1/2	3/4	1	1-1/2	2	
ALLOY STEELS (cont.) 1330 4337 8280 1332 4340 8270 1335 4640 6290 1348 50840 6342 (SEE PAGE 224 FOR COMPLETE MATERIAL LIST)	$45R_c$ to $48R_c$	Quenched and Tempered	20	-	.0005	.001	.002	.002	.003	.003	.004	T15 M33
	$48R_c$ to $50R_c$	Quenched and Tempered	20	-	.0005	.001	.002	.002	.003	.003	.004	T15 M33
	$50R_c$ to $52R_c$	Quenched and Tempered	15	-	.0005	.001	.002	.002	.003	.003	.004	T15 M33
NITRIDING STEELS Nitralloy 125H Nitralloy 135G Nitralloy 135M Nitralloy N Nitralloy 230	200 to 250	Annealed	50	-	.002	.003	.005	.008	.010	.011	.013	M10 M7 M1
	300 to 350	Normalized or Quenched and Tempered	35	-	.002	.003	.006	.008	.009	.010	.011	T15 M33
ARMOR PLATE MIL-A-1258 (ORD)	250 to 320	Quenched and Tempered	25	-	.001	.002	.003	.005	.006	.007	.008	T15 M33
ULTRA-HIGH STRENGTH STEELS	200 to 250	Annealed	55	-	.002	.003	.005	.008	.010	.011	.013	M10 M7 M1
	250 to 300	Normalized	50	-	.002	.003	.006	.008	.009	.010	.011	M10 M7 M1
D6AC BX-2 4340	$43R_c$ to $48R_c$	Quenched and Tempered	20	-	.001	.002	.003	.003	.004	.005	.005	T15 M33
	$48R_c$ to $50R_c$	Quenched and Tempered	20	-	.0005	.001	.002	.002	.003	.003	.004	T15 M33
	$50R_c$ to $52R_c$	Quenched and Tempered	15	-	.0005	.001	.002	.002	.003	.003	.004	T15 M33

Figure 3.20 Example of Drilling Data from the Machining Data Handbook (Metcut)

Several machinability systems are currently marketed that recommend sets of parameters which either optimize machining cost, time or production rate, or simply retrieve data table or calculated values. The FAST system [Parson 1971] is an example of a table value retrieval system. An early General Electric (GE) system used a mathematical model developed by W. W. Gilbert [1950] to calculate one parameter given all others. In Gilbert's model, speed, material, coolant, workpiece surface condition, tool geometry, flank wear, workpiece hardness, tool life, feed, depth of cut, etc. was represented as a mathematical function. A so called machinability computer (analog) using Gilbert's model was marketed by GE in the late 1950's [GE 1957].

More recently (1960 to present), machinability systems have become computer based in order to take advantage of the computational power to include optimization models. For example, a later version of the GE data system contains optimum feed and speed for minimum cost and time using an unconstrained optimization procedure [Parson 1971]. The EXAPT system [Budde 1973] also uses an unconstrained optimization scheme to identify the optimal tool life. Although most commercial machinability data systems use unconstrained optimization, several research papers have appeared which focus on constrained models. This class of problems will be discussed in the following sections.

3.3.4 Process Optimization

Before we begin discussing process optimization, we must first understand the basic tool life equation. A faster metal removal rate will result in both a reduced tool life and reduced machining time. However, whenever a tool has been worn past some practical limits, it must be replaced. Therefore there is a tradeoff between an increased machining rate and machine idle time which results frequent tool changes.

A tool's useful life can be achieved through one of two mechanisms -- erosion (or wear) and breakage (catastrophic failure). Catastrophic tool failure is usually quite unpredictable, and a phenomenon which one tries to minimize; therefore tool life is usually defined as the cut time a new tool undergoes before a certain flank wear is reached. Permissable values of flank wear have been recommended by the International Organization for Standardization (ISO) [1972]. They appear in Table 3.4.

Taylor was the first to develop a generalized tool life equation. In his work he produced an empirical equation which can be written as follows:

$$\frac{V}{V_r} = \left(\frac{t_r}{t}\right)^n \qquad (3.18)$$

where

n = constant

v = cutting speed

t = tool life

V_r = reference cutting speed given tool life t

By rearranging equation (3.18)

$$t = \frac{t_r \, V_r^{\;1/n}}{v^{1/n}}$$

$$= \frac{C}{V^{\alpha}} \qquad (3.19)$$

$\alpha = 1/n$

$C = t_r \, V_r^{\;1/n}$

Later research confirmed that feed and depth of cut also contributed to the tool life. An expanded Taylor tool life equation of the following form resulted

$$t = \frac{\lambda\ C}{V^{\alpha_T} f^{\beta_T}\ a_p^{\gamma_T}} \tag{3.20}$$

where

λ, C:	constants for a specific tool/workpiece combination
$\alpha_T, \beta_T, \gamma_T$:	exponents for a specific tool/workpiece combinations
a_p:	depth of cut

3.3.5 Cost and Time Models

Machining optimization models can be classified as single-pass and multi-pass models. In a single-pass model, we assume only one pass is needed to produce the required geometry. In this case, depth of cut is fixed. In a multi-pass model this assumption is relaxed and a_p also becomes a control variable. Any multi-pass model can be reconstructed into a single-pass model.

Single-pass model

In a single pass model, processing time per component (t_{pr}) can be expressed as the sum of machining time (t_m), material handling time (t_n) and tool change time (t_t).

$$t_{pr} = t_m + t_n + t_t \left(\frac{t_m}{t}\right) \tag{3.21}$$

Equations for t_m and t can be found in Table 3.2. Depth of cut, a_p, in the equations is constant. The production cost per component (C_{pr}) can be written as

$$C_{pr} = \frac{C_b}{N_b} + C_m \left[t_m + t_n + \frac{t_m}{t} \left(t_t + \frac{C_r}{C_m}\right)\right] \qquad (3.22)$$

where

C_b = setup cost for a batch

C_m = total machine and operator rate (including overhead)

C_r = tool, cost (for HSS tool it is the cost of regrinding, for TCT, it is the cost of one insert cutting edge)

N_b = batch size

An optimization model for machining would look as follows

$\min t_{pr}$ (for time) (3.23a)

or

$\min C_{pr}$ (for cost) (3.23b)

Subject to:

(a) Spindle speed constraint

$n\text{-min} < n_w < n\text{-max}$ (for workpiece) (3.24a)

$n'\text{-min} < n < n'\text{-max}$ (for tool) (3.24b)

(b) Feed constraint

$f\text{-min} < f < f\text{-max}$ (3.25)

(c) cutting force constraint

$F_c \quad < \quad F_c\text{-max}$ (3.26)

(d) power constraint

$P_m \quad < P\text{-max}$ (3.27)

(e) surface finish cosntraint

$R_a \quad < R_a\text{-max}$ (3.28)

Equations for n_w, n_t, F_c, P_m and R_a can be found in Table 3.2.

Many solution procedures can be used to solve the above model. Berra and Barash [1967] and later Wysk [1977] used an iterative search procedures which approached optimum. Groover [1976] used a "evolutionary operations" procedure, which is somewhat similar to a Hooke-Jeeves search procedure. Hati and Rao [1976] applied a sequential unconstrained minimization technique (SUMT) [Fiacco and McCormick 1968] in conjunction with the Davidson-Fletcher-Powell (D-F-P) algorithm to solve the problem. Dynamic programming (DP) and other mathematical programming methods have also been used.

3.3.5.1 Multi-pass

Multi-pass models also consider depth of cut as a control variable. Let a_p be the height of material to be removed, and n the number of passes. The time required per component can be written as

$$t_{pr} = t_n + \sum_{i=1}^{n_p} [t_m^i + \frac{t_m^i}{t} t_t] \qquad (3.29)$$

Cost per component:

$$C_{pr} = \frac{C_b}{N_b} + C_m t_n + \sum_{i=1}^{n_p} C_{pr}^i \qquad (3.30)$$

where

$$C_{pr}^i = C_m [t_m^i + \frac{t_m^i}{t} (t_t + \frac{C_r}{C_m})] \qquad (3.31)$$

$$a_t = \sum_{i=1}^{n_p} a_p^i \qquad (3.32)$$

and super script i represents the ith pass.

An additional constraint is also required in the formulation

(f) depth of cut constraint

$$a_p \text{ min} < a_p^i < a_p\text{-max} \qquad (3.33)$$

The additional variable a_p makes the solution procedure more difficult than a single-pass problem. In solving this class of problems Challa and Berra [1976] used a modifed Rosen's gradient search method in their paper. Phillipson and Ravindran [1978] and Subbarao and Jacobs [1978] both used goal programming to deal with the problem. Iwata et al [1977] introduced a DP procedure to solve a multi-stage machining optimization problem. Hayes et al [1981] and Chang et al [1982] transformed certain variables such as depth of cut into the discrete domain. The number of passes can then be obtained iteratively using a DP procedure to optimize the feed and speed. There is no general solution method which can be used for all problems.

Figure 3.23 shows the contours of unit cost and feasible region for an example single-pass problem. In the figure, the bounding constraints as well as the object function are mapped. The bounding constraint for the example is the surface finish. If the surface finish constraint is relaxed, the force constraint becomes binding.

3.3.6 Closing Remarks

There are two types of information which goes into a process planning system: design and process knowledge. The design representation was discussed in Chapter 2. In this chapter, the process knowledge was discussed. Process planning can be said to be a procedure which matches the know-

TOOL : ISO SNMA 120408-P20
HOLDER : ISO PBSNR 2525
MATERIAL : SAE 1045 CD
MACHINE : ENGINE LATHE
feed ~ .05 - 2.5 mm/rev
speed ~ 20 - 1600 rpm
power = 7.5 kW
mach. efficiency = .80

$$t^* = \left(\frac{1}{n} - 1\right)\frac{L}{L+e}\left(t_c + \frac{C_e}{C_m}\right) = 10 \text{ min}$$

$C_m = \$.25/\text{min}$	$t_h = 1.35 \text{ min/pc}$
$C_e = \$.50/\text{edge}$	$t_i = .2 \text{ min/pass}$
$C_b = \$7.20/\text{batch}$	$t_c = 1 \text{ min/edge}$
$D = 100 \text{ mm}$	$L = 250 \text{ mm}$
$e = 50 \text{ mm}$	$N_b = 25$

$p = .1$ $\quad a_p = 3 \text{ mm}$ $\quad K = .3$

$v t^{.2} f^{.3} a_p^{.18} = 220$

$F_c = 250 v^{-.12} f^{.75} a_p \leq 170 \text{ kp}$

$P_e = F_c v / 6120 \leq 6 \text{ kW}$

$R_a = 2.43 \times 10^4 v^{-1.52} f \leq 1.25 \text{ µm}$

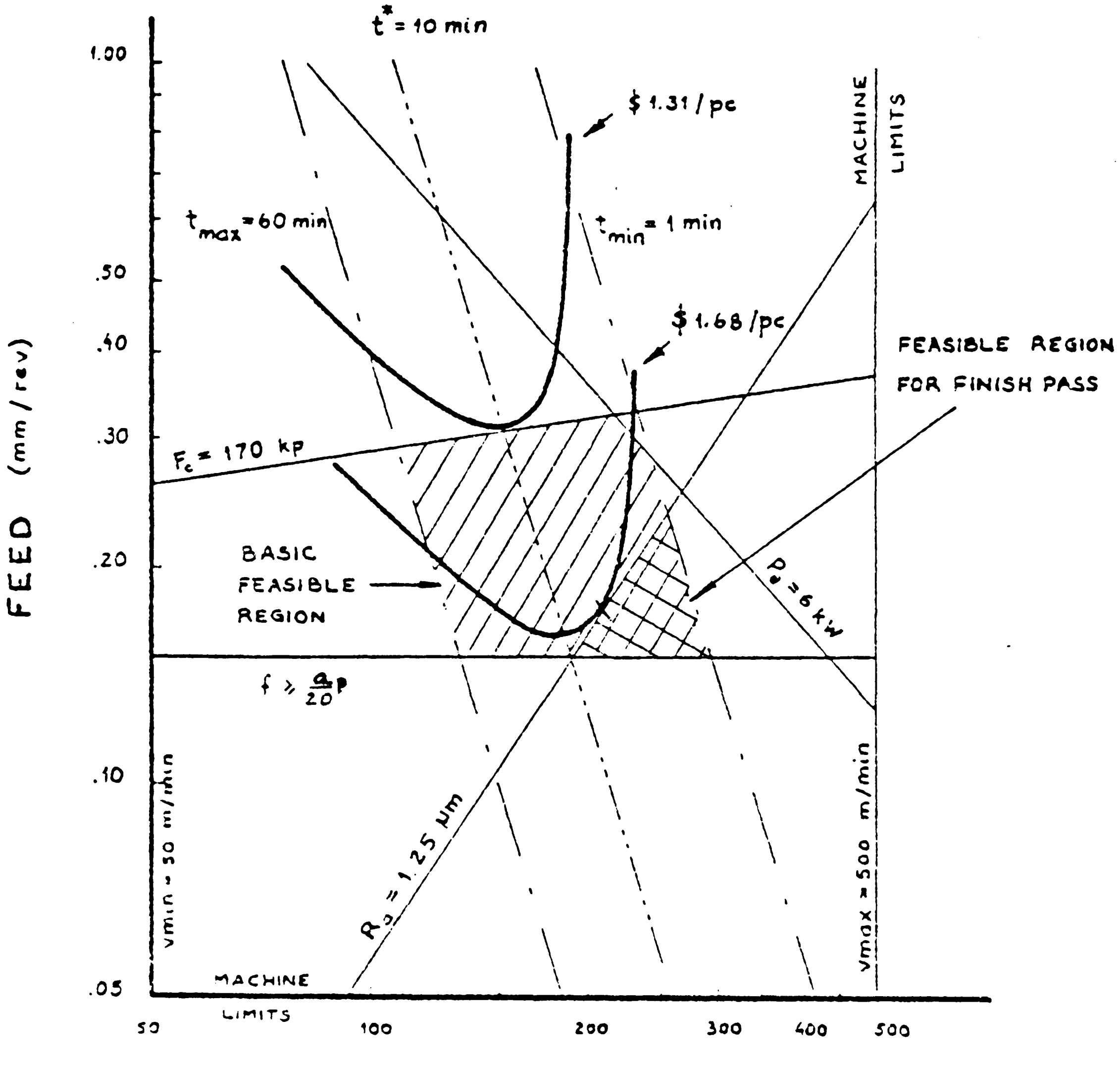


Figure 3.21 Contours of Unit Cost and Feasible Region (taken from Coelho 1980)

ledge of processes with the requirement of the design. Process capability is the knowledge about a process. It is the brain of the process planning system. However, a logical structure is necessary in order to carry-out the matching procedure. This chapter provides both process capability and decision logic. In the next chapter, how to put them into a system will be discussed.

Chapter IV
PROCESS PLANNING

In the previous chapters, design data and production/ process information representations were discussed. They are the input to a process planning system, and the information knowledge base necessary for decision making. The ultimate goal of an automated process planning system is to integrate design and production data into a system which generates usable process plans. An analogy can be drawn for military actions. The goal of the action is like the design. It defines what is to be achieved. In order to accomplish the goal, intellegence reports are gathered to provide knowledge on the goal itself, possible resistance and other factors which may result from the action. No matter how well the goal is defined and how detailed and accurate the intellegence reports are, without a good general staff to prepare an ingeneous operation plan, the action is not likely to be carried out smoothly. The general staff in a process planning system is the control and decision making mechanism. A process plan is prepared based on the design data (the goal) and process knowledge (the intellegence).

As mentioned in Chapter One, there are two approaches used in computer-aided process planning systems: variant and generative. The variant process planning approach uses a data retrieval system to retrieve existing process plans. Systems using this approach are similar in their basic logic. However this is not the case of generative planning systems. There are great variations among generative process planning systems' logic.

In this chapter, we will discuss the structure of process planning systems using variant and generative approaches. The structures discussed represent some existing methods. They do not necessarily cover all the existing methods or possible methods.

4.1 VARIANT PROCESS PLANNING

A variant process planning system uses the similarity among components to retrieve existing process plans. A process plan which can be used by a family of components is called a standard plan. A standard plan is stored permanently in the data base with a family number as its key. There is no limitation to how much detail a standard plan can contain. However, it must contain at least a sequence of fabrication steps or operations. When a standard plan is retrieved, a certain degree of modification is usually necessary in order to use the plan on a new component.

The retrieval method and the logic in variant systems is derived from grouping parts into families. Common manufacturing methods can then be identified for each family. Such common manufacturing methods are represented by standard plans.

The mechanism of standard plan retrieval is based on part families. A family is represented by a family matrix which includes all possible members. The structure of this family matrix will be discussed later.

In general, variant process planning systems have two operational stages: a preparatory stage and a production stage.

4.1.1 The Preparatory Stage

During the preparatory stage, existing components are coded, classified and subsequently grouped into families. A family matrix is also constructed. The process begins by summarizing process plans already prepared for components in the family. Standard plans are then stored in a data base and indexed by family matrices (Figure 4.1). The preparatory stage is a labor intensive process with reports indicating that it can take 18 to 24 man years to complete preparation [PII 1980].

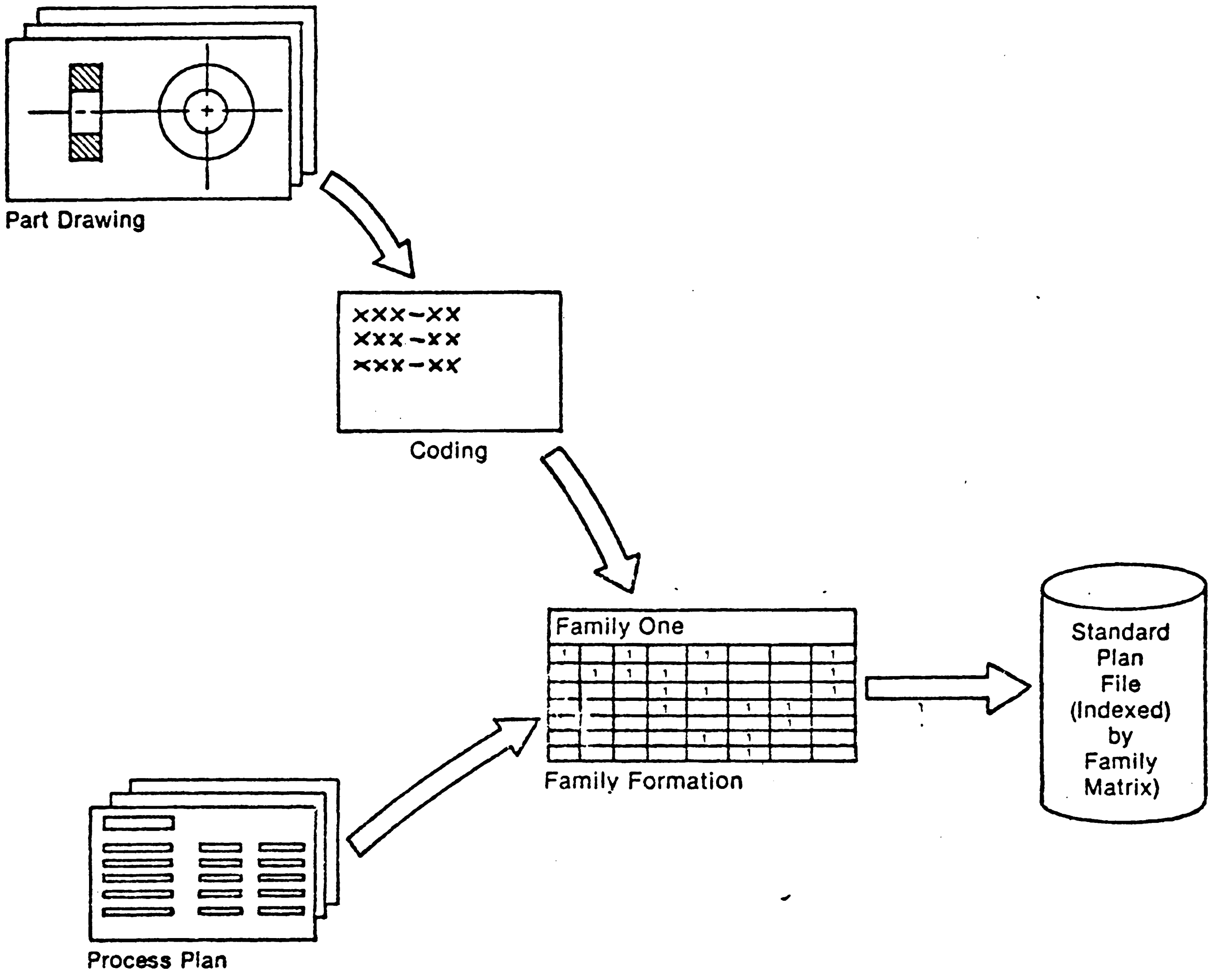


Figure 4.1 Preparatory Stage

4.1.2 The Production Stage

The operation stage occurs when the system is ready for production. New components can now be planned. An incoming component is first coded. The code is then sent to a part family search routine to find the family to which it belongs. The found family number is then used to retrieve a standard plan. The human planner may modify the standard plan in order to satisfy the component design. Figure 4.2 shows the flow of the production stage. Some other functions such as parameter selection, and standard time calculations can also be added to make the system more complete.

4.2 AN EXAMPLE VARIANT PLANNING SYSTEM

An example will be used to show the step by step construction of a variant process planning system. In our example, a simplified coding system will be used. The code table is shown in Figure 4.3. Because it is much simplified, this system will lack detail and will not be appropriate for actual application. However, it is sufficient to represent the principles of coding for process planning. We will call our coding system S-CODE (simple code), and the process planning system VP (Variant Planning).

VP will be used in a machine shop which produces a variety of small components. These components range from sim-

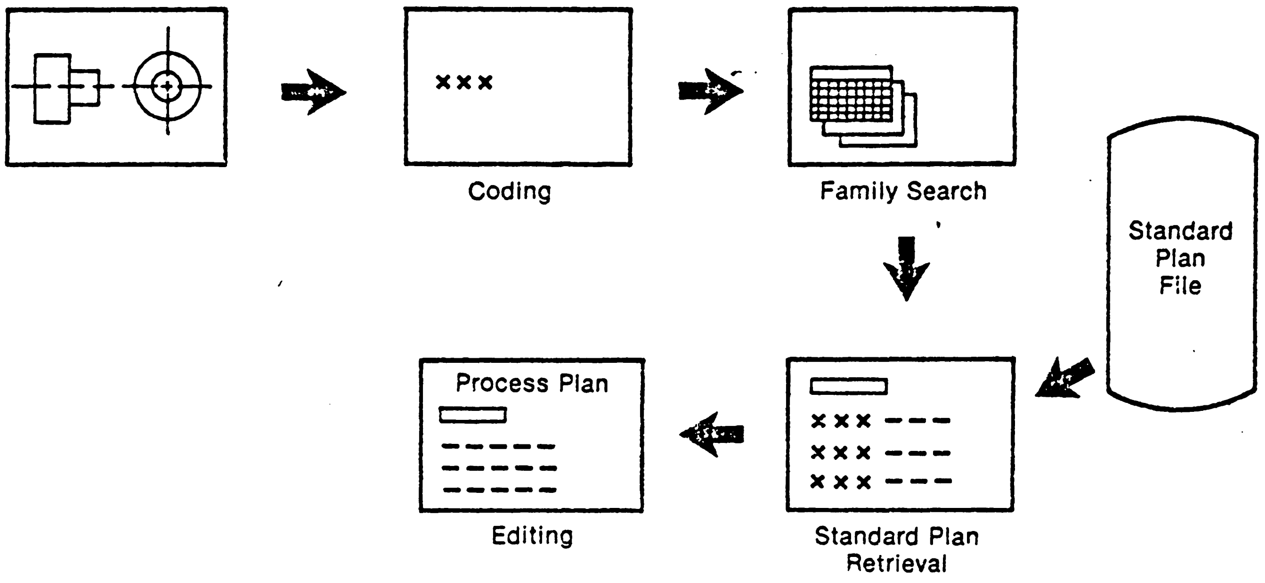


Figure 4.2 Production Stage

		Digit 1	Digit 2		Digit 3		Digit 4
		Primary Shape	Secondary Shape		Auxiliary Shape		Initial Form
0		$\frac{L}{D} \leq 0.05$	No Shape Element		No Shape Element		Round Bar
1	Rotational	$0.5 < \frac{L}{D} < 3$	Steps with Round cross section	No Shape Element	HOLES	No Shape Element	Hexagonal bar
2		$\frac{L}{D} \geq 3$		With Screw thread		With Screw thread	Square bar
3		$\frac{L}{D} \geq 2$ with deviation		With functional groove		With Functional groove	Sheet
4		$\frac{L}{D} > 2$ with deviation	Rotational cross section			Drill with pattern	Plate and Slabs
5	Non-rotational	Flat	Rectangular cross section			Two or more from 2-4	Cost or Forged
6		Long	Rectangular with		Stepped plane surface		Welded Assembly
7		Cubic	Hexagonal bar		Curved Surface		Pre-machined

Figure 4.3 S-CODE

ple shafts to delicate hydraulic pump parts. We will discuss the construction of VP in the followng sequence:

1. family formation,
2. data base structure,
3. search algorithm,
4. plan editing, and
5. process parameter selection.

4.2.1 Family Formation

In a process planning system, family formation is based on production parts or more specifically their manufacturing features. Components requiring similar processes are grouped into the same family. Before any grouping can start, information concerning the design and processing of all the existing components must be collected from existing part and processing files.

Each component's design is coded by using our S-CODE, and the associated process plan is represented in another coded form which is called an operation plan code (OP Code) (Figure 4.4.a). An OP code represents a series of operations on one machine and/or one work station. For example, we can use DRL01 to represent load workpiece on drill press, attach a drill, drill holes, change drill to a reamer, ream holes and unload workpiece from drill. Operations represented by an OP code are called an operation plan. An OP code does not necessarily include all operations required on a machine for a component. It is used to represent a logical group of operations on a machine, so that a process plan can be represented in a much more concise manner. Such a representation is called an OP code sequence (Figure 4.4b) Since we do not want to lose the detail of a process plan, operation plans and OP Code sequences must be stored properly.

The basic premise of an OP Code is to simplify the representation of process plans. A simplified process plan can be stored and retrieved by a computer easily when represented in this way. It also can contribute to the family formation process. For example, we have a total of 24 components in our mini-shop. After coding them, we can obtain a summary in table form as shown in Figure 4.5. There are two methods that we can use to group parts into families: 1) by observation, or 2) by computerized methods such as production flow analysis. It is obvious that when we have many components, the first method (observation) will be difficult to use.

Production flow analysis (PFA) was first introduced by Professor J. L. Burbridge tool to solve the family formation problem for manufacturing cell design [Burbridge 1971,

	Operation Code	Operation Plan
01	SAW 01	cut to size
02	LATHE 02	face end
		center drill
		drill
		ream
		bore
		turn straight
		turn groove
		chamfer
		cutoff
		face
		chamfer
03	GRIND 05	grind
04	INSP 06	inspect dimensation
		inspect finish

a. Operation plan code (OP code) and operation plan

01	SAW 01
02	LATHE 02
03	GRIND 05
04	INSP 06

b. OP Code sequence

Figure 4.4 Operation Plan, OP code and OP code Sequence

Component	Code	Processes	(OP CODE SEQUENCE)		
A-112	1110	SAW01,	LATHE02,	GRIND05,	INSP06
A-115	6514	MILL02,	DRL01,	INSP03	
A-120	2110	SAW01,	LATHE02,	GRIND05,	INSP06
A-123	2010	SAW01,	LATHE01,	INSP06	
A-131	2110	SAW01,	LATHE02,	INSP06	
A-212	7605	MILL05,	INSP03		
A-230	6604	MILL05,	INSP03		
A-432	2120	SAW01,	LATHE02,	INSP06	
A-451	2130	SAW01,	LATHE02,	INSP06	
A-510	7654	MILL05,	DRL01,	GRIND06,	INSP06
A-511					
A-511					
A-512					
A-550					
A-556					
B-105					
B-107					
B-108					
B-109					
B-115					
B-116					
B-117					
B-118					
B-119					
B-120					

Figure 4.5 A Partial Component Process Summary

	A-112	A-115	A-120	A-123	A-131	A-212	A-230	A-432	A-451	A-510
SAW01	/		/	/	/			/	/	
LATHE01				/						
LATHE02	/		/		/			/	/	
DRL01		/								/
MILL02		/								
MILL05						/	/			/
GRIND05										
GRIND06	/		/							/
INSP03		/				/	/			
INSP06	/		/	/	/			/	/	/

Figure 4.6 PFA Matrix

1975]. Many researchers have subsequently developed algorithms to solve the problem. In PFA, a large matrix is constructed. Each row represents an OP code, and each column in the matrix represents a component (Figure 4.6). We can define the matrix as $M_{i,j}$, where i designates OP codes, and j designates components ($M_{i,j}$ =1 if component j has OP code i, otherwise $M_{i,j}$ = 0). The objective of PFA is to bring together those components which need the same or a similar set of OP codes in clusters.

King [1979] presented a rank order cluster algorithm which is quite simple in nature. We will use his method to show how component families can be determined in our shop. King's algorithm can be stated as:

Step 1. For $\forall j$ calculate the total weight of column w_j.

$$W_j = \Sigma_{ij} 2^i M_{i,j}$$

Step 2. If w_j is in an ascending order, go to step 3. Otherwise, rearrange the columns to make w_j fall in an ascending order.

Step 3. For $\forall i$, calculate the total weight of row w_i.

$$W_i = \Sigma_j 2^j M_{i,j}$$

Step 4. If w_i is in decending order, stop. Otherwise, rearrange the rows to make w_i fall in an ascending order. Go to step 1.

Figure 4.7 shows the procedure of rearranging the PFA matrix in Figure 4.6. Note that the last two rows in Figure 4.6. were not used in Figure 4.7 in order to simplify the example.

After we obtain the final matrix, we can determine that components A123, A120, A131, A432, A451 and A112 form a family which needs SAW01, LATHE01, LATHE02 and GRIND05. A115, A212, A230 and A510 form the second family.

This family must then be represented in a manner which is consistent with the S-Code. The representation used is called a part family matrix. A part family matrix is a binary matrix similar to a PFA matrix. We can use P^1_{ij} to represent a part family matrix for 1. i = 1, ..., I, where I is the number of possible values in each code position, and j = 1, ..., J, where J is the code length. In our S-Code, i = 8 and j = 4. $P^1_{ij} = 1$ implies that code position j is allowed to have a value i.

A part family matrix can be constructed in the following manner:

Let

Rank	W_j	M05	D01	G06	L02	S01	G05	M02	L01	
1	18					✓			✓	A123
2	48					✓				A120
3	48					✓				A131
4	48					✓				A432
5	48					✓				A451
7	132		✓					✓		A115
8	256	✓								A212
9	256	✓								A230
6	56				✓	✓	✓			A112
10	448	✓	✓	✓						A510
		256	128	64	32	16	8	4	2	

	G06	G05	M05	M02	D01	L02	L01	S01	
2							✓	✓	A123
4						✓		✓	A120
8						✓		✓	A131
16						✓		✓	A432
32						✓		✓	A451
64				✓	✓				A115
128			✓						A212
256			✓						A230
512		✓				✓		✓	A112
1024	✓		✓		✓				A510
W_i	1024	512	1408	64	1088	602	2	574	
Rank	6	3	8	2	7	5	1	4	

Rank	W_j	G06	G05	M05	M02	D01	L02	L01	S01	
9	138		✓				✓		✓	A112
6	48				✓	✓				A115
2	10						✓		✓	A120
1	6							✓	✓	A123
3	10						✓		✓	A131
7	64			✓						A212
8	64			✓						A230
4	10						✓		✓	A432
5	10						✓		✓	A451
10	336	✓		✓		✓				A510
		256	128	64	32	16	8	4	2	

Figure 4.7 Rank Order Clustering Algorithm (to continue)

Stop

W_j	M05	D01	G06	M02	S01	L02	G05	L01	
18								⟍	A123
24					⟍	⟍			A120
24					⟍	⟍			A131
24					⟍	⟍			A432
24					⟍	⟍			A451
26					⟍	⟍	⟍		A112
160		⟍		⟍					A115
256	⟍								A212
256	⟍								A230
510	⟍	⟍	⟍						A510
	256	128	64	32	16	8	4	2	

	M05	D01	G06	L02	S01	G05	M02	L01	
2					⟍			⟍	A123
4				⟍	⟍				A120
8				⟍	⟍				A131
16				⟍	⟍				A432
32				⟍	⟍				A451
64				⟍	⟍	⟍			A112
128		⟍					⟍		A115
256	⟍								A212
512	⟍								A230
1024	⟍	⟍	⟍						A510
	1792	1152	1024	124	126	64	128	2	W_j
	8	7	6	3	4	2	5	1	Rank

Figure 4.7 (continue)

$C_j^{k,1}$ be the value of code position j for component k in family 1. $k = 1,\ldots,k$

```
Step 1.  For $k = 1$ to $k$, do Step 2, end stop.
Step 2.  For $j = 1$ to $J$ do Step 3, end.
Step 3.  $i = C_j^{k,1}$
         $p_{ij}^1 = 1$
```

Using the above procedure, we can obtain a part family matrix for family one (Figure 4.8). Thus far, we have a complete set of OP code sequences, OP plans and a family matrix. The next step is to store them in a computer interpretable manner so that we can use this information later for new components.

4.2.2 Data base Structure

The VP system contains only a small amount of information as opposed to an industrial application where thousands of components and process plans need to be stored and retrieved. Because of the large amount of information, data base systems play an important role in variant process planning. A data base is no more than a group of cross referenced data files. The data base contains all the necessary information for an application, and can be accessed by sev-

Position

$$P^{1}_{i,j} = \begin{array}{c|c|c|c|c|}
 & 1 & 2 & 3 & 4 \\ \hline
0 & & | & & | \\ \hline
1 & & | & | & \\ \hline
2 & | & & | & \\ \hline
3 & & & | & \\ \hline
4 & & & & \\ \hline
5 & & & & \\ \hline
6 & & & & \\ \hline
7 & & & & \\ \hline
\end{array}$$

k	Part	Code	
1	A123	2010	$C_{1}^{1,1} = 2$
2	A120	2110	$C_{2}^{1,1} = 0$
3	A131	2110	$C_{3}^{1,1} = 1$
4	A432	2120	$C_{4}^{1,1} =$
5	A451	2130	
6	A112	1110	

Figure 4.8 Part Family Matrix

eral different programs for specific applications. There are three approaches to construct a data base; hierarchical, network and relational. Although the concept and structure for these approaches is very different, they all can serve the same purpose.

For commercial programming, there are several available data base management systems such as CODASYL, MARKIV, SPIRES, etc. These systems are high-level languages for data base construction and manipulation. Of course a data base can always be written using procedural languages such as COBOL, FORTRAN, PL/I, etc. No matter what approach and language is used, the basic structure of the data base keeps the same form.

The hierachical approach will be used to construct the data base in the design of the VP system. Figure 4.9 shows the hierarchy of the data. Each family is indexed by its family number. A standard plan is associated with each family and is represented by an OP code sequence. In the sequence each OP code has an associated OP plan stored on a lower level. Data for each level is stored in a file; therefore, VP requires three files: 1) a family matrix file, 2) a standard plan file, and 3) an OP plan file.

The family name and family matrix are stored as a record in the data base. A forward pointer is used is needed to link the next record and another pointers is used locate the associated standard plan in the standard plan file. We can assign two words for the family name, two words for pointers and I x J (8 x 4) words for the family matrix. A total of 36 (8x4+4) words are required for each record. Figure 4.10 illustrates the structure of a family matrix file.

Because the OP code sequence has variable length, the structure for a standard plan file must include variable record lengths. In the file, a directory is used to locate OP code sequences. The rest of the file is divided into segments which can store up to five OP codes. The last word is used to indicate the continuation of the sequence. Expansion, deletion and modification of a record is made possible by pointers in the directory and the continuation flag.

The OP plan file has a structure similar to the standard plan file except it keeps up link pointers to the standard plan file. Since records in the standard plan file have a one to many relationship with those in the OP plan file, it is necessary to keep "where it comes from pointers" in the OP plan file. This organization makes file maintenance easier.

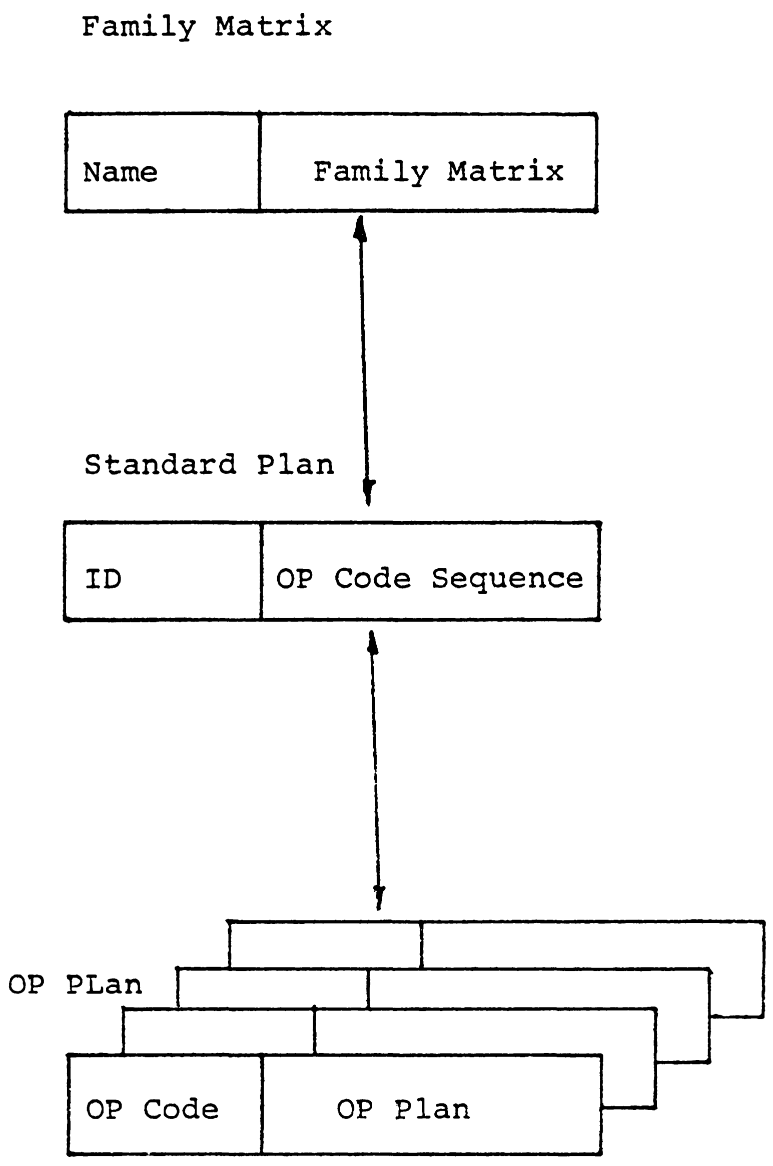
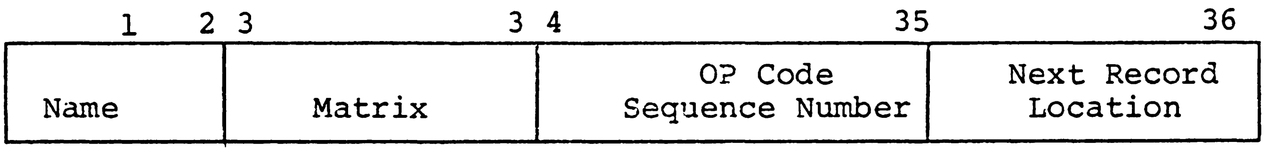
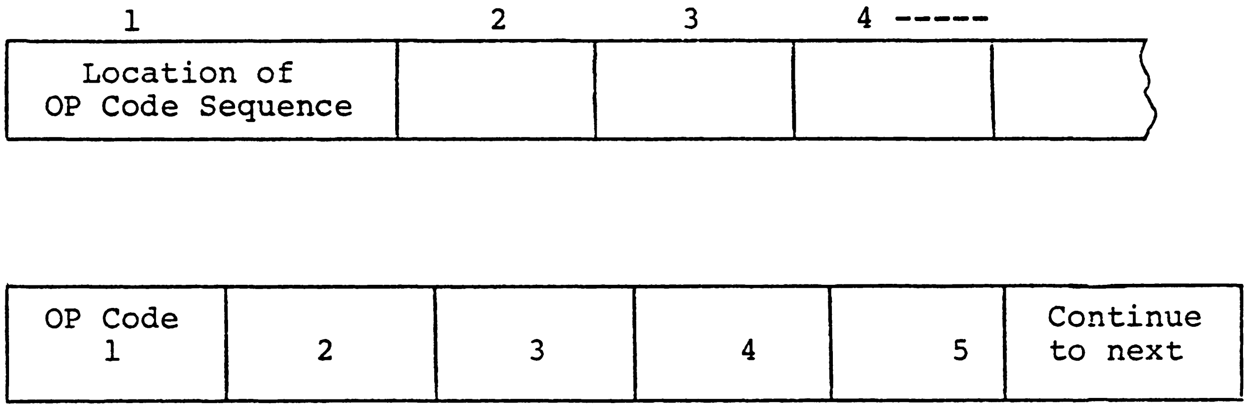


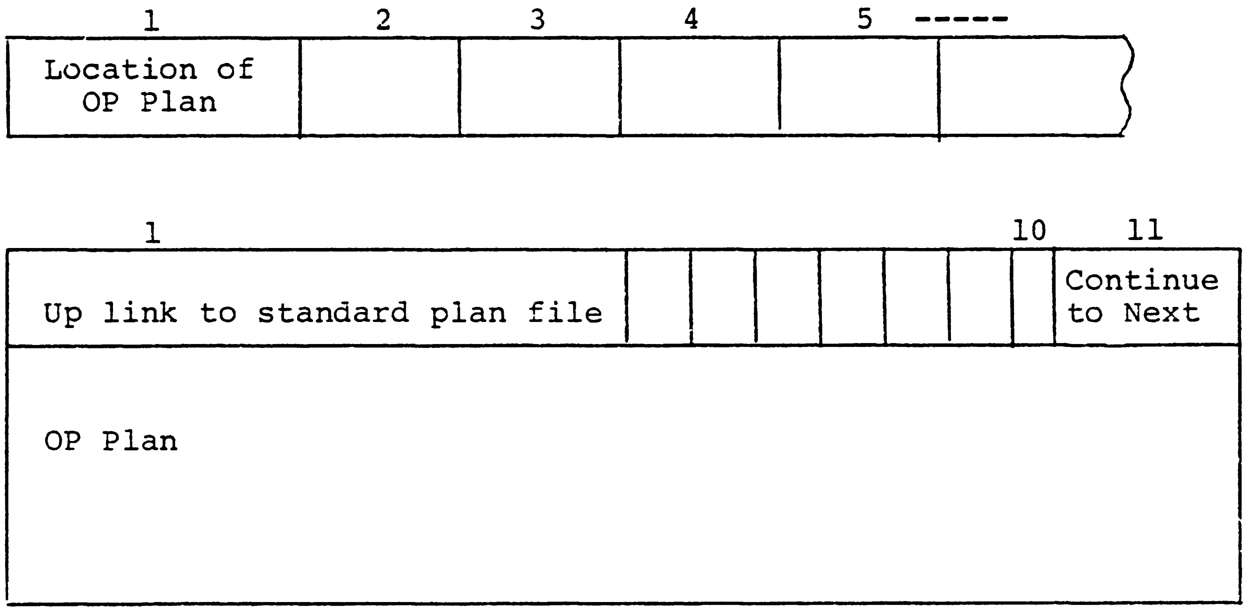
Figure 4.9 Data Hierarchy



A. Family Matrix Record



B. Standard Plan File



C. OP Plan File

Figure 4.10 <u>Data Records Contain</u>

Figure 4.11 shows the overall structure of the VP data base. The storage of family one and two is shown in Figure 4.12.

4.2.3 Search Procedure

Once the preparatory stage has been completed, the variant planning system is ready for production. The spirit of a variant system is to retrieve process plans for similar components. The search for a process plan is based on the search of a part family to which the component belongs. When the part family is found, the associated standard plan can be easily retrieved.

A family matrix search can be seen as the matching of family matrix with a given code. Family matrices can be considered as masks. Whenever a code can pass through a mask successfully, the family is found. The search procedure can be described as follows:

Let C_j be a value of code position j for the given component.

P_n^1 is a pointer for family matrix 1, which links to the next family matrix.

P_s^1 is a pointer for family matrix 1, which links to the directory of the standard plan file.

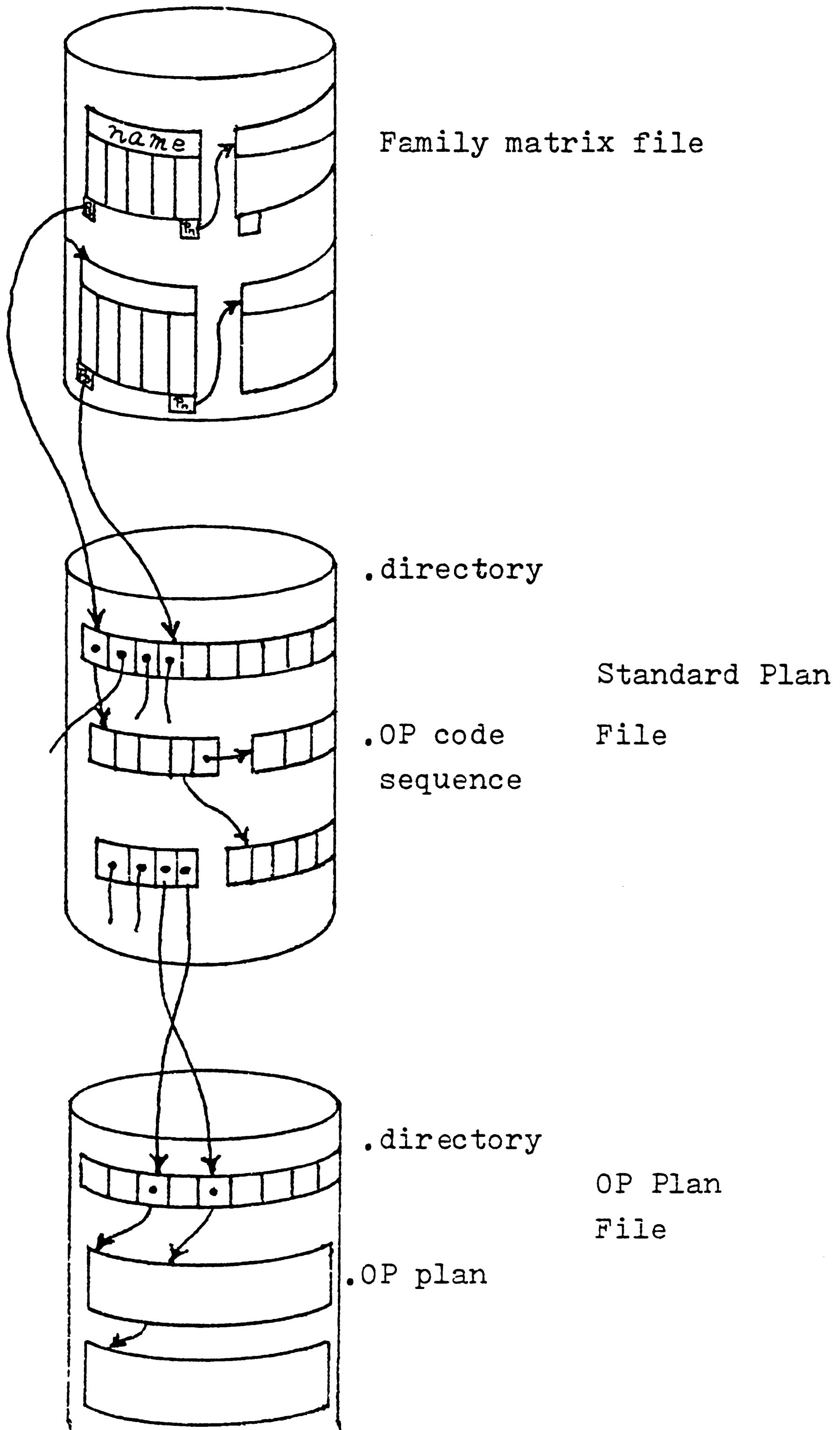


Figure 4.11 <u>Data Base Structure</u>

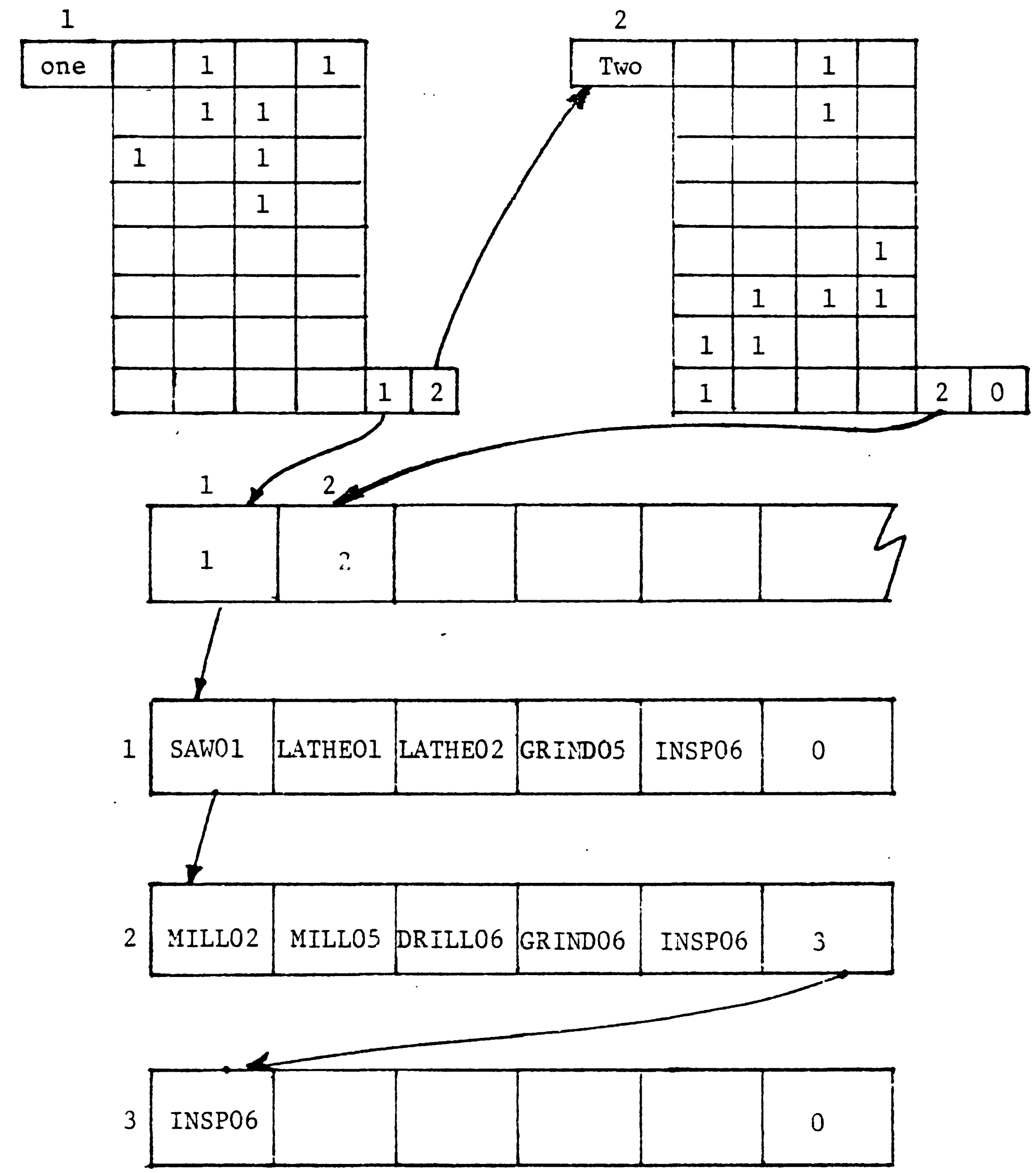


Figure 4.12 VP System Data

P^{l}_{ij} is defined the same as in section 4.1.2.

The following algorithm can be used to find a standard plan.

```
Step 1.  For all $l$, do step 2.  End stop.

Step 2.  For $j = 1$ to $J$ do step 3, end go to step 5

Step 3.  $i = C_j$
            If $P^{l}_{ij} \neq 0$ end step, otherwise next $j$

Step 5.  Standard plan found, $p^{l}_{s}$ is the pointer to
stand
            plan file terminate.
```

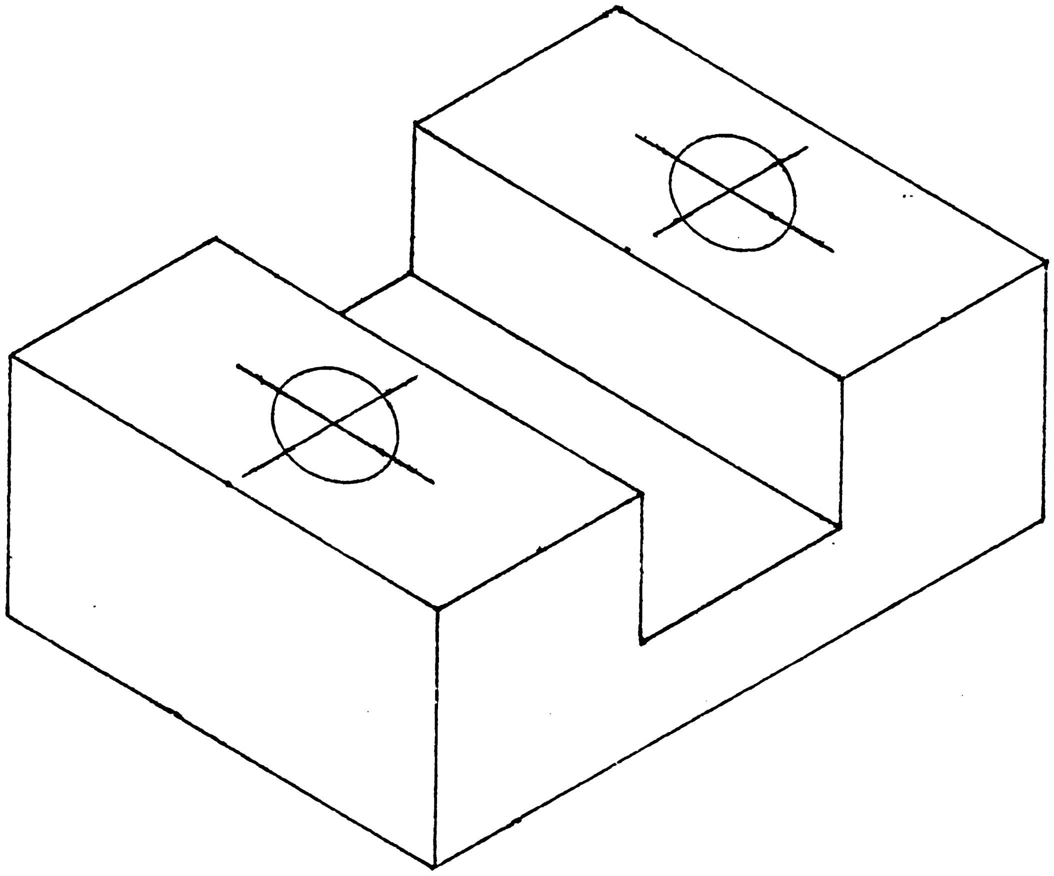
In some commercial systems, a so called matrix search is also used. In a matrix search, c_j is allowed to be a range of values instead of a single value. The above algorithm can be modified to perform a matrix search as follows:

```
Let $C^{*}_{j}$ be a range  $C^{*}_{j} = C_j \pm \epsilon$
Step 3  $\sum_{i \epsilon C^{*}_{j}} P^{l}_{ij} \neq 0$ end step, otherwise next $j$
```

We can demonstrate this search procedure by the use of an example. For the example, the mounting bracket shown in Figure 4.13 will be planned using the VP system. Based on the S-code table in Figure 4.3, a code (6514) can be developed for the component (To make VP even more effective, we could develop a interactive coding system for component coding). This interactive system could eliminate the use of the manual table. (C_1 = 6, C_2 = 5, C_3 = 1 and C_4 = 4). Referring to Figure 4.12, we can start the family search. Figure 4.14 shows the step-by-step search procedure. The result of the search is a standard plan represented by an OP code sequence. It is obvious that some modification is needed before it can be used. OP plans can be retrieved and substituted for OP codes in the OP code sequence. Manual modification of the final process plan is also needed.

4.2.4 Plan Editing and Parameter Selection

Before a process plan can be issued to the shop, some modification of the standard plan is necessary, and process parameters must be added to the plan. There are two types of the plan editing: One is the editing of standard plan itself in the data base, and the other is the editing of the plan for the component. Editing a standard plan implies that a permanent change in the stored plan be made. This editing must be handled very cautiously, because the effectiveness of a standard plan affects the process plans generated for the entire family of components. Aside from the technical considerations of file maintenance, the structure



S-Code : 6514

Figure 4.13 Component to be Planned

C = 6514

```
Step 1   $\ell = 1$
     2   $j = 1$
     3   $i = C_1 = 6,\ P^1_{6,1} = 0$ next
     4   $\ell = P^1_n = 2$,  go to 2
     2   $j = 1$
     3   $i = C_1 = 6,\ P^2_{6,1} = 1$  end step
     2   $j = 2$
     3   $i = C_2 = 5,\ P^2_{5,2} = 1$  end step
     2   $j = 3$
     3   $i = C_3 = 1,\ P^2_{1,3} = 1$  end step
     2   $j = 4$
     3   $i = C_4 = 4,\ P^2_{4,3} = 1$  end step
     2   $j = 5$   go to step 5
     5   plan found
         $P^2_s = 2$
```

	OP Code
1	Mill 02
2	Mill 05
3	Drill 01
4	Grind 06
5	Insp 03
6	Insp 06

Figure 4.14 Search Procedure

of the data base must be flexible enough to accept expansion additions and deletions of data records. As a result, the pointer system in VP may prove to be efficient.

Editing a process plan for a component requires the same expertise as editing a standard plan. However, it is a temporary change; and, therefore does not affect any other component in the family. During the editing process, the standard plan needs to be modified to suit the specific needs of the given component. Some operations or entire OP records need to be removed; others must be changed. Additional operations may also be required to satisfy the design. A text editor is usually used at this stage.

A complete process plan includes not only operations but also process parameters. As discussed in Chapter 3, process parameters can be found in machining data handbooks or can be calculated using optimization techniques. The first approach is easier and more appropriate for the VP system.

Figure 4.15 shows the structure for the parameter file. Data in the file are linked so that we can go through the tree to find the feed and speed for an operation. For example MILL02 for the mounting bracket in Figure 4.13 uses a face milling process. The worpiece material is cast iron (BHN = 180). The depth of cut for roughing is 0.25", and the cutter used is 0.5" in diameter. We can then locate using this information a velocity (V = 55 SFM) and a feed (f = 0.001 ipt).

This parameter file can be integrated into VP to automatically select process parameters. Information such as depth of cut and cutter diameter can be retrieved directly from the OP plan for each operation. The same approach is also appropriate for standard time selection.

With this example we have completed our discussion of the variant process planning approach. A review of the current systems as well as discussion on pros and cons of this approach will be represented in the next chapter.

4.3 GENERATIVE APPROACH

Generative process planning is a second type of computer-aided process planning. It can be concisely defined as a system which synthesizes process information in order to create a process plan for a new component automatically. In a generative planning system, process plans are created from information available in a manufacturing data base without human intervention. Upon recieving the design model, the system can generate the required operations and operations sequence for the component. Knowledge of manufacturing must be captured and encoded into efficient software. By applying decision logic, a process planner's decision-

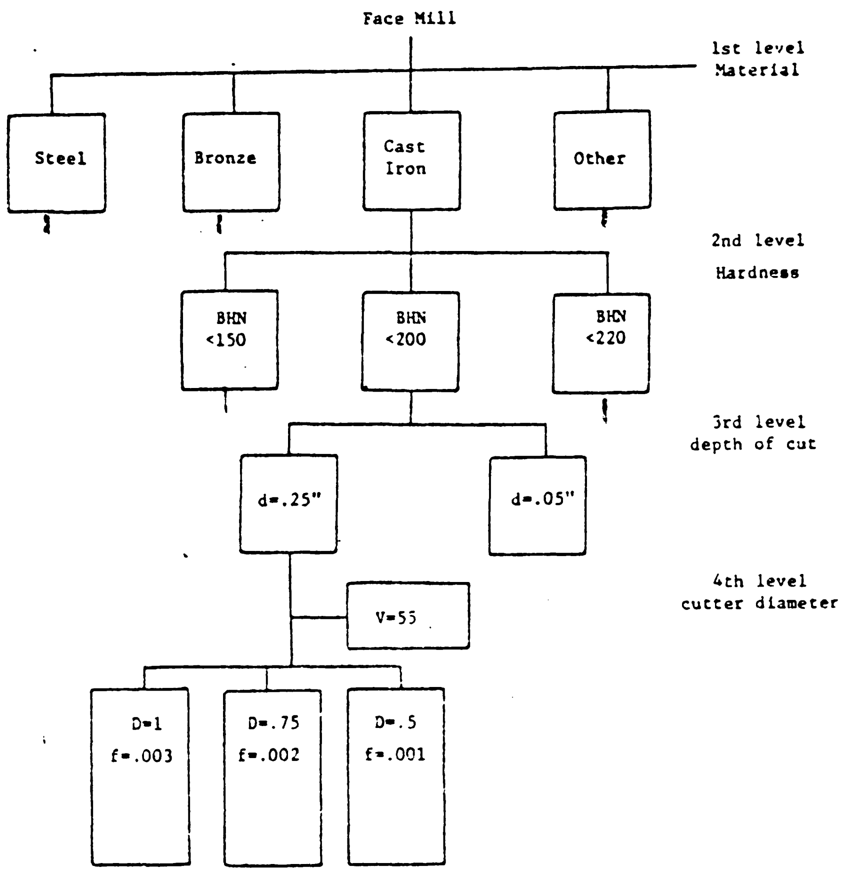


Figure 4.15 Process Parameter File

making process can be imitated. Other planning functions, such as machine selection tool selection, process optimization, etc. can also be automated using generative planning techniques.

The generative planning approach has the following advantages:

1. It can generate consistent process plans rapidly.
2. New components can be planned as easily as existing components.
3. It can potentially be interfaced with an automated manufacturing facility to provide detailed and up-to-date control information.

Decisions on process selection, process sequencing, etc. are all made by the system. However transforming component data and decision rules into a computer readable format is still a major obsticle to be overcome before generative planning systems become operational. Successful implementation of this approach requires the following key developments:

1. The logic of process planning must be identifed and captured.
2. The part to be produced must be clearly and precisely defined in a computer-compatible format (e.g. 3-D model).

3. The captured logic of process planning and the part description data must be incorporated into a unified manufacturing data base.

Today, the term "generative process planning" is relaxed from the definition given above to a less complete system. Sytems with built-in decision logic are often called generative process planning systems. The decision logic is usually an ability to check some conditional requirements of the component and select a process. Some systems have decision logic to select several "canned" process plan fragments and combine them into a single process plan. However, no matter what kind of decision logic is used and how extensively it is used, the system is usually categorized as a generative system.

There are several generative process planning systems (based on the relaxed definition) used in industry (e.g. AUTOPLAN of Metcut (Vogel 1979, Vogel, et. al. 1981), CPPP of United Technology (Dunn, et. al. 1978, Mann, et. al. 1977).

Ideally, a generative process planning system is a turn-key system with all the decision logic contained in the software. It possesses all the necessary information for process planning; therefore, no preparatory stage is required. This however is not always the case. In order to generate a more usable process planning system, variables such as process limitations and capabilities, process costs, etc. must be defined prior to the production stage. Systems, like CPPP [Kotler 1980], require user-supplied decision logic (process models) for each component family. A wide range of methods have been and can be used for generative process planning. In the following sections, we will discuss a few methods which have been successfully implemented.

4.3.1 Forward and Backward Planning

In variant process planning, process plans are retrieved from a data base. A direction for the planning procedure does not exist since plans are simply paired to a code. However, in generative process planning, when process plans are generated, the system must define an initial state in order to reach the final state (goal). The path taken (initial - final or final - initial) represents the sequence of processes. For example, the initial state is the raw material (workpiece), and the final state is the component design. If a planner works on modifying the raw workpiece until it takes on the final design qualities, then the type of component is shown in Figure 4.16. The raw workpiece is a 6.0 x 3.0 x 2.5 block. Using forward planning, we start from the top surface, S_1 . A milling process is used to create the final dimension for S_1. After S_1 has been planned, S_2 can be planned. For S_2, a chamfering process is first selected, and then the hole is drilled and tapped. The progression begins from the raw material and proceeds to the finished requirements.

Backward planning uses a reversed procedure. Assuming that we have a finished component, the goal is to fill it to the unmachined workpiece shape. Each machining process is considered a filling process. A drilling process can fill a hole. A reaming process can fill a thin wall (cylinder), etc. When applied to the example component, the bottom most surface is planned first. A tapping process is selected (this reduces the threaded surface to a smaller diameter hole with a rough surface finish). Drilling is then selected to fill the hole, and so for and so on, until we finally obtain one block.

Forward and backward planning may seem similar; however, they affect the programming of the system significantly. Planning each process can be characterized by a pre-condition of the surface to be machined and a post-condition of the machining (its results). For forward planning, we must know the successor surface before we select a process. Because the post condition of the first process becomes the

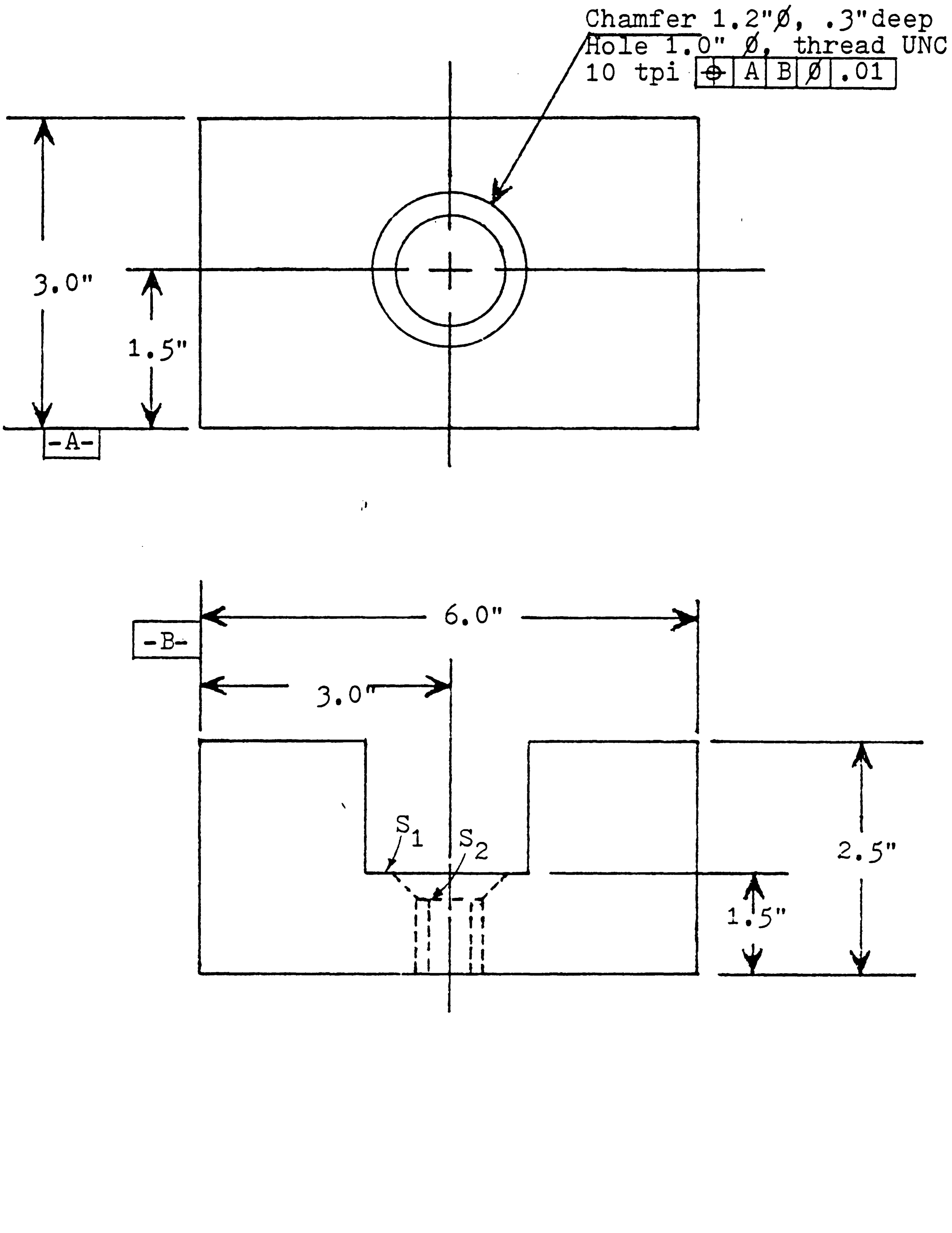


Figure 4.16 Component to be Planned

precondition for the second process. For example, when we selected drilling, for the threaded hole, we knew that the thread was going to be cut. Therefore, we rough drilled the hole using a smaller drill. Otherwise, we might have choosen a larger drill, then no thread can be produced. Backward planning eliminates this conditioning problem since it begins with the final surfaces form and processes are selected to satisfy the initial requirements. The transient surface (intermediate surface) produced by a filling process is the worst precondition a machining process can accept, i.e., depth of cut left for finish milling, etc. Any filling process which can satisfy the transient surface can be selected as the sucessor process.

In forward planning, the objective surface must always be maintained even though several operations must be taken to gaurantee the result. On the other hand, backward planning starts with the final requirements (which helps to select the predessor process) and searches for the initial condition or something less accurate (which is easy to satisfy).

4.3.2 Input Format

The input format of a process planning system affects the ease with which a system can be used, and the capability of the system. A system using a very long special description language as its input is more difficult to use. The translation from the original design (either an enginering drawing or a CAD model) to a specific input format may be tedious and difficult to automate. In this case, it is probably easier and faster to plan a component manually than to prepare the input. However such input can provide more complete information about a component, and more planning functions can be accomplished using the input. (This does not imply that a system using a long and special descriptive language always provides more planning functions).

Many different input formats have been used in process planning systems. Although no/few systems use the same input format, we can categorize them into the following classes.

CODE

As discussed in the variant approach, GT codes can be used as input for variant process planning systems. Some generative systems such as APPAS [Wysk 1977] GENPLAN [Tulkoff 1981], etc. also use part codes as input. Codes used in generative systems are more detailed, and sometimes mix code digits with explicitly defined parameter values (refer to Chapter 2.3). Since code is concise, it is easy to manipulate. When process capabilities are represented by a code, a simple search through the process capability to match the component code will return the desired process. In order to determine the process sequence, a code for the entire component is appropriate, because it provides global information. However when processing detail is required, surface coding is unavoidable. A surface code normally describes the surface shape, dimensions, surface finish and tolerances (both dimensional and geometric) rather than the entire part.

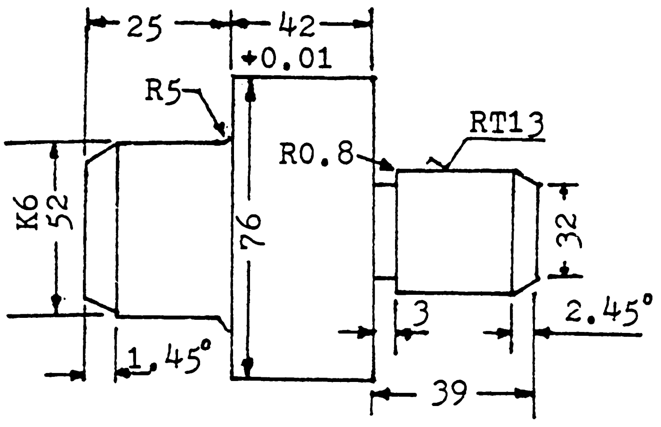
Although surface code is easy to manipulate and store, it is difficult to generate this code automatically through software. A human interface between design and process planning facilitates the translation of information from one system to another.

DESCRIPTION LANGUAGE

Specially designed part description languages can provide detailed information for process planning systems. A language can be designed to provide all of the information required for the necessary functions of a process planning system. The format can be designed such that functions can easily accomplish their task from the provided information.

AUTAP system [Eversheim et. al. 1980] uses a language similar to a solid modeling language (Chapter 2.2). A component is described by the union of some primitives and modifiers. Figure 4.17 shows the description of a rotational component. CYLE (cylinder), CHAL (chamfer left), CHAR (chamfer right), UNCUL (undercut) and RADIR (radius right -curved chamfer) are primitives and modifiers. A process planner can model a component using the language. Material, processes, machine selection and time estimates can be selected by the system using the input model. Although reasonably complex components can be modeled, this language lacks a complete set of Boolean operators, and modeling a complex component may be difficult. The process sequence is also directly affected by the sequence with which a component is modeled. Although the system models a component from left to right, it does not reduce the number of possible models for a component to one.

Another system CIMS/PRO developed by Iwata et. al. [1980] uses an input language called CIMS/DEC [Kakino et. al. 1977]. In the CIMS/DEC system, component shape is modeled by sweeping (translation or rotation) to generate surfaces (Figure 4.18). In CIMS/PRO, a pattern recognition module automatically identifies machined surfaces such as planes, cylinders, threaded shafts, holes, grooves etc.



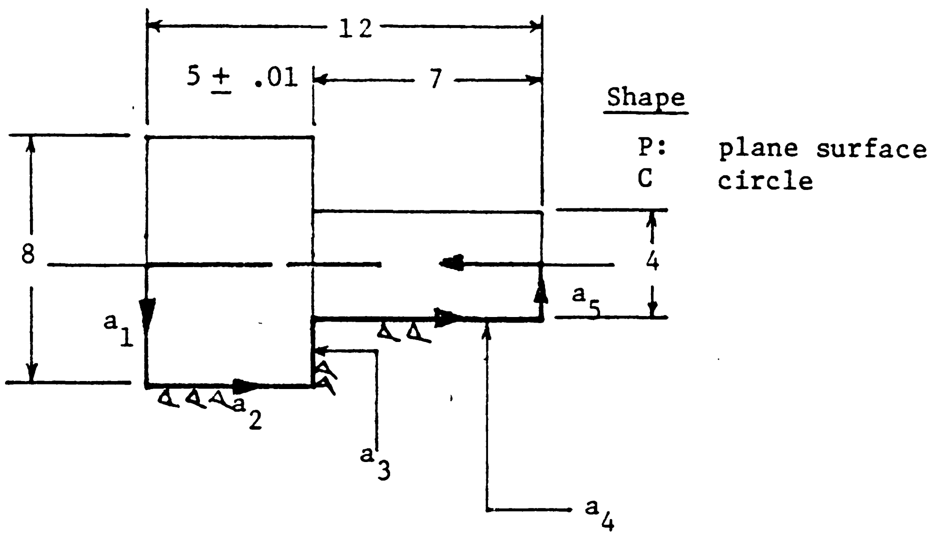
```
10 CYLE/52,25/
11 DFIT/K,6/
12 CHAL/1,45/
13 RADIR/5/
20 CYLE/76,42/
21 LTOL/+0.01,-0.01/
30 CYLE/32,39/
31 UNCUL/3,0.8/
32 VALHE/2,13/
33 CHAR/2,45/
```

Figure 4.17 <u>AUTAP Data Input</u>
(taken from Eversheim 1980)

Twenty-six different types of machined surfaces can be characterized. Tool approach can also be determined (every machined surface has a set of approach directions predefined). This input language can model both rotational (by rotation sweep) and box-like (by translation sweep) components. It is most powerful for modeling rotational components without axial shape elements. However if the component is very complex, it can be difficult to model.

GIRI [Descotte and Latombe 1981] is an Artificial Intelligence (AI) problem solver. It is one of the earliest attempts to use AI in process planning. A component can be described by some system words such as diameter, surface-finish, etc. (Figure 4.19). Rules can then be applied to determine the processes and machines needed to produce a part. The knowledge base (where process and machine capabilities are stored) uses the same set of system words; therefore decisions can be reached by searching the knowledge base in order to satisfy the input description. The input description must however be prepared by a human operator. For a complex component, the translation of original design to this input language can be very tedious and difficult.

There are many other systems (such as CPPP [Kotler 1980], AUTOTECH [Tempelhof 1980], etc.) which use their own



	Shape Code	Dimensions		S.F.	Tol.	Adjacent Shape
a_1	P	0	4		±.01	a_6
a_2	C	5	0	▷▷▷		a_1
a_3	P	0	2	▷▷		a_2
a_4	C	7	0	▷▷		a_3
a_5	P	0	2			a_4
a_6	*	-12	0			a_5

Figure 4.18 CIMS/DEC Component Modeling
(Taken from Iwata 1980)

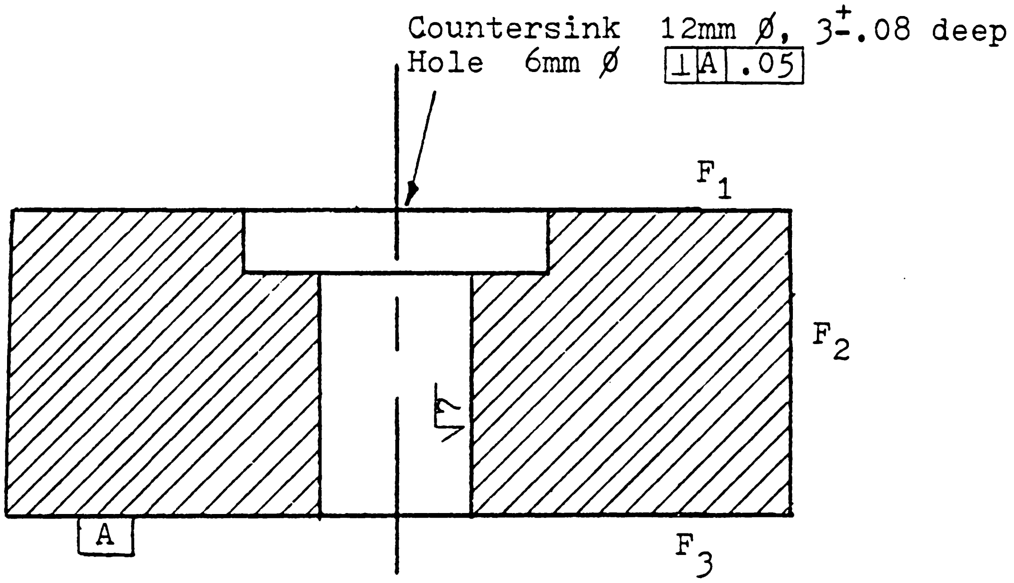
special description language. Basically all of these systems contain a surface shape code, dimensions and technological data. Although description languages can provide complete information for process planning functions, the main problem (the difficulty to generate the original design automatically)is still unresolved. The next class of input format is aimed at eliminating this problem.

4.3.3 CAD MODELS

Since a design can be modeled in a CAD system effectively, using a CAD model as input to a process planning system can eliminate the human effort of translating a design into a code or other descriptive form. The increasing use of CAD in industry further points to the benefits of using CAD models as process planning data input.

A CAD model contains all the detailed information about a design. It can provide information for all planning functions. However, an algorithm to identify a general machined surface from a CAD model does not currently exist. Additional code is needed to specify the machined surface shape from raw material shape.

CADCAM [Chang and Wysk 1981] uses a CAD model as its input. Several other sytems (AUTOPLAN [Vogel 1981], GENPLAN, etc.) also use a CAD data base interactively for



```
( H  ( type  countersink-hole )
     ( starting-from  F1 )
     ( opening-on  F3 )
     ( diameter  6 )
     ( countersink-dia  12 )
     ( surface-finish  7 ) )
( distance  H  F2  19  )
( countersink-depth  H  F1  3  +- 80 )
( perpendicularity  H  F3  +- 50  )
```

Figure 4.19 GIRI Component Model

tool and fixture selection. There is tremendous potential for using CAD data for process planning input. However this frontier still must be developed.

4.3.4 Decision Logic

In a generative process planning system, the system decision logic is the focus of the software and directs the flow of program control. The decision logic determines how a process or processes are selected. The major function of the decision logic is to match the process capabilities with the design specification. As discussed in Chapter 3, process capabilities can be described by "IF --- THEN ---" expressions. Such expressions can be translated into logical statements in a computer program. Perhaps the most efficient way to translate these expressions is to directly code process capability expressions into a computer language. Information in handbooks or process boundary table can be easily translated using a high level computer language. However such programs can be very long and inefficient. Even more disadvantagous is the inflexibility of such software: this inflexibility leaves customized codes of this type virtually useless in process planning.

In Chapter 3, we also discussed process capability representation methods. Several methods can be used to describe the decision structure of process planning. The knowledge representation methods are directly related to the decision logic in these systems. The static data is the representation and the dynamic use of the data becomes the decision logic. In this section, we will discuss the following decision logic as applied to process planning systems:

1. decision trees,

2. decision tables, and

3. artificial intelligence.

This list is by no means complete. However, this classification forms a handy framework for discussion.

4.3.4.1 Decision Tree

A decision tree is a natural way to represent process information. Conditions (IF) are set on branches of the tree and predetermined actions can be found at the junction of each branch.

A decision tree can be implemented as either: 1) computer code, or 2) presented as data. When a decision tree is implemented in computer code, the tree can be directly translated into a program flow chart. The root is the start node (Figure 4.20), and each branch is a decision node. Each branch has a decision statement, a true condition and a false condition. At each junction, an action block is included for the true condition. For a false condition, another branch might be taken or the process might be directed to the end of the logic block. When the false condition includes another branch, these two branches are said to branch from an OR node. When the false condition goes directly to the end of an action block (which is rooted from the same decision statement), then the current branch and the following branch are part of the same AND node. A decision statement can be a predicate or a mathematical expression.

Figure 4.21 shows an sample decision tree and its flowchart representation. It can be written in a programming language (pseudo-language) as follows:

```
;root
;
     IF E1 then do N1 enddo
     else if E1 then do A5 end do
     else end if endif stop

: node N1
;
procedure N1
If E2 then do N2 enddo
     else if E3 then do A4 enddo
```

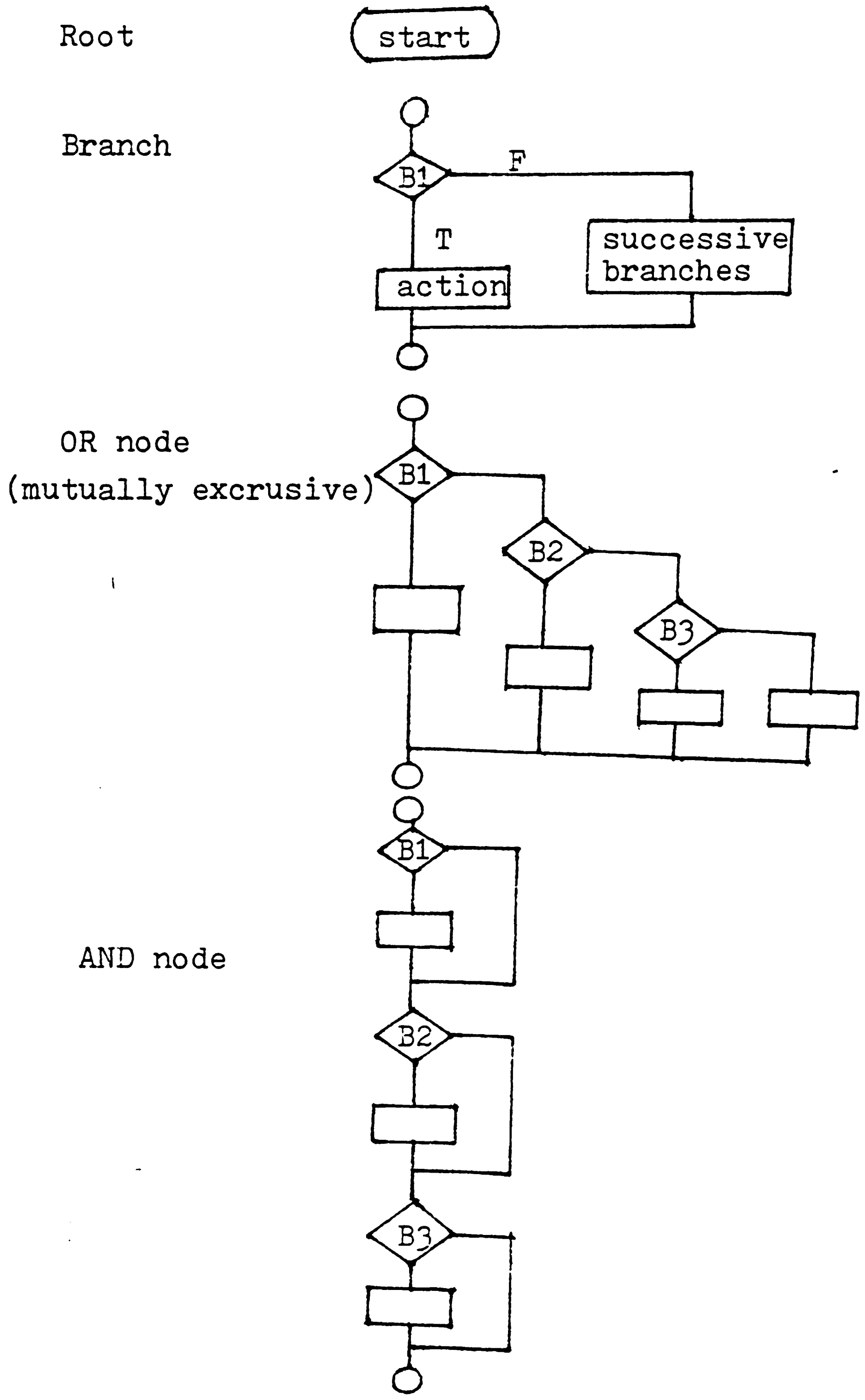
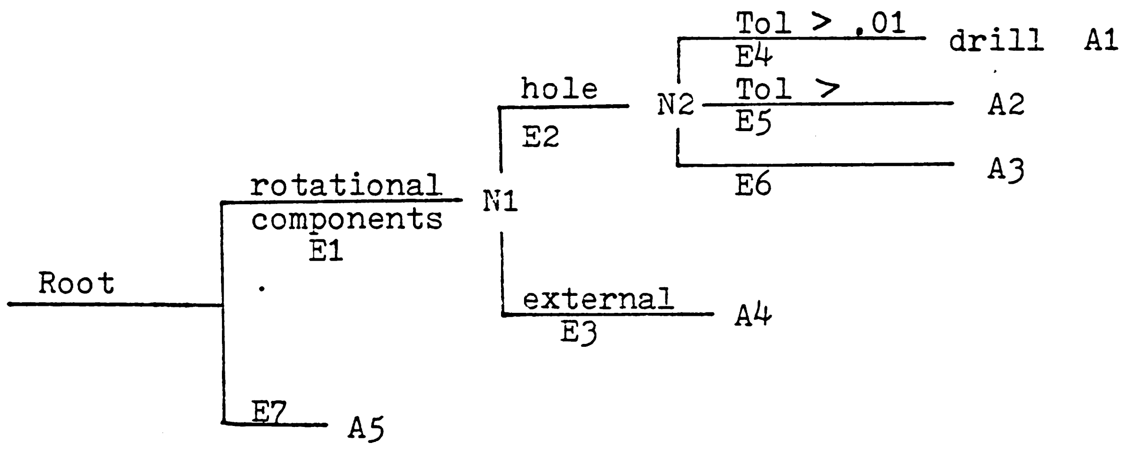


Figure 4.20 Structured Flow Chart Corresponds a Decision Tree

```
     else end if endif return
; node N2
;
procedure N2
If E4 then do A1 enddo
     else if E5 then do A2 enddo
     else if E6 then do A3 enddo
     else endif endif endif return.
```

This language (or interpreter) format allows for the easy construction of decision trees which are frequently used in generative process planning systems. Similar language formats using FORTRAN, PASCAL, PL/I, etc. have been developed for general purpose algorithms. APPAS [Wysk 1977] is a typical example of decision tree logic used for process planning. Although the approach is easy to implement, system expansion work can be difficult, especially for a programmer other than the creator.

When implementing a decision tree in data form, another program (system program) is required to interpret the data and achieve the decision tree flow. This approach is more difficult to develop (for the system program). However, once the system program has been developed, the implementation and system maintenance is significantly leasened. Again however, it can be extremely difficult to



E_i : represent an expression or a series of expressions

A_i : represent an action

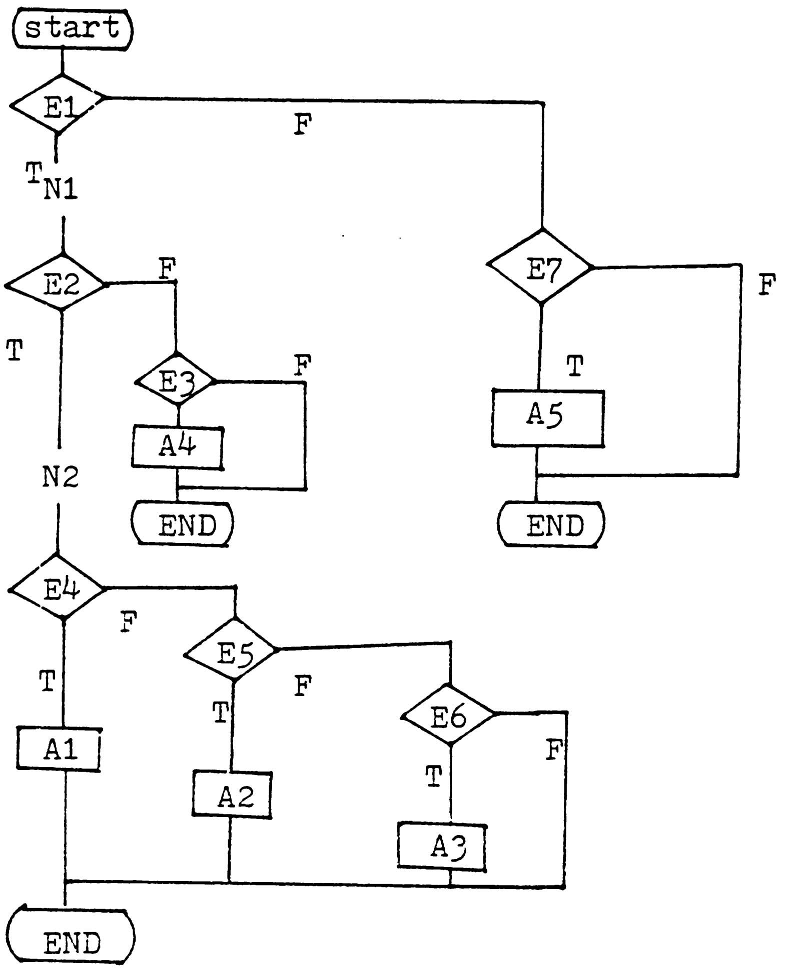


Figure 4.21 Implementation of a Decision Tree in a Program

add a function which was not originally included in the system program. There are many methods that can be used to design such a system program. A simple example will be presented to demonstrate a basic structure that one can use.

We will call the example system DCTREE. DCTREE uses a query system to obtain design information, and then print the final conclusions. In DCTREE, there are three major components: 1) the decision tree data; 2) a compiler; and 3) a system run time module. Part one is supplied by a user who translates his decision tree from graph form into DCTREE input language format. The DCTREE compiler compiles the input, and saves it in a computer usable format. Finally, the system run time module uses the compiled decision tree to generate questions, make decisions and print out conclusions.

We will first look at the input language. There are two parts of each input: 1) expression definition, and 2) tree structure definition (Figure 4.22). In the expression definition, each expression is followed by an expression identifier (id). Each id must be unique. An expression with an id initial of Q or A (query or action) is simply stored in a buffer. Other expressions will be compiled as condition expressions. A condition expression (such as &1$\leq$ .002) uses post fix notation and stack operations. A variable (&1SX) will cause the run time module to input a real number and store it on a stack. Therefore these expressions can be compiled as a simple code instead of a variable.

The tree strtucture is represented by expression ids and pointers. For instance, an arrow $\rightarrow$ represents "point to". The syntax is:

$$E_{n_o} \leftrightarrow \begin{bmatrix} \text{AND} \\ \text{OR} \end{bmatrix} (E_{n_1}, E_{n_2}, \ldots E_{n_m}) \mid A_i$$

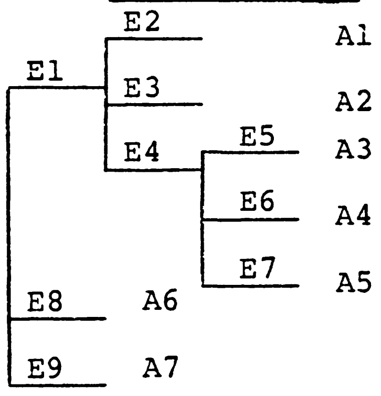
where:

E_{n_o}	:	root branch (sthece)
E_{n_i}	:	expression number (destimation action)
A_i	:	execution action
$\mid$	:	either E's or A's but not both

During compiling each E_n is assigned an address in the tree structure file. E_{n_i} 's in parenthesis are substituted by pointers. A_i 's are marked with negative values to indicate their actions. Figure 4.22 shows how a decision tree can be represented by DCTREE.

The system run time module does the I/O and decision making. In Figure 4.23, a run time module algorithm is shown. Where $E_i \rightarrow$ represents the expression pointed to by the root branch, each time $E_i \rightarrow$ is called in a procedure, it increments forward to the next branch (i.e. $E_{n_o} \rightarrow$ yields AND or OR, $E_{n_o} \rightarrow$ again yields E_{n_1} , $E_{n_o} \rightarrow$ yields E_{n_2}, ...).

Decision Tree



Expression Definition

Q1	Hole diameter?
E1	&1 > 0.0
Q2	True position?
E2	&1 ≤ .002
E3	(&1 ≤ .01).AND.(.002 < &1)
E4	&1 > .01
Q5	Tolerance?
E5	&1 ≤ .002
E6	(&1 ≤ .01).AND.(.002 < &1)
E7	.01 < &1
A1	Rapid travel out, True position = .01
A2	Finish bore, True position = .02
A3	Finish bore, Tolerance = .01
A4	Semi-finish bore, Tolerance = .02
A5	Drill, Diameter = 0
08	Slot?
09	Internal thread?
E8	&1
E9	&1
A6	Mill
A7	Tap

Tree Structure Definition

```
 0 → OR (E1 E8 E9)
E1 → AND (E2 E3 E4)
E2 → A1
E3 → A2
E4 → AND (E5 E6 E7)
E5 → A3
E6 → A4
E7 → A5
E8 → A6
E9 → A7
```

Figure 4.22 Input to DCTREE

A(K) and Q(K) are obtained from expression definitions. E(K) signal the system to evaluate expression definition E_k. The recursive algorithm can evaluate the entire tree structureand return conclusions.

4.3.4.2 Decision Tables

Decision tables have long been a popular method of presenting complex engineering data. Decision tables can also be easily implemented on a computer. Using decision tables for process planning however normally requires a special preprocessor program or computer language to implement the table and control the operation of the table. Such software is generally called a decision table language. A decision table language consists of:

1. a base language,
2. a decision table, and
3. an outer language.

A base language is the foundation of a decision table language. For example, FORTAB of the RAND corporation and S/360 DLT of the IBM corporation use FORTRAN as their base language [McDaniel 1970]. DETAB/65 [Silberg 1971] of SIGPLAN of ACM (Association of Computing Machinery) uses COBOL as its base language. A base language is extended to

```
Procedure DCTREE

Do tree (0 , J) enddo
   (:0: root)

stop
;
proceudre tree ($E_i$,J)

set J := 'F'    (: J = 'T' search successful)

set iflag:= 'T' (:iflag = 'T' or node)

set K  := $E_i$ → (:K pointer to first object)

while K ≠ 0 do

  if K < 0 then write A(-K) and set J := 'T'

     (: reach the terminal)

  else if K = 'and' then set iflag := 'F'

                (: define and node)

                else write Q(K) and

                if E(K) then do tree (K, J) and

                   if J·iflag then set K = 0

                   else K = $E_i$ → end if enddo

                endif endif endif

   enddo return
```

Figure 4.23 Run Time Module Algorithm

include statements which can describe a decision table in a more easily implemented manner. A preprocesser is occassionally written to translate the decision table language program into its base language program.

The decision table is the most essential part of a decision table language program. A decision table is represented in its original table format. For example the decision tree in Figure 4.21 can be represented by the decision table in Figure 4.24. Using a pseudo-language, this decision table can be written as shown in Figure 4.25. Decision table techniques discussed in Chapter 3 can be used to simplify and/or parse complex tables.

The third element (the outer language) is used to control the decision table. Figure 4.25 illustrates that a decision table does not contain any input/output or control statements. Therefore, it is not a complete program. An outer language can eliminate this void. Again, we can use the example to show how this can be done, and DCTABLE will be the name given the decision table language. The table in Figure 4.25 will be written in DCTABLE. A procedure, TAB(N), will evaluate a table N. All variables in DCTABLE are global variables, therefore, no parameters are passed. When table parsing is required, a TAB(N1) can be added to the stub of a table N; therefore, several tables can be

Hole	X	X	X	X	X		
Diameter = 0.0	X	X	X	X	X		
Slot						X	
Internal Thread							X
T.P. $\leq$ .002	X						
.002 $<$ T.P. $\leq$ .01		X					
.01 $<$ T.P.			X	X	X		
Tol $\leq$ .002			X				
.002 $<$ Tol $\leq$ .01				X			
.01 $<$ Tol					X		
Rapid travel out	X						
Finish bore		X	X				
Semi-finish bore				X			
Drill					X		
Mill						X	
Tap							X
T.P. = .01	X						
T.P. = .02		X					
Tol = .01			X				
Tol = .02				X			
Diameter = 0					X		

Figure 4.24 A Decision Table

```
$ 100 table
C
  Shape  = hole                   .T.   .T.   .T.   .T.   .T.
  Dia    > 0.0                    .T.   .T.   .T.   .T.   .T.   .T.   .T.
  Shape  = Slot                                                 .T.
  Shape  = I thread                                                   .T.
  TP     ≤ .002                   .T.
  (.002 < TP).and.(TP ≤ .01)            .T.
  .01 < TP                                    .T.   .T.   .T.
  Tol ≤ .002                                  .T.
  (.002 < TOL).and.(TOL ≤ .01)                      .T.
  .01 < TOL                                               .T.
C
  P:= R-T-O                        X
  P:= F-B                                X     X
  P:= S-F-B                                          X
  P:= Drill                                                X
  P:= Mill                                                       X
  P:= Tap                                                              X
  TP  := .01                       X
  TP  := .02                             X
  TOL := .01                                   X
  TOL := .02                                         X
  DIA := 0                                                X      X     X
C
$ENDT
```

Figure 4.25 Decision Table Program

connected. The following program demonstrates the control of a decision table.

```
(: Decision table program for the process selection)
read shape, dia., TP, TOL
Set IFIND := 1
set P := b̸
while IFIND = 1 DO TAB(100) and
write 'Process selected' , P and
If P = b̸ then set IFIND := 0
else set IFIND := 1 endif and
set P := b̸ enddo
stop
```

In the above program shape, diameter, true position and tolerance data are input. A process array P which stores a selected process name is set empty (b̸). Flag IFIND is set as 1. When none of the rules in the table are true, IF1ND becomes zero. A procedure: while --- do --- enddo, is executed until no rule is true.

A simple algorithm for TAB(N) can be shown as:

Let C_i be the condition stub expressions $i = 1,2,...,n$

A_j be the action stub expressions $j = 1,2,...,n$

R_{kl} be the rule entries $k = 1,2,...,n+m$

$l = 1,2,...,M$

Step 1. Set $l := 0$, while $l < M$ do step 2 through step 5 enddo.

Step 2. Set $k := 0$, set LOGIC: := .T., set $l := l + 1$

Step 3. Set $k := k + 1$ while $R_{kl} \neq$ b̸ do set 4 enddo.

Step 4. If $C_k \neq R_{kl}$ then set $k := n + 1$ else LOGIC:= .F. endif.

```
Step 5.  If LOGIC = .T. then set l:= M + 1 and do step 6.
         else endif
Step 6.  For j  = 1 to m do step 7 enddo.
Step 7.  If $R_{j+n,l} \neq \not{b}$ then do $A_j$ enddo else endif.
```

In the above example, we assume a decision table language is used. However, one may not find a formal decision table language appropriate for the programming of his entire process planning system. One can always implement decision table logic using a procedure oriented language, such as FORTRAN, PL/I, PASCAL, etc. Although the implementation is not as easy as using a decision table language, it is not so difficult so as to prohibit the use of decision table logic. The algorithm for TAB can be used with minor modification. C can be a procedure (subprogram) which consists of n expressions, and A another procedure which consists of m expressions (Figure 4.26). The variables used are global variables stored in common blocks. Each time C is called, a parameter, k, is required to index the expression The logic returns a Boolean value, .T. or .F.. A is similar to C except it does not return any value.

Figure 4.26 and 4.27 show an implementation using FORTRAN. Entries of the table can be assigned as either data statements or input as data.

4.3.5 Artificial Intelligence

Artifical Intelligence (AI) has become one of the major topics of discussion in computer science. AI can be defined as the ability of a device to perform functions that are normally associated with human intelligence. These functions include reasoning, planning, problem solving, etc. Applications for AI have been in natural language processing, intelligent data base retrieval, expert consulting systems, theorem proving, robotics, scheduling and perception problems [Nilsson 1980]. Process planning applications have been considered as part of an expert consulting system.

In an expert system, the knowledge of human experts is represented in an appropriate format. The most common approach is to represent knowledge by using rules. Rule-based-deduction is frequently used to find an action. Since AI is too large a subject to discuss in this text, potential applications in process planning will be illustrated.

There are two types of knowledge involved in process planning systems: component knowledge and process knowledge. The component knowledge defines the current state of

```
     LOGIC FUNCTION

     Common IShape, TP, TOL, DIA, P

     INTEGER DATA HOLE/'HOLE'/,SLOT/'SLOT'/,THREAD/'THREAD'/


     Go To (10, 20, 30, 40, 50, 60, 70, 80, 90, 100),k

 10  C = IShape .EQ.Hole
     return

 20  C = DIA .GT. 0.0
     return

 30  C = IShape .EQ. SLOT
     return

 40  C = IShape .EQ. THREAD
     return

 50  C = TP .LE. .002
     return

 60  C = TP.GT.0.002 .AND. TP .LE. 0.01
     return

 70  C = TP .GT. 0.01
     return

 80  C = TOL .LE. 0.002
     return

 90  C = 0.002 .LT. TOL .AND. TOL .LE. 0.01
     return

100  C = 0.01 .LT. TOL
     return
     end
```

Figure 4.26 <u>Condition Stub Implemented in a Function Subprogram</u>

```
{ action stub
{ subroutine A(k)

   common ISHAPE, TP, TOL, DIA, P

   Integer Data  RTO/'RTO '/,FB/'F-B '/,SFB/'SFB '/,
                 DRL/'DRL '/,MILL/'MILL'/,TAP/'TAP '/

   Go To (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110),k

 10   P = RTO
      return

 20   P = FB
      return

 30   P = SFB
      return

 40   P = DRL
      return

 50   P = MILL
      return

 60   P = TAP
      return

 70   TP = .01
      return

 80   TP = .02
      return

 90   TOL = .01
      return

100   TOL = .02
      return

110   DIA = 0
      return
      end
```

Figure 4.27 Action Stub implemented in a Subroutine

the problem to be solved (it is also called declarative knowledge). On the other hand, the knowledge of processes defines how the component can be changed by processes (It is also called procedural knowledge). Applying the process knowledge to a component in a logical manner is called control knowledge.

There are several methods available to represent declarative knowledge. First order predicate calculus (FOPC), frams and semantic networks [Nau 1981] are two popular methods. Since FOPC is used more often, we will only consider it. FOPC is a formal language in which a wide variety of statements can be expressed. In a predicate calculus language, a statement is expressed by predicate symbols, variable symbols, function symbols, and constant symbols. For example.

Depth (Hole(X), 2.5)

represents the depth of hole(X) as 2.5 units. Such a representation is called an atomic formula. In the atomic formula, depth is a predicate symbol. Hole is a function symbol; X is a variable symbol; and 2.5 is a constant. A legitimate atomic formula is called the well-formed formula (wff). The above atomic formula is a wff. A wff can either be T or F. For example, the depth of hole (1) is 2.0;

therefore, depth (HOLE(1),2.5) is false. When we use FOPC to describe a component, all wff must be true. The example shown in Figure 4.19 can also be considered descriptive knowledge for the hole design.

Procedural knowledge can be represented by

IF (condition) THEN (action)

statements which are similar to decision trees or decision tables. In AI, such statements can be called production rules. A system using production rules to describe its procedural knowledge is called a production system. In a production rule, a condition can be a conjunction of predicates (wff). For example,

```
( = (shape &x) hole)
( > (DIA &x) 0.0)
( ≤ (TP &x) 0.002))
==>
(rapid-travel-out &x)
( := (TP &x) 0.01) 0.8)
```

where (= (shape &x) hole), (> (Dia &x) 0.0) and (≤ (TP &x) 0.002) are wffs (prefix notation). They form the condition of a production rule. The symbol '==>' represents 'THEN'. Actions of the rule are assigned rapid-travel-out to surface

&x and change TP (true position) to 0.01. The 0.8 in the action implies a weighting factor or preference of accepting these actions. It is useful when several rules can be applied for a certain state (e.g. several processes or sequences can be used). The AI process planning system GARI and the medical consulting system MYCIN [Davis 1977] use a similar representation for their procedural knowledge.

Even after the descriptive and procedureal knowledge have been represented, conclusions (process plans) still cannot be obtained, because we do not have a mechanism to apply the appropriate rule(s) to the problem domain (descriptive knowledge). The control knowledge is similar to that of the human knowledge on reasoning which deduces certain facts from the knolwedge base concerning a problem. This can be very difficult to program on a computer.

Nilsson [1980] listed three control strategy approaches. First, a control strategy problem can be solved by another AI production system. The system for the control strategy problem is called a meta-level AI system. A meta-level AI system has declarative and procedural knowledge relevant to the control of the original problem.

A second approach uses AND/OR solution graphs. In an AND/OR graph, either a goal or a fact can be decomposed and later combined by production rules to reach the opposite state (i.e. from the goal to the fact -- backward planning, or from the fact to the goal -- forward planning). An AND/OR graph is shown in Figure 4.28. In the graph, the goal is to find A. Rule 1 is (B AND C => A), Rule 2 is (D OR E => B), Rule 3 is (F AND G => C). D, E, F and G are facts in the original state. Using rules 1, 2, 3, we can find the goal A.

A third method embeds the control knowledge in the rules. The rule also acts like a program. For example, a GOAL statement will search for a goal from the knowledge base; (i.e, run a rule program); an ASSERT statement adds a new fact (predicate expression; a cp statement returns the control of the program to the program from which it was invoked; and an IF statement, evaluates the predicate expression, "If it is T then terminate the current program and return". In the system we have three facts and two rules, prefix notation is adopted.

The top-level control program can be written:

```
BEGIN
Read "surface id" ?x
goal (make-hole ?x)
end
```

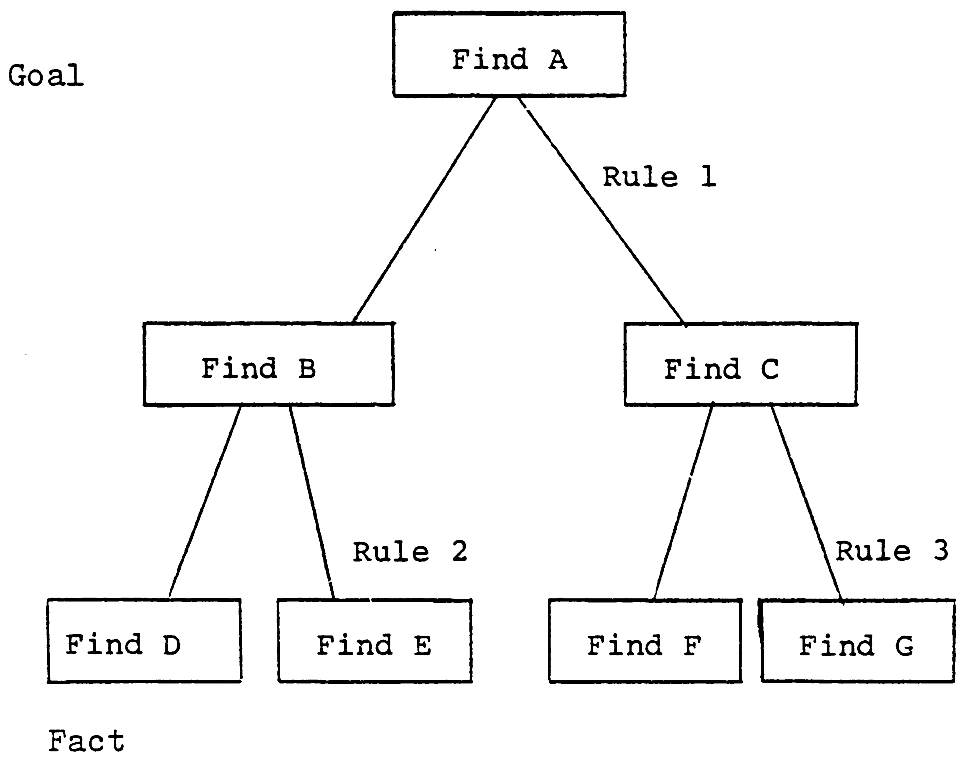


Figure 4.28 <u>AND/OR GRAPH</u>

When this program is running, the first input will be for hole-1, the first surface to be planned. The GOAL (make-hole hole-1) will run R3. The first expression in R3, GOAL (Bore hole-1) runs R2. In R2, the first expression GOAL (hole-1 dia 1) returns ?y=2. Since ?y > 0.0, control passes to the next expression. The ASSERT expression changes Tel and TP in hole-1 to 0.01. Then BORE is printed. Using the same logic, DRILL can be found and printed. The above example only shows a very simple application. More complicated control knowledge can also be represented.

AI is a comparably new field in Computer Science (CS). Although there are numerous research programs in CS, few applications for process planning have evolved. Because of the fast deduction capability, AI systems have potential to plan a component in a global sense (consider interrelationships of surfaces) compared with the local planning (surface-by-surface planning) of most other approaches. However a significant amount of effort is required before it can become a practical tool for process planning.

4.3.6 Closing Remarks

In this chapter, all the information discussed in the previous chapters was integrated. A Computer-aided process planning system is a system which reads in design specifications then uses the built-in knowlege to determine the required manufacturing processes. As discussed, there are two approaches -- variant and generative used. The concept, structure and complexity of systems using these two approaches are very different. These two approaches have been distinguished. A simplified variant system (VP) was discussed in detail to illustrate how to build systems. Since most variant process planning systems are "similar", the VP system represents a typical variant process planning system.

There are many methods (decision logic) which have been and can be used in generative planning. Three major decision procedures were discussed. The majority of existing systems use one of these three methods. However, the detailed structure is limited only by one's imagination.

Thus far, the complete concept of computer-aided process planning has been presented. In the next chapter, some existing systems will be reviewed to show how they are constructed.

Chapter V

COMPUTER-AIDED PROCESS PLANNING SYSTEMS

5.1 IMPLEMENTATION CONSIDERATIONS

From the discussion in previous chapters, we can conclude that the process planning function is manufacturing system dependent. This implies that no one single process planning system can satisfy all of the different manufacturing systems needs. The same process planning system can be used for two different manufacturing systems with little modification of the process knowledge data base, if these two manufacturing systems are similar (equipment and products). However this is no longer true when the manufacturing systems are significantly different (equipment and products).

There are several factors which must be considered when one attempts to implement a process planning system. These include:

1. Manufacturing system components,
2. Production volume/batch size, and
3. Number of different production families.

5.1.1 Manufacturing System Components

The planning function is directly affected by the capability of the manufacturing system which will be used to produce the component. A manufacturing system which has precision machining centers can produce most of its products on a single machining center. However, in other systems, several addition processes and machines may be required in order to achieve the same precision. This is true for process and machine selection, and even more true for parameter selection or optimization. Process knowledge depends on the manufacturing system.

A variant process planning system contains no process knowledge per se. Process plans are retrieved from a data base. The standard process plans in the data base were manually prepared by human process planners. The capability of the manufacturing system was taken into account by the process planners when they prepared the process plans. Consequently, the standard plans must be changed when major factory renovation takes place. Variant process planning structures have been adopted by different manufacturing systems; however, the data base must be rebuilt for each system. The lengthy preparation stage is unavoidable. Implementation of a variant planning system requires that a unique data base of component families and standard plans be constructed.

On the other hand, a generative process planning sytsem has its own knowledge base which stores manufacturing data. The user modifies the knowledge base in order to suit his/ her system. A generative process planning system provides the user with a method to describe the capabilities of his manufacturing system. The process knowledge representation as described in Chapter 3 can be created and stored using several methods. It is essential however that the user interact with the system so that he/she can easily change the process knowledge base. This must be considered when purchasing or creating a generative process planning system.

5.1.2 Production Volume/Batch Size

Production volume is a major consideration in the selection of production equipment. As a basic rule of thumb, special purpose machines and tooling are used for mass production, and general purpose equipment is used for small batch production. The economics of production determines this decision. These same "economics of scale" must also be applied to process planning.

Different process plans should be used for the same component design when the production volume is significantly different. For example, it is appropriate to turn a thread on an engine lathe if we only need one screw. However, when ten thousand screws are required, a threading die should be considered. Machining a casting is also more desirable than complete part fabrication for large production volume.

In a variant process planning system, the production volume can be included as a code digit. However, standard plans must be prepared for different levels of production. Preparing standard plans for each production level may not be feasible, since components in the same family may need quite different processes when the production volume increases. When the batch size is not included in the family formation, manual modification of the standard plan for different production volumes is necessary.

In a generative process planning system, the production volume can be considered a variable in the decision model. Ideally, the knowledge base of a system includes this variable. Processes, as well as machines, and tool selection are based not only on shape, tolerances and surface finish but also on the production volume.

5.1.3 Number of Different Production Families

The number of different production families used is a function of how different the components being planned are. A variant proces planning system is of little value if there are many families and few similar components (family members), because the majority of effort will be spent adding new families and standard plans. A generative process planning system is more desirable for this type of manufacturing environment.

Certain manufacturing systems specialized in making similar or identical components for different assemblies. These components can usually be designated by their features (we typically call them composite components). Since each feature can be machined by one or a series of processes, a model can be developed for each family. In the model, processes corresponding to each feature are stored. A process plan can be generated by retrieving processes which correspond to the features on a new component. This type of generative process planning system utilizes the major benefit of Group Technology. It is easier to build this type of system than a general generative process planning system; yet, it can provide very detailed plans (depending on the part families being producted).

For a moderate amount of component families and many similar components in each family, a variant process planning system is usually the most economic automated planning alternative. Although a long variant preparation stage is still necessary, a variant process planning system is much easier to develop. Such a system can be very effectively used when there are many similar components. On-the-other-hand, significant product variation usually dictates generative process planning as the most economic alternative. Figure 5.1 illustrates the economic regions for the different planning alternative.

5.2 A SURVEY OF PROCESS PLANNING SYSTEMS

The majority of existing process planning systems are of a variant nature, e.g. CAPP, MIPLAN, MITURN, MIAPP, ACUDATA/UNIVATION, CINTURN, COMCAPP V, etc. However there are some generative sytems, such as CPPP, AUTAP, APPAS, etc. Table 5.1 contains a list of some process planning systems and their characteristics. A check in the various columns of the table indicates that that corresponding planning function, shape or other characteristic is part of a planning systems.

Input: Input in Table 5.1 refers directly to the information required for selecting processes. A code implies that a part or GT code is used as input for describing parts. When a particular coding system is used, it is indicated in the entry. "Special language" implies that a part description language

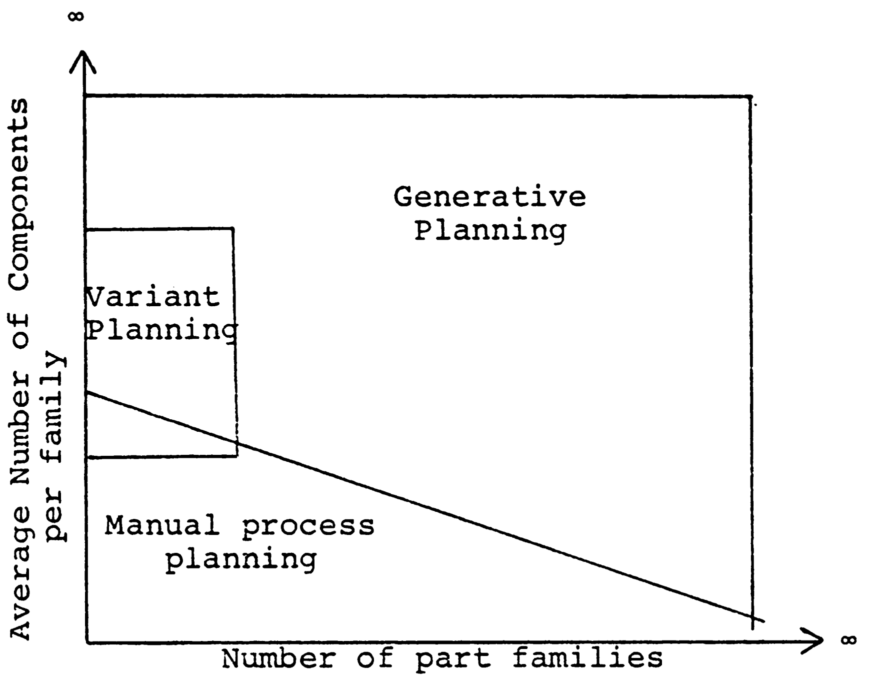


Figure 5.1 Economic Regions for Different Process Planning Systems.

is used. "CAD data base" means that the input description for a part comes from a CAD model. When two entries are marked, it implies that either both types of input are required or the system is capable of using either.

Planning approach: Planning approaches can either be variant or generative. "Code based" implies that the process plan retrieved or generation system is based on the code (the input section). When more than one entry is marked, it implies that more than one approach can be used. "NA" indicates it is not applicable.

Output content: Various types of inforamtion can be retrieved or generated by process planning sytems. In the case of a pure retrieval system, detailed information can be retrieved from a standard plan. For this case only the process sequence entry is marked.

User supplied logic: When user supplied decision logic is required before the system can function, a check is indicated in this entry.

Commerical/Academic: This cateogry indicates the nature of the system development. When a system is especially developed for a single company or when it is unclear what the nature of the system is, the entry is left blank.

Designer refers to the individual or organization that developed the system.

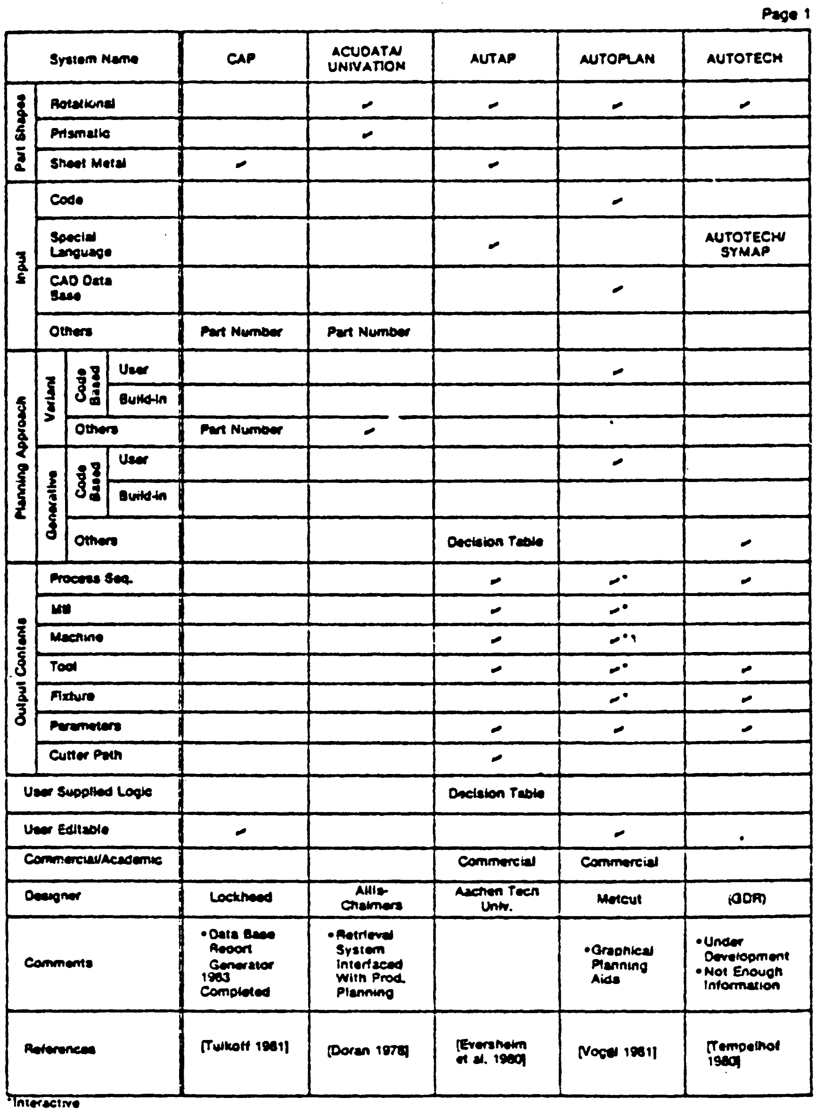
References: References can be located at the end of this thesis.

In the following sections, we will discuss some process planning systems in more detail.

5.2.1 CAM-I CAPP System

The CAM-I Automated Process Planning (CAPP) system [Link 1976, Schilperoot 1975] is perhaps the most widely used of all process planning systems. It is a variant system developed by McDonnell Douglas Automation Company (McAu-

Table 5.1 Summary Table (taken from Chang 1982)



Page 1

System Name				CAP	ACUDATA/ UNIVATION	AUTAP	AUTOPLAN	AUTOTECH
Part Shapes	Rotational				✓	✓	✓	✓
	Prismatic				✓			
	Sheet Metal			✓		✓		
Input	Code						✓	
	Special Language					✓		AUTOTECH/ SYMAP
	CAD Data Base						✓	
	Others			Part Number	Part Number			
Planning Approach	Variant	Code Based	User				✓	
			Build-in					
		Others		Part Number	✓			
	Generative	Code Based	User				✓	
			Build-in					
		Others				Decision Table		✓
Output Contents	Process Seq.					✓	✓*	✓
	Mill					✓	✓*	
	Machine					✓	✓*	
	Tool					✓	✓*	✓
	Fixture						✓*	✓
	Parameters					✓	✓	✓
	Cutter Path					✓		
User Supplied Logic						Decision Table		
User Editable				✓			✓	•
Commercial/Academic						Commercial	Commercial	
Designer				Lockheed	Allis-Chalmers	Aachen Tech Univ.	Metcut	(GDR)
Comments				• Data Base Report Generator 1983 Completed	• Retrieval System Interfaced With Prod. Planning		• Graphical Planning Aids	• Under Development • Not Enough Information
References				[Tulkoff 1981]	[Doran 1978]	[Eversheim et al. 1980]	[Vogel 1981]	[Tempelhof 1980]

*Interactive

Table 5.1 (continue)

Page 2

System Name			APPAS	AUTOPROS	CADCAM	CAPP	CINTURN
Part Shapes	Rotational					✓	✓
	Prismatic		✓		Hole Making	✓	
	Sheet Metal					✓	
Input	Code		COFORM			✓	
	Special Language						
	CAD Data Base				✓		
	Others						
Planning Approach	Variant	Code Based: User				✓	
		Code Based: Build-in					
		Others					
	Generative	Code Based: User					
		Code Based: Build-in	Decision Tree				
		Others			Decision Table Logic		
Output Contents	Process Seq.		✓	✓	✓	✓*	
	Mtl						
	Machine						
	Tool		✓		✓		
	Fixture						
	Parameters		✓		✓		
	Cutter Path				✓		
User Supplied Logic						Std Plan	
User Editable						✓	
Commercial/Academic			Academic	Commercial	Academic	Commercial	Commercial
Designer			Wysk/Purdue	NAKK	Chang/Va Tech	McAuto/CAM-I	Cincinnati Milacron
Comments			• Detailed Surface Code 1977	• Not Enough Information	• Graphics Input • Extension of APPAS 1980	• Retrieval System 1976	• Not Enough Information
References			[Wysk 1977] [Wysk et al. 1980]	[Chisolm 1973]	[Chang 1980, 1981]	[Link 1976] [Schilperoot 1975] [Tulkoff 1978]	

*Interactive

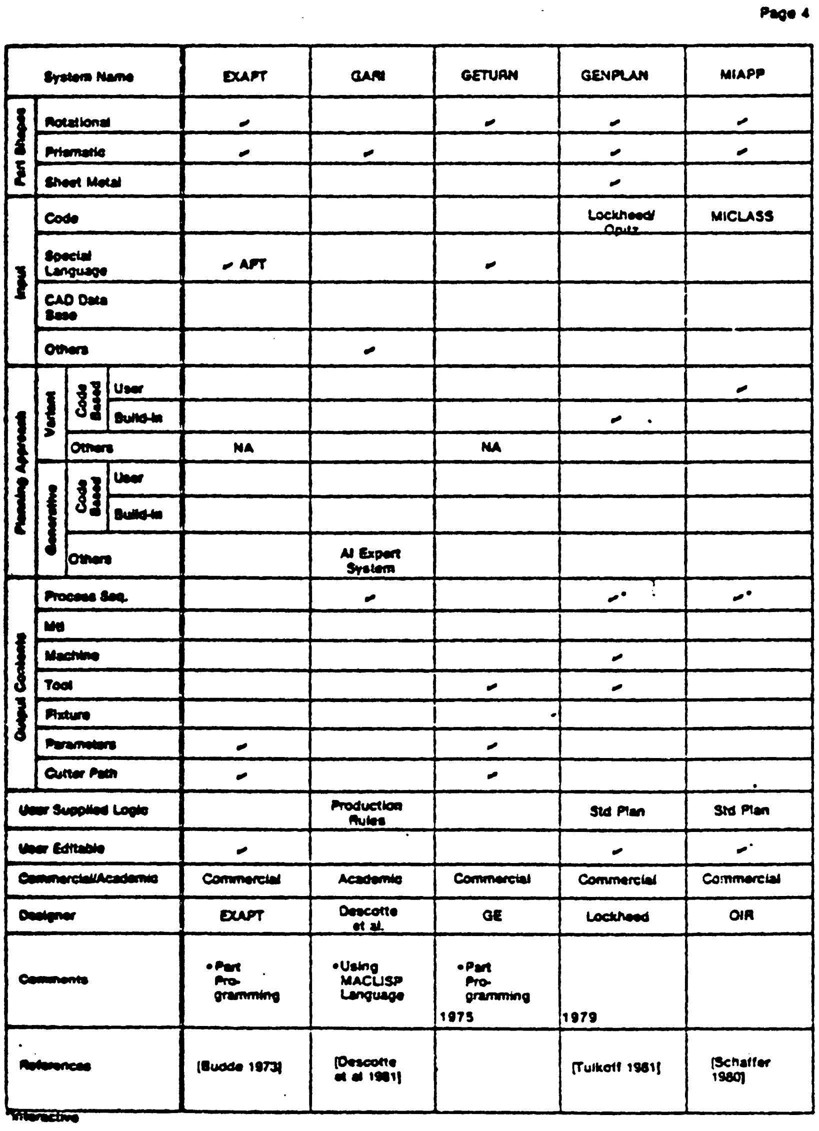
Table 5.1 (continue)

Page 3

				CIMS/PRO	COBAPP	COMCAPP V	C?PP	DCLASS
System Name								
Part Shapes	Rotational				✓	✓	✓	✓
	Prismatic			✓		✓		✓
	Sheet Metal							✓
Input	Code				APPOCC	CODE	✓	DCLASS
	Special Language			CIMS/DEC				
	CAD Data Base							
	Others						Detailed Information	
Planning Approach	Variant	Code Based	User					
			Build-in			✓		
		Others						
	Generative	Code Based	User					
			Build-in		✓			
		Others		Graph Theory			Decision Model	Decision Tree Logic
Output Contents	Process Seq.			✓	✓	✓*	✓	✓
	Mtl							✓
	Machine			✓			✓	✓
	Tool			✓			✓	✓
	Fixture			✓				
	Parameters						✓	
	Cutter Path			✓				
User Supplied Logic						Std Plan	Process Model	Decision Tree
User Editable						✓	✓	✓
Commercial/Academic					Academic	Commercial	Commercial	Commercial
Designer				Iwata et al (Japan)	Phillips/ Purdue	MDSI	UTRC	Allen/U. Utah
Comments				• Capable of Recognize the Surface Shape	1978	• Retrieval System	• Special Process Modelling Language	
References				[Iwata 1980]	[Phillips 1978]		[Dunn 1978] [Mann 1977] [Kotler 1980]	[Allen 1974, 1978, 1979, 1980]

*Interactive

Table 5.1 (continue)



Page 4

			System Name	EXAPT	GARI	GETURN	GENPLAN	MIAPP
Part Shapes			Rotational	✓		✓	✓	✓
			Prismatic	✓	✓		✓	✓
			Sheet Metal				✓	
Input			Code				Lockheed/ Opitz	MICLASS
			Special Language	✓ APT		✓		
			CAD Data Base					
			Others		✓			
Planning Approach	Variant	Code Based	User					✓
			Build-in				✓	
		Others		NA		NA		
	Generative	Code Based	User					
			Build-in					
		Others			AI Expert System			
Output Contents			Process Seq.		✓		✓*	✓*
			Mtl					
			Machine				✓	
			Tool			✓	✓	
			Fixture					
			Parameters	✓		✓		
			Cutter Path	✓		✓		
User Supplied Logic					Production Rules		Std Plan	Std Plan
User Editable				✓			✓	✓*
Commercial/Academic				Commercial	Academic	Commercial	Commercial	Commercial
Designer				EXAPT	Descotte et al.	GE	Lockheed	OIR
Comments				• Part Programming	• Using MACLISP Language	• Part Programming 1975	1979	
References				[Budde 1973]	[Descotte et al 1981]		[Tulkoff 1981]	[Schaffer 1980]

*Interactive

Table 5.1 (continue)

Page 5

System Name				MIPLAN	MITURN	PI-CAPP	PURDUE/ RECURSIVE	RPO
Part Shapes	Rotational			✓	✓	✓	✓	✓
	Prismatic			✓		✓		
	Sheet Metal					✓		
Input	Code			MICLASS	MICLASS	✓	✓	✓
	Special Language							
	CAD Data Base							✓
	Others			Part Number				
Planning Approach	Variant	Code Based	User					✓
			Build-in	MICLASS	✓	✓		
		Others						
	Generative	Code Based	User					✓
			Build-in				✓	
		Others						
Output Contents	Process Seq.			✓*	✓*	✓*	✓	✓*
	Mtl							✓*
	Machine						✓	✓*
	Tool						✓	✓*
	Fixture							✓*
	Parameters						✓	✓
	Cutter Path							
User Supplied Logic				Std. Plan	Std. Plan	Std. Plan		
User Editable				✓	✓	✓		✓
Commercial/Academic				Commerical	Commercial	Commercial	Academic	Commercial
Designer				GE/OIR	OIR	Planning Institution	Barash et al. Purdue	GE/Metcut
Comments				1980	1974	• Super CAPP • Multiple User • Graphics Output Allowed 1980	• Expenmental System	• Based on AUTOPLAN
References				[Schaffer 1980] [TNO 1981]	[TNO 1981]	[PII 1980]	[Barash 1980]	[Tipnis 1980] [Vogel 1980]

*Interactive

Table 5.1 (continue)

Page 6

System Name				SAGT	TAUPROG	XPS-1		
Part Shapes	Rotational			✓		✓		
	Prismatic					✓		
	Sheet Metal							
Input	Code			✓		For Variant		
	Special Language							
	CAD Data Base							
	Others					Code & Description		
Planning Approach	Variant	Code Based	User			✓		
			Build-in	SAGT				
		Others						
	Generative	Code Based	User					
			Build-in					
		Others				Decision Table Logic		
Output Contents	Process Seq.			✓	✓	✓		
	Mill							
	Machine				✓			
	Tool				✓			
	Fixture							
	Parameters			✓	✓			
	Cutter Path							
User Supplied Logic						Decision Table		
User Editable						✓		
Commercial/Academic				Academic		Commercial		
Designer				Abou-Zeid/ Purdue	Toth et al	/CAM-I		
Comments				1975	•Not Enough Information	•Both Variant & Generative •Under Development		
References				[El Gomayel 1975]	[Toth et al. 1975]	[CAM-I 1980]		

*Interactive

to) under a contract from CAM-I. CAPP was first demonstrated and released to its sponsoring members in 1976.

CAPP (Figure 5.2) is a data base management system written in ANSI standard FORTRAN. It provides a structure for a data base, retrieval logic, and interactive editing capability. The coding scheme for part classification and the output format are added by the user. A 36-digit (maximum) alpha-numeric code is allowed. A coding scheme tailored to the user application is usually appropriate. For example, Lockheed-Georgia used a modified Opitz code for their CAPP system and achieved successful results [Tulkoff 1978]. PI-CAPP, an extension of CAPP, has its own (built-in) coding and classification system. (This eliminates the requirement of a user developed coding scheme [PII 1980].

A typical process planning session using CAPP would probably look as follows. The main menu of the CAPP system (Figure 5.3) contains 11 entries. A Header Display (HD) is used for each new plan to be created. Assuming that an arbitrary coding scheme has been implemented, Figure 5.4 shows the kind of information which would be required to create a new plan. The family number and GT code attributes (page 1 of the header menu) are used for standard plan retrieval. FS (part Family Search), MS (Matrix Search), and SA (Search Attributes) are used to retrieve process plans.

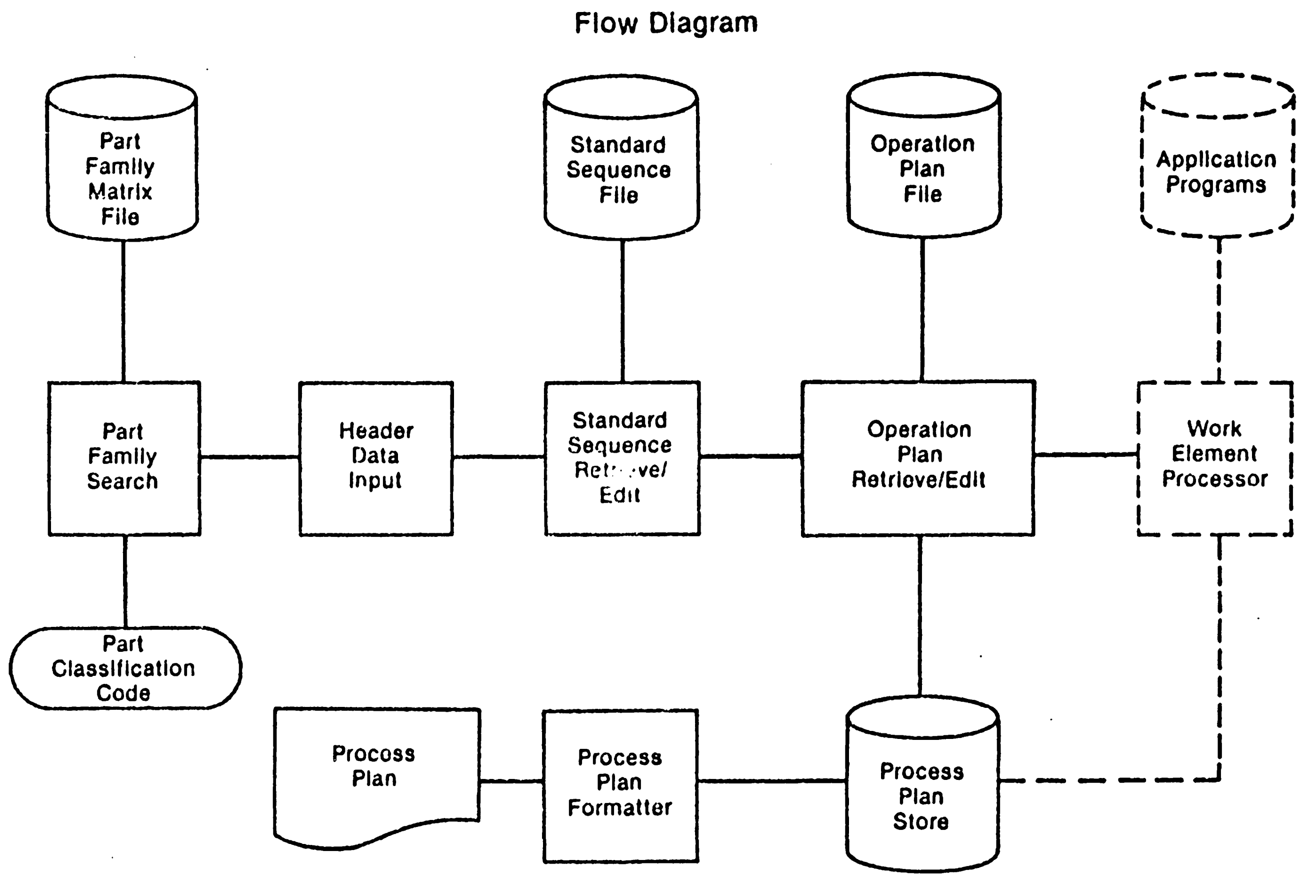


Figure 5.2 <u>CAPP System</u>

```
     FS    MS    SA    FP    DP    HD    OS    OP    PP    GT    LO
                 ** CAPP RELEASE 2.1 **
PART FAMILY SEARCH      FS
MATRIX SEARCH           MS/CLASSIFICATION CODE
SEARCH ATTRIBUTES       SA/COL. NO., VALUE(S)
FORMAT PLAN             FP/PARTNO,PLAN TYPE,STATUS
DELETE PLAN             DP/PARTNO,PLAN TYPE,STATUS
CREATE NEW PLAN         HD
RETRIEVE OPCODE SEQ     OS/PARTNO,PLAN TYPE,STATUS
RETRIEVE OP PLAN        OP/PARTNO,PLAN TYPE,STATUS
PROCESS PLAN REVIEW     PP/PARTNO,PLAN TYPE,STATUS
GROUP TECHNOLOGY        GT
LOGOFF                  LO
```

Figure 5.3 CAPP Main Menu

```
OS  SI  SC  RR  FS  DS  MM  PD  PU  FI  LO

FILL IN THE FOLLOWING FOR STORAGE AND RETRIEVAL PURPOSES

PLANNING I.D. =                          /
PLANNING REVISION NO. =                   /
PLANNING TYPE= /
FAMILY NUMBER=001/
G/T CODE=                                  /

NOW PAGE DOWN TO FILL IN THE ACTUAL HEADER DATA
```

HEADER MENU (PAGE 1)

```
OS  SI  SC  RR  FS  DS  MM  PD  PU  FI  LO

<+> <+> <+> <+> <ASSEMBLY INSTRUCTIONS > <+> <+> <+> <+>
                                                      /
PROGRAM    =                      D/M=    PA=         /
PLAN TYPE =                FAMILY NO.=    DATE=MM-DD-YY/
PART NO.  =                       REV=      PC=       /
PART NAME =                                           /
PLANNER    =                              DATE=MM-DD-YY/
CHECKER    =                              DATE=MM-DD-YY/

<+> <+> <+> < ENGINEERING REFERENCE DRAWINGS > <+> <+> <+>

ENG DWG=                 REV=                         /
EO'S=                    REV=                         /
P/L=                     REV=                         /
EO'S=                    REV=                         /
W/L=                                                  /
EO'S=                                                 /
SCH=                                                  /
EO'S=                                                 /
NEXT ASSY=                                            /
```

HEADER MENU (PAGE 2)

Figure 5.4 New Plan Creation

In Figure 5.5, a part family searching session is illustrated. The upper portion of the figure represents an imaginary part family matrix file. The code used is a 5-digit code with 5-values for each digit. After the search is completed, a standard plan can be retrieved. The standard plan is structured in a manner so that it also returns a standard sequence for the part. The sequence is called an OP (Operation) sequence in the CAPP terminology. In the OP sequence, operations are represented by OP codes which are alphanumeric codes. For each OP code in the sequence, an operation plan consists of a description of the detailed operation steps, machine, tools, fixtures, operation time, etc., all of which can be retrieved. The complete process plan is a plan with operation plans imbedded in the sequence.

CAPP also allows the user to use an existing GT system in the process plan search. This feature enables the user to make a minimum modification during the system implementation. CAPP-like systems are easy to learn and easy to use. The range of components which can be planned by CAPP is dependent on the capability of the coding scheme. Approximatedly 400 CAPP systems (1980) have been purchased through the CAM-I library or the CASA Organization of the Society of Manufacturing Engineers.

Part Family Matrix File

Family 01

0	1		1	1	1
1		1		1	1
2		1	1	1	1
3					1
4					

Family 02

0		1		1	1
1		1	1		
2	1		1	1	
3					
4					

Family 03

0		1	1		1
1	1	1		1	1
2		1	1		1
3				1	
4					

Family 04

0		1		1	1
1	1	1	1		
2			1	1	
3					
4					

Command	Explanation
(Main Menu Display)[1]	
FS[2]	Select Part Family Search
(Part Family Search Menu Display)	
MS/21120	Search for Code-21120
(Std Seq of Family)	2 Does not Match Family 01, Matches Family 02
02 Display	1 Matches Second Position of Family Matrix 02
	1 Matches Third Position of Family Matrix 02
	2 Matches Fourth Position of Family Matrix 02
	0 Matches Fifth Position of Family Matrix 02
	So, 21120 Belongs to Family 02

1. (....): display by computer
2. others : manual input

Figure 5.5 Part Family Search

5.2.2 MIAPP and MIPLAN

Both MIAPP [Hewgley 1979] and MIPLAN [Schaffer 1980, OIR 1981] were developed in conjunction with OIR (Organization for Industrial Research, Inc.). They are both variant systems, that use the MICLASS coding system in for part description. MIAPP is a data retrieval system (Figure 5.6) which retrieves process plans based on the part code. Given the code of a part, parts with similar code (user-defined similarity) are retrieved. The process plan for each part is then displayed and edited by the user. MIAPP is similar to the CAM-I CAPP system with MICLASS imbedded as part of the system.

5.2.3 APPAS and CADCAM

APPAS (an anacronym for Automated Process Planning and Selection) is a generative system for detailed process selection. CADCAM is an extension of APPAS. CADCAM operates using a CAD "front-end" to interface with APPAS. It also uses decision table logic for the process selection. The major difference between APPAS and other planning systems is in the surface description. APPAS describes the detailed technological information of each machined surface by means of a special code. A single machined surface is typically described using a data string of 30 to 40 attributes, and a

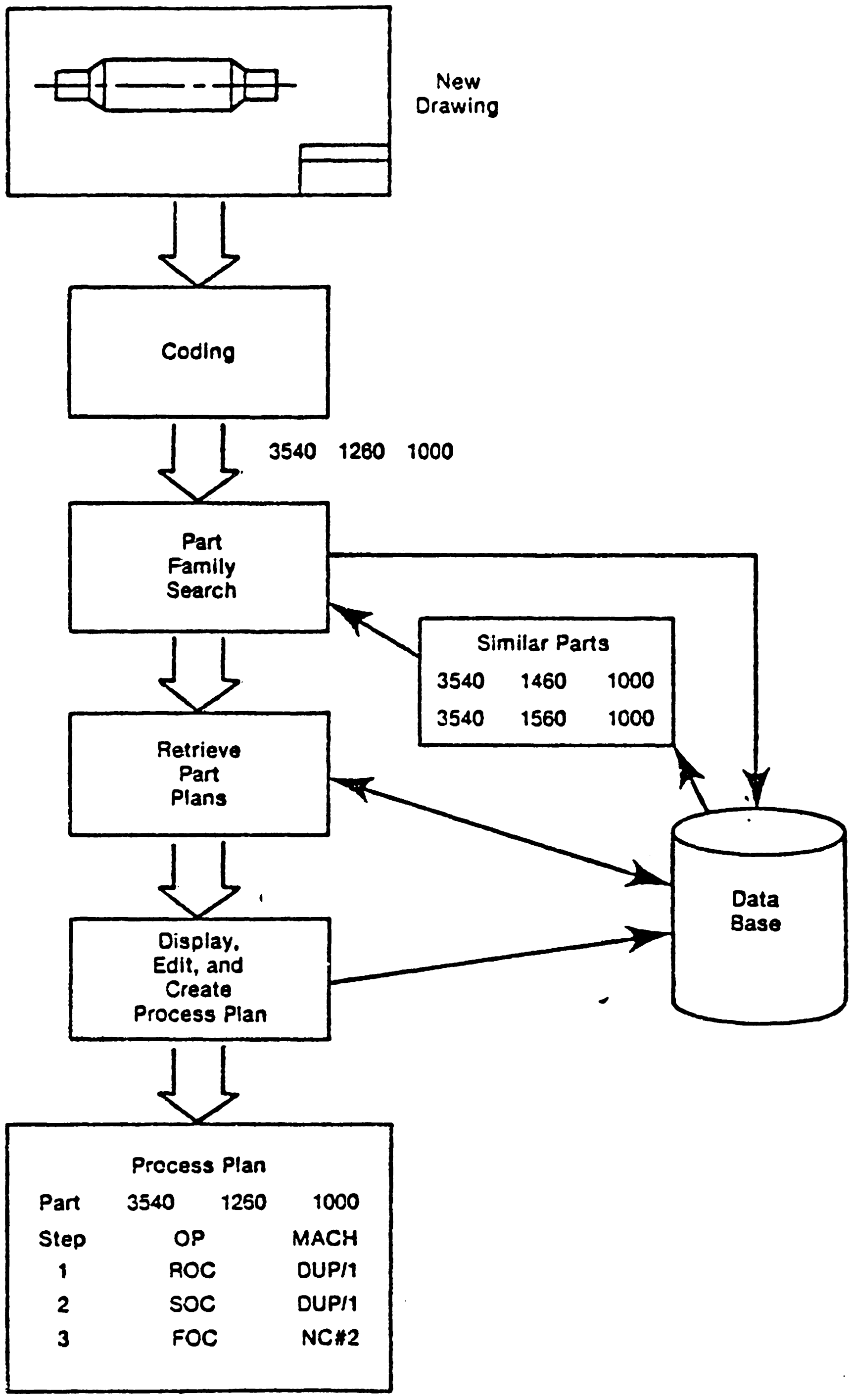


Figure 5.6 <u>MIAPP System</u>

built-in decision tree controls the decision making for the selection of detailed processes. The selection criteria are the capabilities of the individual processes, which are represented in a process boundary table (Table 5.2). The system is capable of selecting multiple passes and processes for the designated machined surface (such as: twist drill -> rough ream -> finish ream). Multiple diameter holes with special features such as an oil groove, slot, or thread can be planned as single machined surfaces. The details of APPAS include the selection of feed rate, cutting speed, diameter of the tool, number of milling cutter teeth, length of the tool, and length or depth of cut for each tool pass. The user can select the detail method i.e., use pure table look-up or optimizatize machining parameters. Time and cost estimates are also given as part of the process plans.

CADCAM provides an interactive graphics interface to APPAS. Components can be modeled graphically and edited interactively. A decision table approach also lessens the difficulty required to expand the system. An example of the CADCAM system is shown in Figures 5.7 and 5.8. Both APPAS and CADCAM are written in standard FORTRAN.

Table 5.2 Process Boundary Table

```
****************************************
 RECORD 4                          FINISH BORE
----------------------------------------
 LINKAGE ADDR:        0    0    0    0
----------------------------------------
 MINIMUM DIA :  0.375  MAXIMUM DIA : 10.000
 MAXIMUM DEPTH : 10.000 X HOLE DIA
 TOLERANCE :
    POSITIVE : .0002 X DIA ** 0.0 +.00010
    NEGATIVE : .0002 X DIA ** 0.0 +.00010
 STRAIGHTNESS : .0 (L/DIA) ** 0.0 +.00030
 PARALLELISM : .0  (L/DIA) ** 0.0 +.00050
 ROUNDNESS : 0.0003
 TRUE POSITION : .0001
 SURFACE FINISH (BEST) : 8.
```

5.2.4 AUTOPLAN and RPO

AUTOPLAN [Vogel 1981] and RPO [Tipnis 1980] were both developed by Metcut Research Associates. RPO is an installation of AUTOPLAN in the GE Aircraft Engine Group for Rotating Parts Operation. AUTOPLAN is generative only in the detailing of the part. The process selection and process sequencing level does not differ significantly from CAPP or MIPLAN. The four major modules of the system are:

(1) GT retrieval - process plan retrieval

(2) Graphical planning aides - tooling layout, verification, and work instruction preparation

(3) Generative process planning - tooling recommendations, cut recommendations, and machine tool settings

(4) Process optimization - manimum cost or maximum production rate

The unique feature of the system is the graphics interface. This provides the user with the capability to interactively view the selection and verify the plans. A GT code with decision tree logic is used for the generative process planning. AUTOPLAN took 10-12 man-years to develop (Vogel 1981), and contains some 325 subprograms and 40,000 lines of source code.

```
Process Plan for Hole:  (0.7500, 2.0000, 2.0000)

— Hole Located at Center: 0.7500      2.0000      2.0000
  Process Sequence:
    1: Twist Drig
    2: Tap

— For Hole Located at Center: 0.7500      2.0000      2.0000

Deep Hole Application     Length-Diameter Ratio = 3.00
Twist Drill This Hole
Drill Diameter = 0.242000

Feed: 0.0155        Speed: 1802.98          Time (HSS Tool): 0.0393
Spindle Stroke: 1.1002

Cost for This Step: 0.04

Ductile Material, Use Fluteless Tap

Tap Size: 0.2500        Feed: 0.10000        Speed: 456.55        Time: 0.02
Cost for Tapping: 0.02

Total Cost for This Hole: 0.06
```

Figure 5.7 Process Plan for Hole

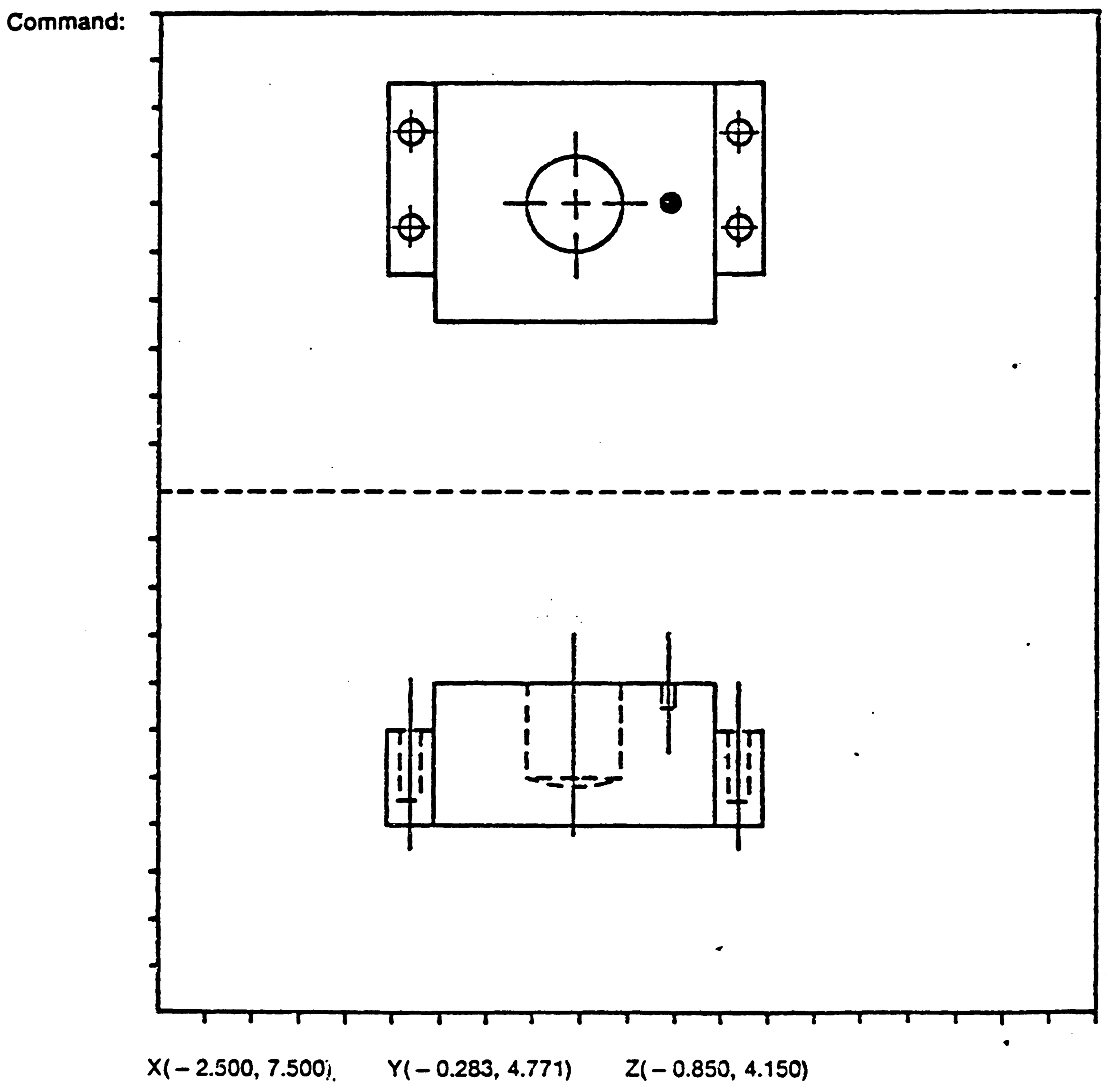


Figure 5.8 Design Model of CAD/CAM

5.2.5 AUTAP

The AUTAP system is one of the most advanced planning systems in use today. AUTAP is capable of material selection, process selection, process sequencing, machine tool selection, tool selection, lathe chuck selection (for turned parts), and part program generation. (The last three are functions of AUTAP-NC). The feature that makes AUTAP unique is its part description language. AUTAP uses primitives to construct a part, similar to a constructive solid geometry (CSG) language. A somewhat limited modeling process is used in AUTAP, and the user of AUTAP must follow the part modeling procedures using only union operations (as compared with the entire set of Boolean operators of CSG).

Figure 5.9 shows a rotational component which is described using the language. The component is decomposed into three dinstinct entities starting from the left. The codes starting with "1" (i.e. 10, 11, 12, and 13) describe the first entity, which consists of a cylinder (10), a straight chamfer (12), and a fillet chamfer (13). A unique descriptor is assigned to each feature. Therefore, any shape and technological information (i.e. geometric tolerances, etc.) can be described. The complete input includes the part description and organizational data such as part number and lot size. Currently, AUTAP can plan rotational and sheet metal parts.

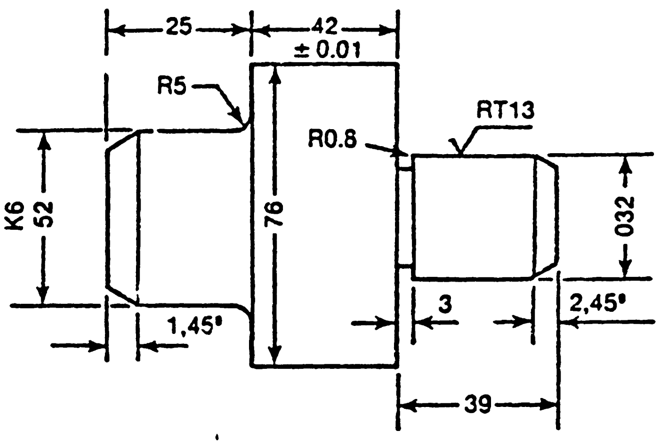
In the process planning phase of AUTAP; first, the part description is evaluated, and then the outermost contour is selected. By comparing these with the existing stock, the raw material is selected. The process selection, process sequence selection, and machine tool selection are based on decision table logic (Figure 5.10). Decision table contents (variables) are supplied by the user for his specific application.

The interface of AUTAP with AUTAP-NC can provide further planning functions, such as tool selection, fixture selection and part program generation. AUTAP-NC is capable of planning rotational parts. The final output of AUTAP-NC is an NC part program. By coupling AUTAP with the EXAPT system [Eversheim 1980], final Cutter Location DATA (CLDATA) and a verification drawing can also be obtained.

AUTAP is a system designed especially to interface with a CAD system. It can be installed as part of an integrated CAD/CAM system, and applications by several German companies have been reported.

5.2.6 CPPP

CPPP (Computerized Production Process Planning) was developed by the United Technologies Research Center, partially under US Army funding. It was designed for planning cyl-



```
10 CYLE/52,25/
11 DFIT/K,6/
12 CHAU/1,45/
13 RADIR/5/
20 CYLE/76,42/
21 LTOL/+0.01,-0.01/
30 CYLE/32,39/
31 UNCUL/3,0.8/
32 VALHE/2,13/
33 CHAR/2,45/
```

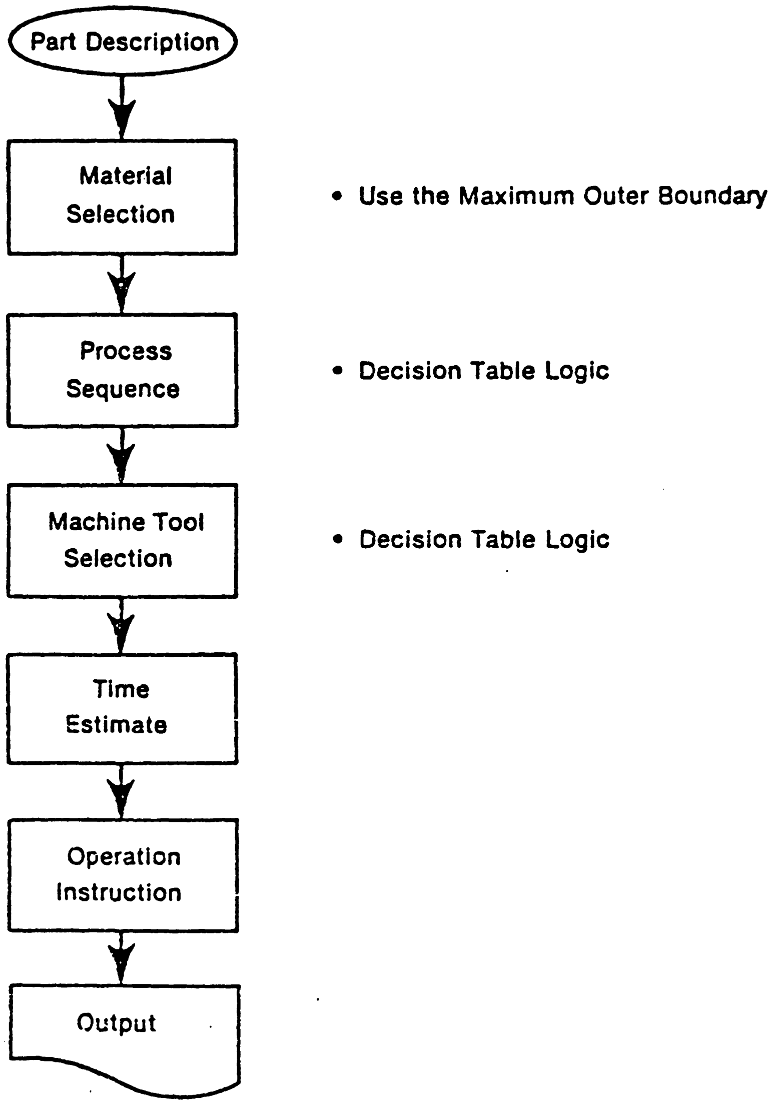


Figure 5.9 AUTAP System (taken from Eversheim 1980)

Machine Tool Selection						
		No.	1	2	3	4
Condition	300 < Length < 500	1	X	X		
	Dia < 200	2	X	X		
	Max Speed < 3000	3		X		
	Tolerance < 0.01	4	X			X
	Lot Size > 100	5		X	X	
	Fixture 123 Exist	6		X	X	
	Fixture 125 Exist	7			X	X
Conclusion	Machine 1001	1	X			X
	Machine 1002	2		X		
	Machine 1003				X	

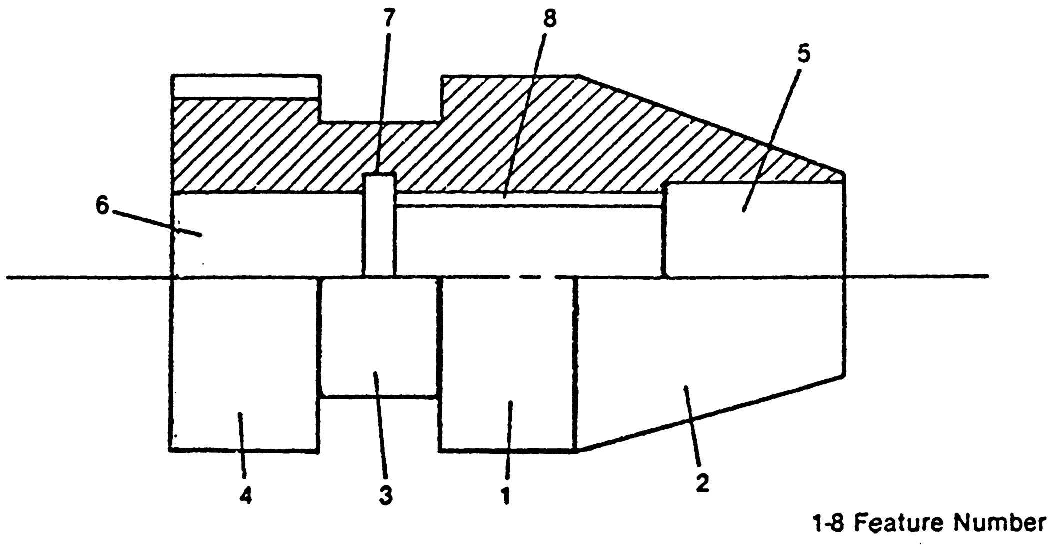
Figure 5.10 Decision Table

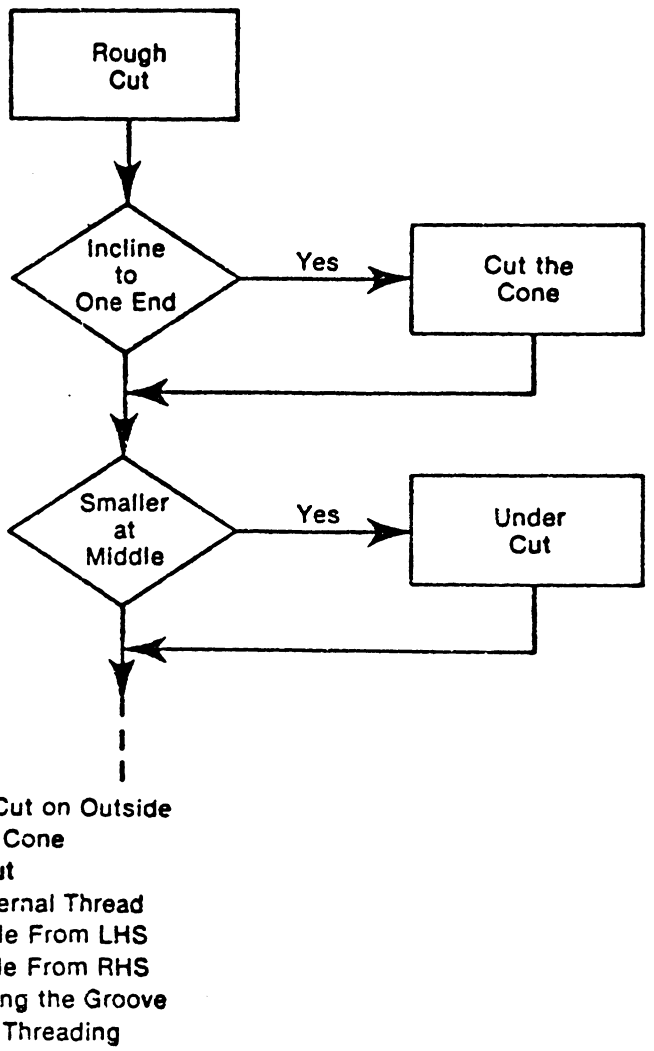
indrical parts. CPPP is capable of generating a summary of operations and the detailed operation sheets required for production. Operation sheets include sketches of the fully dimensioned workpiece with tolerances. Machine tool, cut sequences, reference surfaces, clamping surfaces, tools, and machining parameters are also specified.

The principle behind CPPP is a composite component concept. A composite component can be thought of as an imaginary component which contains all the features of components in one part family. By building a process model which contains the solution for every feature, components in the entire family can be planned (Figure 5.11). CPPP incorporates a special language, COPPL, to describe the process model. COPPL is an English-like language that can be used by manufacturing personnel with little training or computer programming experience. The following is a valid COPPL expression.

"Turn outside surface on manual lathe if surface is an open diameter (and) diametral tolerance is >= 0.002$."

Process selection and sequencing is based on the process model discussed previously. The results of selection and sequencing are combined with the part surface



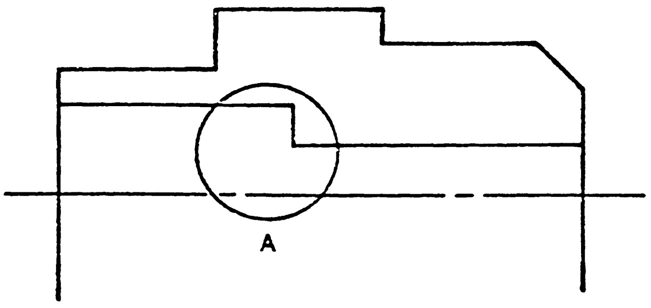


1. Rough Cut on Outside
2. Cut the Cone
3. Undercut
4. Cut External Thread
5. Drill Hole From LHS
6. Drill Hole From RHS
7. Recessing the Groove
8. Internal Threading
9. Part Off

Figure 5.11 Composite Component

	Part Surface/Feature									
Operations	1	2	3	4	5	6	7	8	9	Machines
Turn	X			X	X	X		X		Bar
Turn		X	X						X	Chucker
Grind	X									Surface Grinder
Grind					X	X				OD Grinder
Grind									X	ID Grinder
Nitride									X	Furnace
Drill							X			Drill Press
Grind									X	ID Grinder
Hone									X	Auto Hone
Total Operations	2	1	1	1	2	2	1	1	5	

Operation Matrix



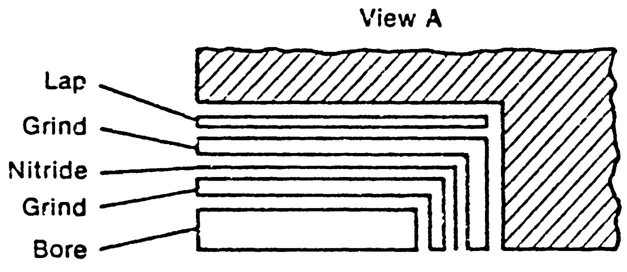


Figure 5.12 Process Steps Defined by Operation Matrix for a Part Surface

and operation data (tooling, machine tool, set up, etc.) to form an operation matrix (Figure 5.12). Cut sequence (the order of surface cuts within the same process), machine tool, and tools are selected by using the information in the operation matrix. Each cut can be displayed on a CRT screen for verification. CPPP also allows an interactive mode whereby the planner can interact with the system at several fixed interaction points.

The development of a process model is necessary for every part family. An average of 4 man-months effort is required for each model [Tipnis 1980]. CPPP is most suitable for those applications in which there are few part families, but each family member contains many variations.

5.2.7 GARI

GARI is an experimental problem solver which uses Artificial Intelligence (AI) techniques. The unique feature of GARI is the representation of planning knowledge. GARI employs a production rule knowledge base to store process capabilities. The form of each rule is:

conditions ==> pieces of advice

Each piece of advice is assigned a weighting factor to represent its importance. Since GARI is implemented in the MACLISP language, LISP notations are adapted here. A typical rule can be stated as:

```
-(>== (surface-quality &x) 6.3)
==>
(9 (roughing-cut &x))
```

which means:

"If the surface quality of an entity "&x" is higher (poorer) than 6.3, then one is advised (with a weight equal to 9) to avoid a roughing cut for &x."

The part being planned is described using a special notation. The notation includes a maximum of twenty different attributes which represent shape, tolerance, surface finish, etc. An example can be shown:

Let H be a hole to be described. S1, S2, and S3 are defined surfaces for H:

```
(H (type countersunk-hole)
(starting-from S1)
(opening-on S3)
(diameter 6)
(countersink-diameter 12)
(surface-quality 7))
(line feed space)
(distance H S2 19)
(countersink-depth H S1 3 + -80)
(perpendicularity H S3 + -50)
```

In a manner similar to the medical consultation counterparts (e.g. MYCIN [Shorliffe 1976], a search is used to find the goal -- plan the part. When a different solution (contradic conclusions from different rules) is detected, the system is able to discard certain pieces of advice in order to resolve the problem. This kind of system can be developed by consulting expert planners and adding new rules to the system. This, in fact, is the major advantage of rule-based systems. Because other functions of process planning such as machine, tool, and fixture selection are more routine, and parameter optimization is more mathematical, AI is not appropriate tool for those functions. GARI does not try to plan these functions; only process selection and process sequencing can be planned with this system. No practical application of GARI has been reported.

Chapter VI

TIPPS - AN INTEGRATED PROCESS PLANNING SYSTEM

Thus far, several topics related to process planning have been discussed. The input for process planning (engineering design or CAD), process capabilities, approaches to process planning and information decision systems have all been discussed in significant detail. Although all of these topics are closely related to process planning, an integrated approach to generative process planning has yet to be presented. Unfortunately, this has also typified much of the research on process planning both in the U.S. as well as abroad. In this chapter, a unified approach to process planning will be presented.

The discussion in this chapter focuses on a structured approach to process planning. A process planning system called TIPPS (Totally Integrated Process Planning System) is presented in the chapter. The organization of TIPPS, its structure, its software and its hardware are discussed. Some examples of components planned using TIPPS are also presented.

6.1 TIPPS - AN OVERVIEW

TIPPS is a generative process planning system that has evolved from the APPAS [Wysk 1979] and CADCAM [Chang 1981] systems. TIPPS is one of the few (perhaps the only) systems that integrates CAD and generative process planning into a unified system. In TIPPS, the logical divisions of process planning are broken into functional modules (See Figure 6.1). The modules within the dashed lines in Figure 6.1 have already been implemented as TIPPS software. However, even this software is small when compared to the "Total Manufacturing Planning System" shown in Figure 1.11.

TIPPS input comes from a CAD data base. In TIPPS, an object (component) is represented by its boundaries (boundary model). Faces are defined by their limiting edges, and these faces form the boundaries of the component. Dimensions can be measured from the edges, and technological information such as size, tolerance and surface finish are also included in the data base and are linked to corresponding faces. The model of an object contains the information necessary for display as well as for process planning. The format for TIPPS input will be discussed later. Currently, TIPPS is confined to box-shape components with holes, and machined surfaces which can be approached only from the top of the part. Graphical displays can be in either of two

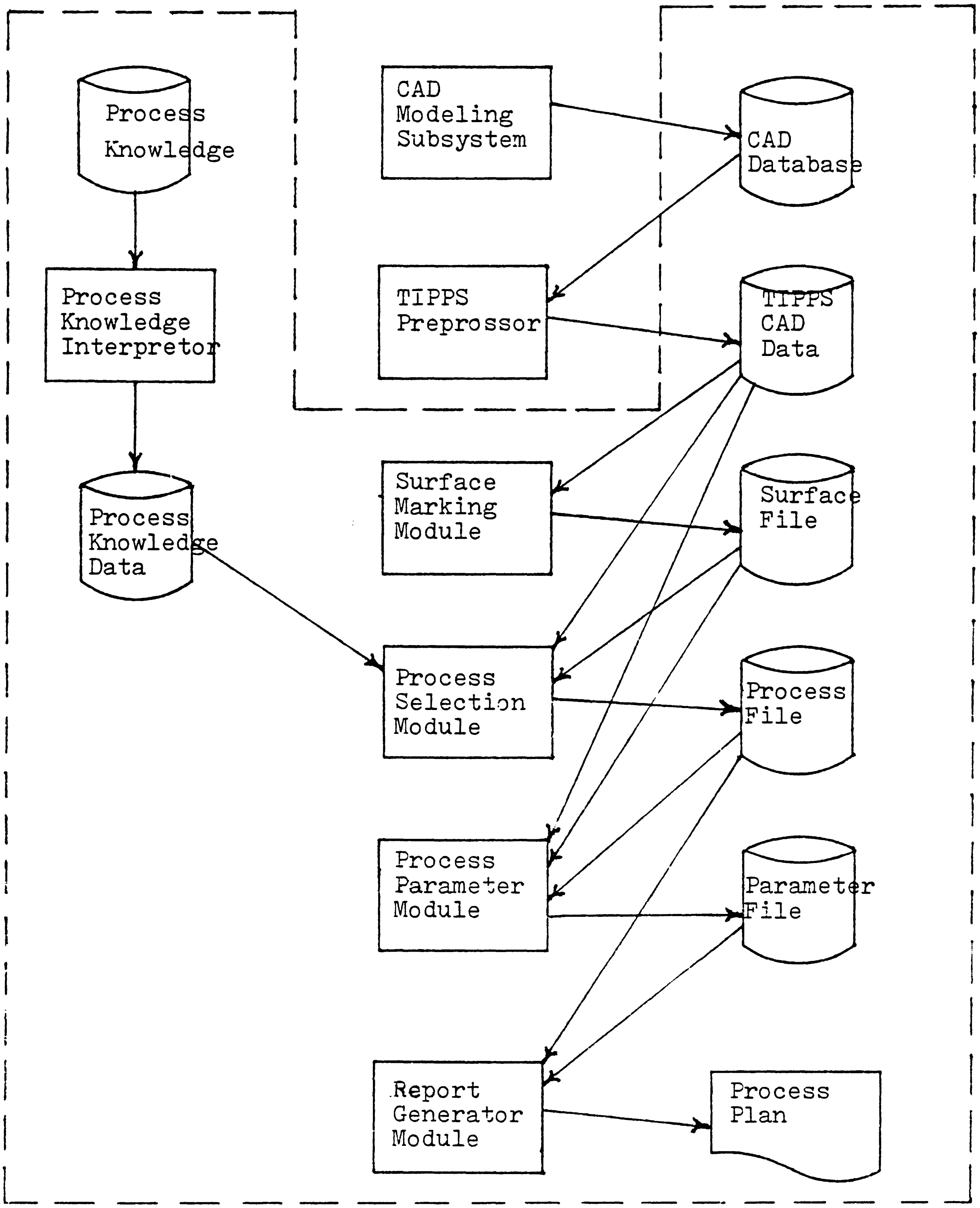


Figure 6.1 TIPPS Structure

forms: 2-D or 3-D (without hidden line removal). Figure 6.2 illustrates these two drawing modes.

In TIPPS, surfaces which require machining are identified by an operator, designer or process planner. (TIPPS provides a tool (cursor) to identify (mark) surfaces interactively). The descriptive information relevant to the marked surface is stored in a surface file which contains surface codes, as well as the faces corresponding to each surface and surface relationships. TIPPS also allows the user to modify the surface file. Figure 6.3 shows a graphical display of different mark surfaces.

The process selection module uses information from the surface file and CAD data base to select appropriate processes. The process knowledge which represents the capabilities of a process is described by using a simple description language - PKI. The description is translated into a link list data file -- process knowledge data. When the process selection module is invoked, it runs as a language interpreter in which the process knowledge data becomes the program. Backward planning and recurrent calls are used in the process selection phase. A feasible sequence of processes and tools can be selected for a component (Figure 6.4). The resultant processes are saved in a process file.

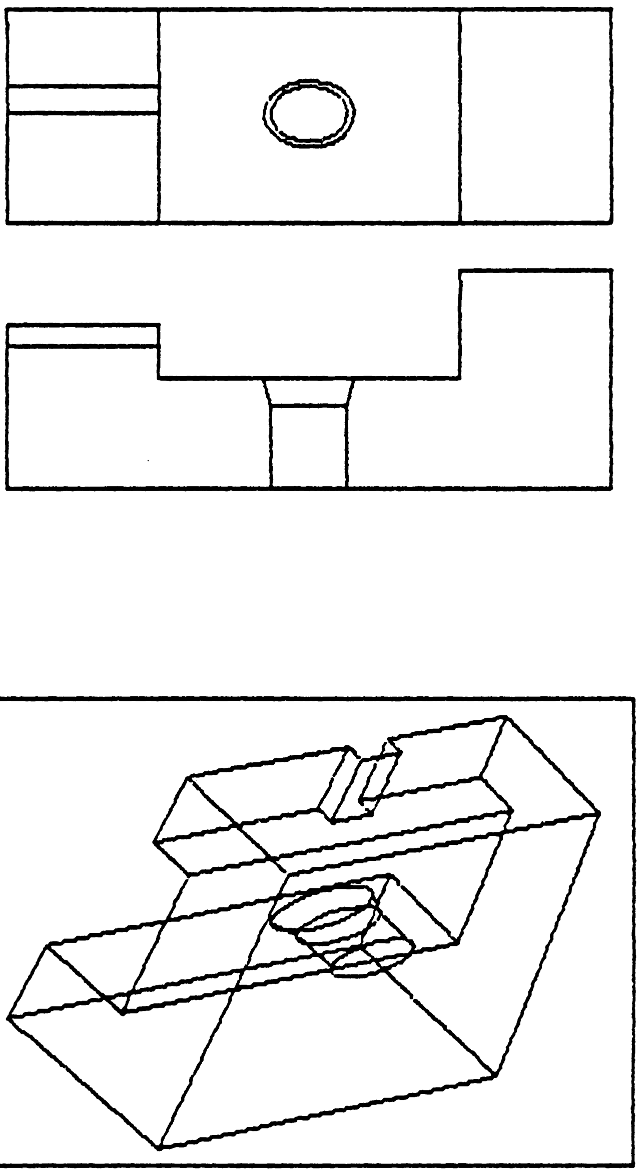
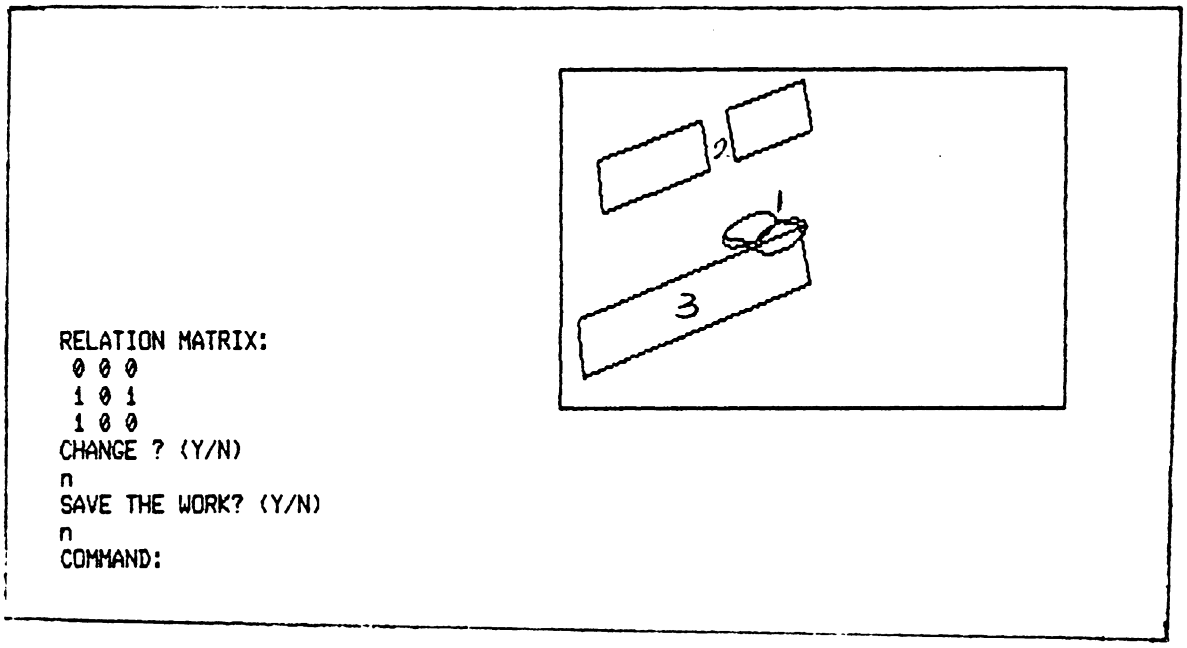


Figure 6.2 <u>2-D and 3-D Display of a Component Using the TIPPS Planning System</u>



```
RELATION MATRIX:
 0 0 0
 1 0 1
 1 0 0
CHANGE ? (Y/N)
n
SAVE THE WORK? (Y/N)
n
COMMAND:
```

Figure 6.3 Surface Identified (S_1, S_2, S_3)

```
COMMAND:
proc

 ****  SURFACE :  2 ****

PROCESS:  R F MILL

 ****  SURFACE :  3 ****

PROCESS:  R F MILL

 ****  SURFACE :  1 ****

PROCESS:  TWST DRL
PROCESS:  RGH BORE
COMMAND:
```

Figure 6.4 Process Selection Information from TIPPS

Process parameters (feed and speed) are determined by the process parameters module. A table look-up procedure is currently used in selecting appropriate parameters. Input to this module includes material type, hardness, tool material, tool diameter, surface shape and dimension.
The output are feed, speed and machining time. (Although the module is capable of finding optimum paramters).

Finally there is a report generator which prints the final process plan in a human readable format. Figure 6.5 shows a printout of a process plan which contains processes, tools, feeds, speeds and machining time.

TIPPS is a research-based system which is by no means complete; however, it does demonstrate how a generative process planning system can be developed. It also provides a structure for future expansion.

6.2 TIPPS DESIGN PHILOSOPHY

The major concept used in designing TIPPS is to construct a generic framework for a modular system. The system can interface with a CAD system and can be expanded easily because of the structure of its process knowledge and functional capabilties. A partial list of some important attributes of TIPPS includes:

1. It has a modular structure,

```
+------------------------------------------------------------------------------------------------------------------------+
|                                                                                                                        |
|                                                                                                                        |
|                                              P R O C E S S   P L A N                                                  |
|                                                                                                                        |
|                                                                                                                        |
| PART NAME: BRACKET TWO                                                          PART NUMBER: B-10002                   |
| MATERIAL:            2                                                          PLANNER:     T. C. CHANG               |
|                                                      DATE :AUGUST 13, 1982                                             |
|                                                                                                                        |
+------------------------------------------------------------------------------------------------------------------------+
| OP |    SURFACE     |    L     |    D     |          | PROCESS  |   TOOL   |   FEED   |  SPEED   |   TIME   | REMARKS|
| NO |  ID   | TYPE   |    L     |    W     |    H     |          |   DIA    |  (IPR)   |  (RPM)   |  (MIN)   |        |
+------------------------------------------------------------------------------------------------------------------------+
|    |       |        |          |          |          |          |          |          |          |          |        |
|  1 |   2   |  FLAT  |  4.000   |  2.000   |   0.0    |R F MILL  |  2.040   | 0.02091  |  147.34  |   1.30   |        |
|    |       |        |          |          |          |          |          |          |          |          |        |
|  2 |   3   |  FLAT  |  4.000   |  2.000   |   0.0    |R F MILL  |  2.040   | 0.02091  |  147.34  |   1.30   |        |
|    |       |        |          |          |          |          |          |          |          |          |        |
|  3 |   1   |  HOLE  |  1.000   |  0.995   |   0.0    |TWST DRL  |  0.995   | 0.02139  |  204.25  |   0.23   |        |
|    |       |        |          |          |          |          |          |          |          |          |        |
|  4 |       |        |          |          |          |RGH BORE  |  1.000   | 0.00412  |  124.10  |   1.95   |        |
|    |       |        |          |          |          |          |          |          |          |          |        |
+------------------------------------------------------------------------------------------------------------------------+
|                                                                        TOTAL PROCESSING TIME :      4.78 MINUTES     |
|                                                                                                                        |
+------------------------------------------------------------------------------------------------------------------------+
```

Figure 6.5 TIPPS Process Plan

2. It can interface with a CAD system,
3. It allows for interactive surface identification, and
4. It contains a process/knowledge description language.

6.2.1 <u>Modular Structure</u>

Generative process planning is a complex task. If one was to try to solve the entire planning problem as a whole, it would be too difficult (if not impossible). The system would also be complex to understand. Expansion or modification of the system would be extremely difficulty. However, using a modular approach to docompose the tasks and construct modules for each decomposed subtask (function) makes generative planning achievable. Each module uses its input to create output, and is functionally separate from the other tasks. However, through data input and output, modules can communicate with each other and achieve total process planning.

Since each module contains a decomposed subtask; thus, the planning tasks become more meaningful and are more easily understood. Expansion and modification of the system is possible by simply modifying modules. Interactive modules can be implemented when automatic planning modules do not exist or are difficult to develop. After a solution technique for a missing function is developed, the corresponding modules can be changed easily to interface with other modules.

The major functions (modules) of process planning can be listed as:

1. Material selection

2. Process selection
   - surface identification and classification
   - design information retrieval
   - candidate process selection

3. Machine selection
   - candidate machine extraction
   - machine selection

4. Tool selection
   - candidate tool extraction
   - tool selection

5. Intermediate surface determination

6. End effector selection

7. Fixture selection
   - fixture model retrieval

fixture verification

8. Process sequencing

9. Machining parameters selection
   model selection
   optimization

10. Cutter path generation
    APT program generation
    CLDATA generation

11. Cost and time estimation

12. Process plan preparation
    resource list - tools, fixtures, machines, end effectors

13. Data file maintenance
    process boundary file
    machine capability file
    tool capability file
    end effector file

14. Plan verification

Each function listed can be constructed as a software module. Information requirements for each function are shown in Table 6.1.

TABLE 6.1

Functional Relationship

A_m	: Material availability.	A_{mc}	: Machine availability.
A_t	: Tool availability.	A_f	: Fixture availability.
C_m	: Cost of material.	C_p	: Process cost.
C	: Total cost.	CL	: Cutter path.
D	: Dimensions.	E	: End effector selected.
F	: Fixture selected.	G	: Geometrical shape.
I	: Intermediate surfaces.	P_m	: Machining parameters.
M	: Material selected.	M_c	: Machine selected.
P	: Process selected.	S	: Process sequence.
S_f	: Special features.	T	: Tool.
T_m	: Time.	Tol	: Tolerances.
W	: Weight.		

$M = f_1 (G , A_m , C_m)$

$P = f_2 (G , D , Tol , S_f , C_p)$

$M_c = f_3 (P , A_{mc})$

$T = f_4 (P , D , M , A_t)$

$I = f_5 (G , T , CL)$

$E = f_6 (G , D , W)$

$F = f_7 (G , P , M_c , I , A_f)$

$S = f_8 (P , M_c , T , F , E)$

$P_m = f_9 (G , Tol , C , P , T)$

$CL = f_{10} (G , P , T)$

6.2.2 CAD Interface

As CAD becomes more and more efficient in the field of engineering design, more and more design work will be done using a CAD system. It follows that the input to a process planning system will be either a hard copy prepared by a CAD system or a geometric model in a CAD data base. In either case, a CAD model for the component will be resident in (or can be obtained in a straightforward manner from) the CAD system. It would be wasteful to have a process planner interpret a 2-D or 3-D display since the interpretation can be time consuming and prone to interpretation error. The effort to translate a design display to a format acceptable by the process planning system can be more time consuming than preparing the process plan. This process can be likened to a non-Japanese speaking American foreman in a Japanese factory, where every order would require the service of an interpretor.

TIPPS links directly to data in a CAD data base. Component models in any CAD system can be postprocessed into a TIPPS (or other planning system) compatible format. The TIPPS data (an internal representation) is a legal CAD data representation. Furthermore, TIPPS allows the user to manipulate both 2-D and 3-D displays interactively. Another function of TIPPS is that it directly retrieves dimensions and technological information from a CAD data base. This not only reduces the time required for preparing process planning data input, but also reduces the likelihood of errors.

The CAD interface for process planning is a small step toward closure of the gap between CAD and CAM. The ultimate goal in CAD/CAM is to have an integrated system in which the same data is used throughout the system. No "out of flow" data preparation is required.

6.2.3 Interactive Surface Identification

Surfaces which require machining are normally indicated on engineering designs. It is a standard practice to put surface finish symbols or machining indicators on those surfaces that need special care. It is impractical to assume that one can design a system which will determine which surface(s) need to be machined automatically. Two major reasons lead to this conclusion. The first is the technical difficulty required. The machining of a surface depends on the design as well as the workpiece selected. It is not clear how one determines the differences between a design and workpiece automatically using a computer. Also, given the same design, machined surfaces are definitely different for casting and slab workpieces. The second reason that makes automatic identification impractical is the efficiency of such a system. Even if the first issue could be resolved, it is again necessary to construct a model of the workpiece. The latter is another additional task which may require substantial effort.

In TIPPS, interactive surface identification is used. A menu driven identification subsystem with graphics capability is used to identify machined surfaces. Since a human has a well developed pattern recognition ability, TIPPS asks the user for the surface shape and faces included as part of the surface. The latter process is accomplished by using a crosshair cursor. At this stage, it provides a convenient way to identify machined surfaces. The user's responsibility is to select an item (shape) from a menu and locate the faces of a surface with a cursor. This reduces both the surface identification process and programming task.

It is conjectured that, in the future, a pattern recognition subsystem will be capable of identifying machined surfaces. However, the interactive approach employed in this research is both feasible and sound.

6.2.4 Process Knowledge Description Language

Since process knowledge is manufacturing system dependent, it is necessary to maintain flexibiliy in modifying process knowledge. The modification should be easy, so that it will not prohibit changes necessary to the system. The best approach is to provide a language to describe process knowledge, because a descriptive language can provide maximum flexibility. However, such a descriptive language should be constructed so that it is easy to use and isolated from other system programs.

In TIPPS, a description language, PKI (Process Knowledge Interpretor language), is provided. It follows the general format of an IF --- THEN --- clause. A user can describe process boundaries using reserved system words. Each reserved word is defined in the system program. PKI either fetches a piece of data from one of the system data files, stores a piece of data, or performs some computation. A user is exempt from knowing the system program and data file structure; yet, he is able to describe a process precisely.

6.3 TIPPS DATA STRUCTURE

In the following sections, implementation of the previously mentioned philosophy will be discussed. First the data structure of TIPPS system will be explained.

The data structure is the most fundamental element of storing information. It is most important to TIPPS, since data files interact with the modules used to produce process plans. Data files in TIPPS include:

1. CAD data,
2. Surface data,
3. Process data,
4. Parameter data, and
5. Process knowledge data.

6.3.1 CAD Data

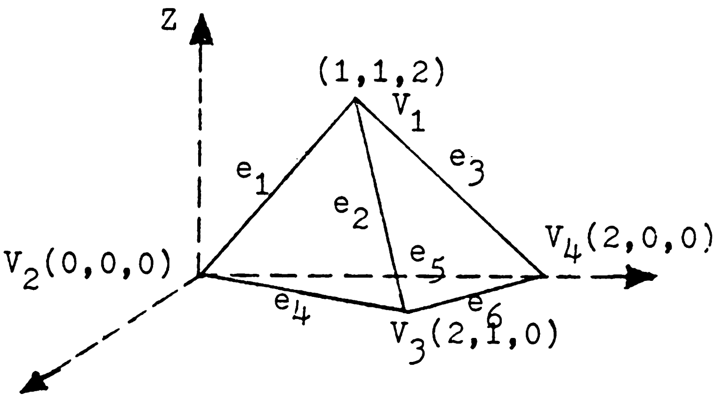
In TIPPS, there are two types of CAD data. One type of input is the geometric CAD drawing. This data is in a format which allows data exchange with other CAD systems. The CAD drawing data can be stored on standard 80 column cards or another sequential data medium such as a card image on a disk. Each card image is called a record, and in each record, the first item is a code which specifies the record type (Table 6.2). The second item is the object, i.e. face number, edge number, vertex number, etc. It is used to identify the object in the record. The remainder of items in the record either represent a data value or identifications of objects associated with the current object.

Figure 6.6 represents the input data for a tetrahedron. The first line of the data contains the leader (the part name and part number). The line beginning with 100 has a material code 20 (carbon steel) and a Brinell Hardness Number (BHN) 250. Column one entries containing "110" indicate faces (face one consists of edges 2, 3 and 6). A minus sign before an edge identification indicates that the direction of the edge is reversed. The direction of edges follows a right hand rule to point to the interior of a part. "210" records contain vertex lists, i.e., edge 1 consists of vertices one and two. Finally the "310" records contain the coordinates of a vertex. As long as all digits are defined, the sequence is not important.

It is obvious that the above data structure is not efficient for graphics, i.e., it results in a lot of unused space and requires interpretation when used. However, it provides an easy way to communicate with other systems. In order to eliminate the inefficiency, an internal data structure is used. The internal data structure requires less storage space and is much easier to manipulate.

Table 6.2 Input Data Record Type Code

Code	Type
100	Material
110	Plane face
120	Cylindrical face
130	Hole
140 ⋮ 200	Reserved
210	Edge
220	Circle
230	Spline
240 ⋮ 300	Reserved
310	Vertex
320 ⋮ 490	Reserved
500	Technological information

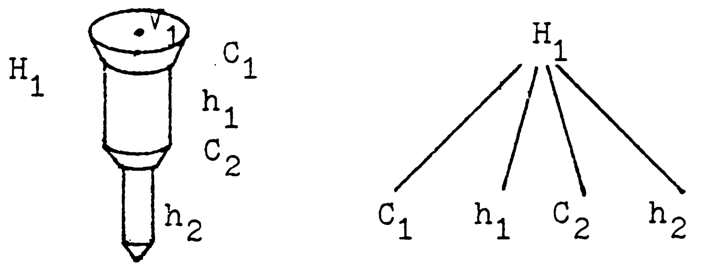


```
Tetrahedron                P-1000
100     20    250
110     1      -2     3     -6
110     2      -1     2     -4
110     3       1     5     -3
110     4       4     6     -5
210     1       1     2
210     2       1     3
210     3       1     4
210     4       2     3
210     5       2     4
210     6       3     4
310     1       1     1      2
310     2       0     0      0
310     3       2     1      0
310     4       2     0      0
999
```

Figure 6.6 <u>TIPPS Input Data Format</u>

Figure 6.7shows an example of the internal data structure of TIPPS. Because TIPPS uses a boundary model, there are four arrays in the example (more arrays are used for other geometries). PFACE stores pointers of the face information in the FACE array. It is necessary because face information has a variable data record length, i.e. three edges for triangular face, four edges for rectangular face, etc. In the FACE array, the first item in a record is the number of edges, the remaining items are pointers to the EDGE array. The EDGE array is a two dimensional array which stores pointers of the terminal vertex. The VERTEX array is also a two dimensional array which stores data values for the vertex (x, y, z) coordinates.

Hole data is stored using a different format. The record length of a hole segment is fixed. In order to store multi-diameter holes, a link list is used (See Figure 6.8). The HOL array is similar to PFACE in that it locates the position were hole data is stored. The HOL array is a link list which stores the hole information. Each record in HOL has a fixed format, i.e., it either stores a chamfer or a hole segment. Records are linked to accomodate any hole/record length.



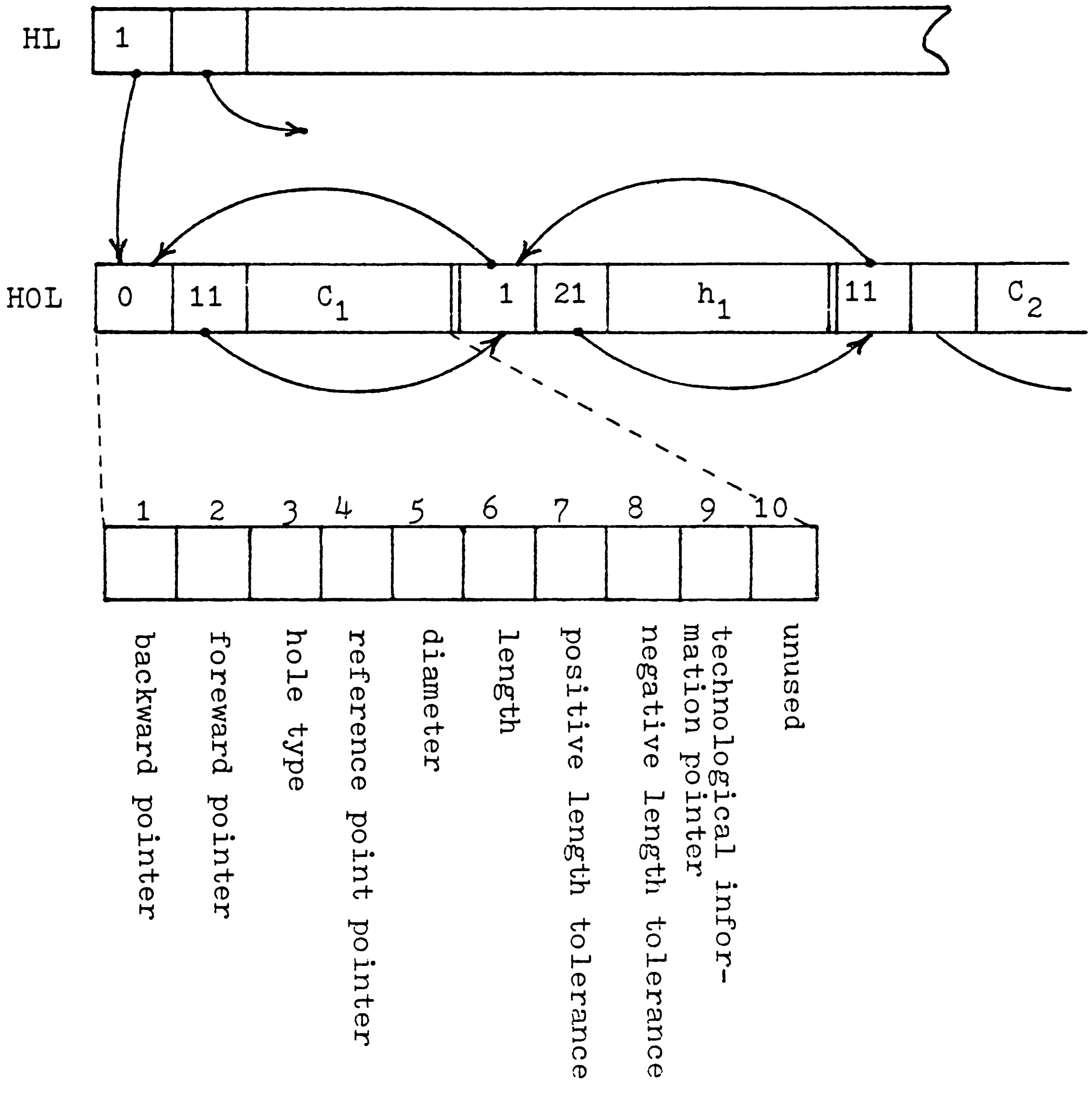
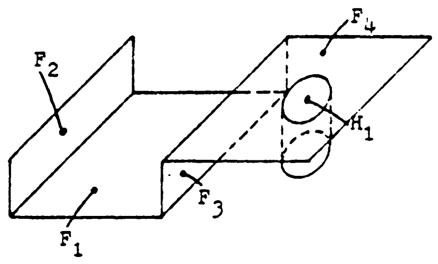


Figure 6.7 <u>Hole Data Structure</u>

6.3.2 Surface Data

A surface identification is used to identify a machined surface. A machined surface can be a hole or a set of faces. For example, a straight slot is a machined surface, since it can be cut by a milling cutter or a shaper. Such a slot consists of a minimum of three faces -- a bottom face and two side faces. A surface data record contains a code which shows the surface type and pointers which link the faces and hole together to form the surface. A double link list is used to store surface data (Figure 6.8), since a variable number of faces is expected for each surface. Such a data structure also provides flexibility for surface identification modification.

Figure 6.9 illustrates the data structure. IRSUR is a relationship matrix. IRSUR (i,j) = 1 implies surface i must be machined before surface j. This matrix is used to define the cut sequence. IPSUR is a pointer array, whose first row stores surface type, i.e. 111 - hole, 203 - straight slot, and 201 - flat. The second row of IPSUR stores the pointers which locate surface information in the NSUR array. Each record in the NSUR array has four items, forward and backward pointers, face type code and pointers to the face.



$S_1 = \{H_1\}$
$S_2 = \{F_1, F_2, F_3\}$
$S_3 = \{F_4\}$

$$\begin{array}{c|ccc|} \text{IRSUR} & 0 & 0 & 0 \\ & 0 & 0 & 0 \\ & 1 & 1 & 0 \end{array}$$

Relationship matrix
(S_3 must be done before S_1 and S_2 can start)

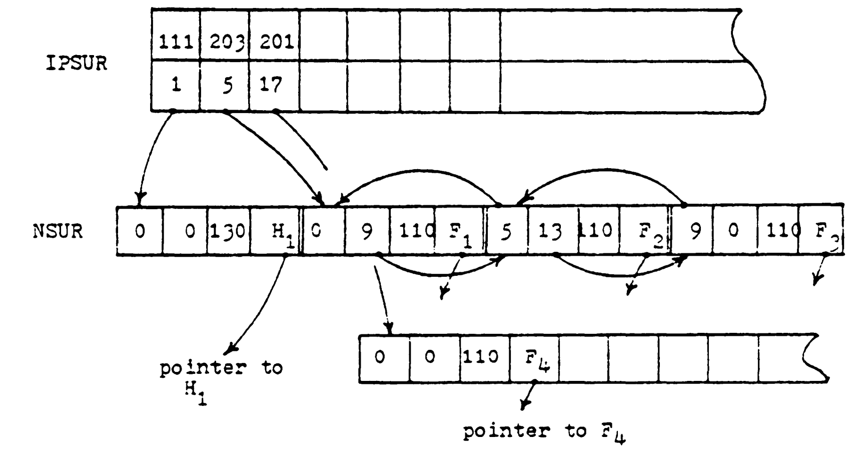


Figure 6.8 Surface Data Structure

6.3.3 Process Data

Process selection is generated by the system. A variable number of processes can be used to machine each surface. Information for the selected process has a fixed length process type and tool size needed in case of hole making processes. A pointer array and a double linked list are used (Figure 6.10). The double link list stores process code and tool size. In the figure, F.B. indicates finish bore, R.B. - rough bore, and DRL - drill. Multiple processes can be linked together for each surface.

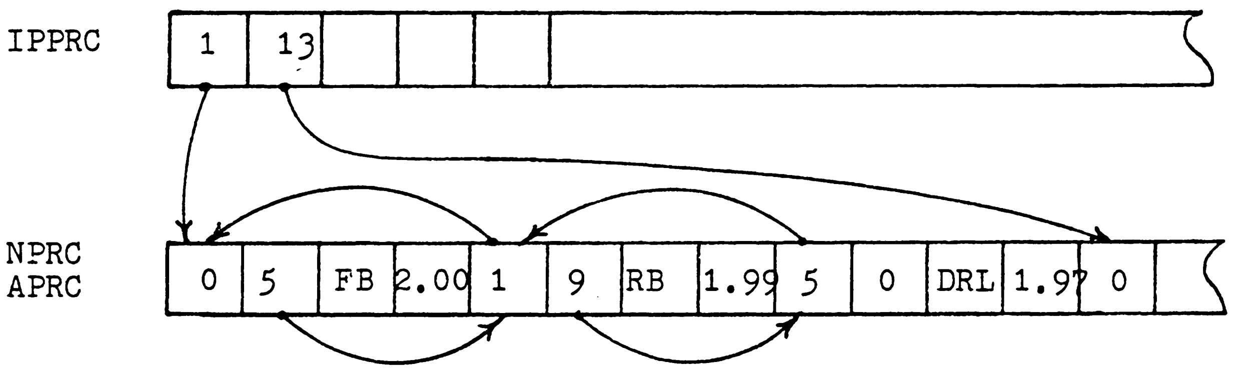
6.3.4 Paramter Data

Parameter data includes operational information for a machining process. Included in this data are: feed (f), speed (V) and machining time (t). The data structure for parameter data is also shown in Figure 6.9.

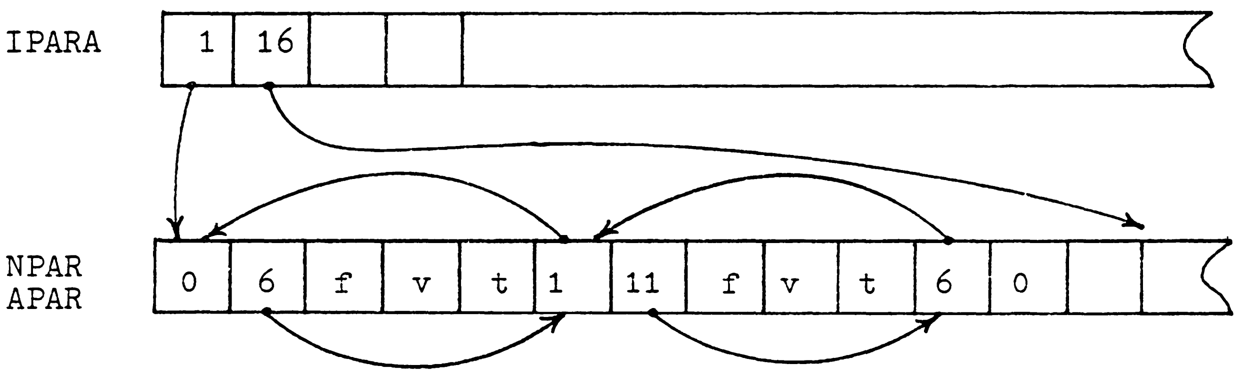
6.3.5 Process Knowledge Data

Process knowledge in TIPPS is used as a basis for process selection via a decision table methodology. The data structure is organized so that it can be used as an interpretor language.

A link list and a data array are used to store process knowledge (Figure 6.11). In Figure 6.10, a statement ((IF



Process Data



Parameter Data

Figure 6.9 Process Data and Parameter Data Structure

(Shape ! 111 =)) (THEN (8 process)) is stored in the list. Each record in the list contains two elements. They can be either pointers (when they are positive), or a pointer and a variable or a data value (negative). Variables such as IF, THEN, SHAPE, etc. are represented by a code with a negative sign, and the data value 8 is represented by a negative pointer which points to the data array. This arrangement reduces confusion between pointers in the link list and variables or data values.

6.4 PKI - A PROCESS KNOWLEDGE INTREPRETER LANGUAGE

The heart of the process selection mechanism is the process knowledge data base. A process knowledge description language PKI (Process Knowledge Interpretor) is used to define process capabilities. Although it is only used to define process capabilities, it also has the potential to describe machine and tool capabilites. PKI uses a simple language structure in which every process is defined by a statement. A statement is written in an "IF --- THEN ---" form to specify the required conditions and actions. Conditions are "pre-conditions" of a machined surface to be planned. Actions output the process selected and change the surface conditions, to reflect the result of a machining process.

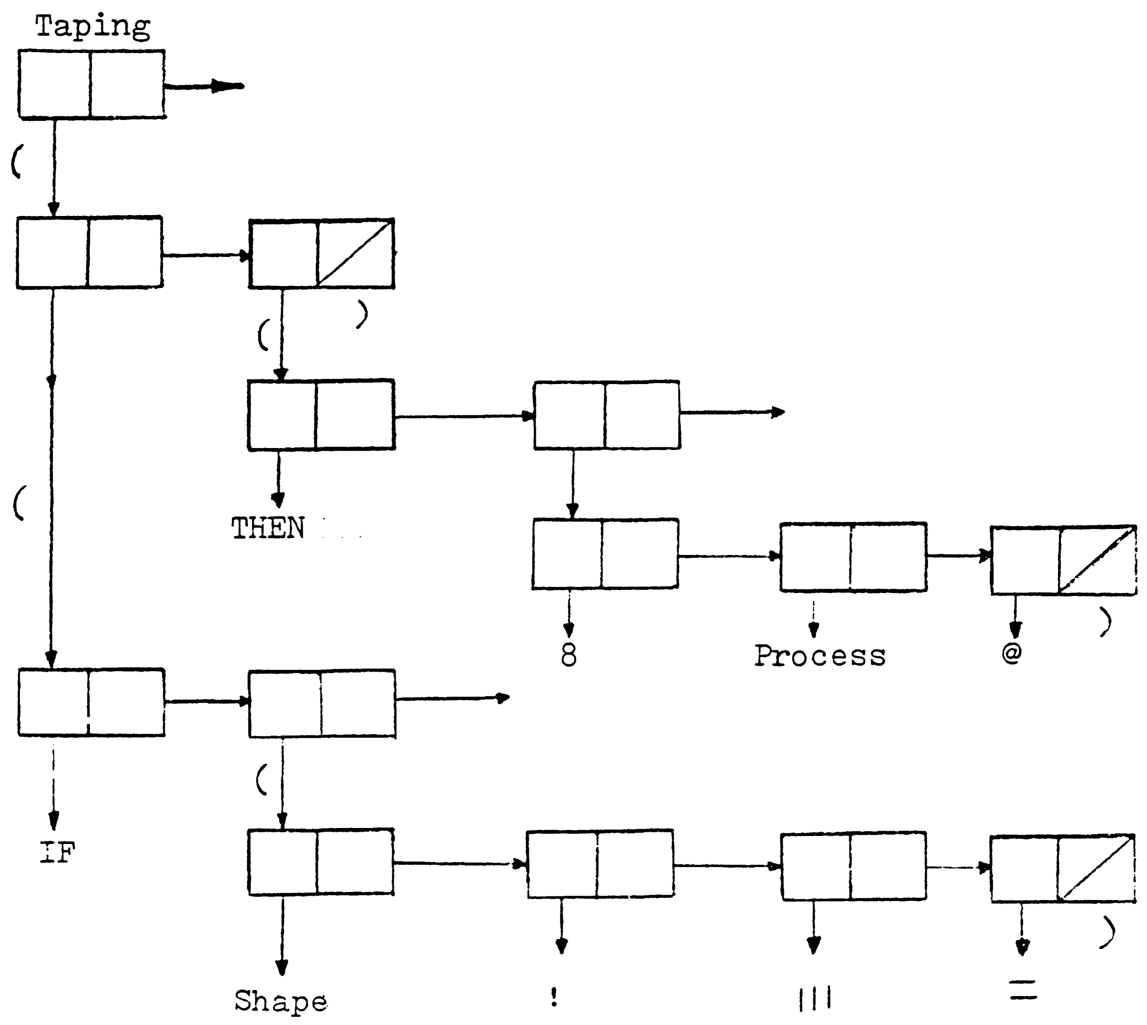


Figure 6.10 Data Structure

Describing process capability with PKI is very simple, yet effective. PKI consists of a vocabulary of words necessary for process knowledge description. A user does not have to know anything about the interface with TIPPS. With minimum training, a user should be able to interpret and modify a process knowledge base written in PKI. In the following sections, the syntax of PKI as well as an example will be shown.

6.4.1 PKI Syntax

The basic element in PKI is a statement. The entire data base is constructed using these statements. A PKI statement has the form:

```
((IF
   (expression 1)
   (expression 2)
         .
         .
         .
   (expression n)
 )
 (THEN
   (expression n + 1)
   (expression n + 2)
         .
         .
         .
   (expression m)
 )
)
```

where

```
<expression>:  = <atom> <atom> --- <atom>

<atom>:  = <literal atom> | <numerical>
```

<literal atom>: = <system variable> | <operator>

<numerical>: = integer or real number

<system variable>: = DIA, LENGTH, TLP, TLN, etc.

<operator>: = <mathematical operation> | <stack operator>

<mathematical operator>: = t, -, *, /, **, =, >, <,
>=, <=, ><, NEG, ABS

<stack operator>: = @ , !, DUP, SWAP, ROT

(:= equal, | or)

Expressions 1 to n (in the IF portion) are conditional expressions which can either be true (T) or false (F). When used in a single expression, several conditions can be specified. As a single expression these conditions are "ORed" together. Expressions 1 to n are then "ANDed". In the THEN portion, expressions do not take any truth value. Instead, they are actions given to a true condition. Condition expressions always have at least one condition operator (=, >, <, >=, <=, ><). However, action expressions do not.

Expressions use post-fix notation (an operator follows the operands). A complete set of available literal atoms is shown in Table 6.3.

During the evaluation (execution) of a PKI statement, stack arithmetic is used. Expressions are evaluated separately. An atom is pushed onto the arithmetic stack, if it is a numeral atom. The address of an atom is pushed onto

Table 6.3 PKI Operators and System Variables

Code	Reserved Words	Meaning	Code	Reserved Words	Meaning
-1	>	greater than	-27	roundness	
-2	<	smaller than	-28	angularity	
-3	+	addition	-29	parallelism	
-4	-	subtraction	-30	straightness	
-5	*	multiply	-31	shape	
-6	/	divide	-32	SF	surface finish
-7	=	equal	-33	width	
-8	>=	great equal	-34	height	
-9	<=	less equal	-35	then	
-10	**	exponent	-36	IF	
-11	><	not equal	-37	feed	
-12	@	save	-38	speed	
-13	!	fatch	-39	proc	process
-14	NEG	negation	-40	free	terminate
-15	ABS	absolt. value	-41	DTL	tool diamter
-16	DUP	duplicate	-42	MACHINE	
-17	SWA	swap	-43		
-18	ROT	rotate	-44		
-19			-45		
-20			-46		
-21	DIA		-47		
-22	LENGTH		-48		
-23	TLP	positive tolerance	-49		
-24	TLN	negative tolerance	-50		
-25	FLATNESS				
-26	TRU	true position			

the variable stack when the atom is a system variable. If it is an operator, the appropiate operation is applied.

For example in Figure 6.12 a conditional expression (DIA ! 0.0625 >=) implies "diameter greater than or equal to 0.0625". The normal mathematical term is written: DIA >= 0.0625; however, in post-fix notation, the operator is put at the end of an expression. When the above expression is evaluated, the address of DIA (position in which that value is stored) is pushed onto the variable stack. The symbol "!" means pop out the first item from the variable stack and use it as the address to move the data value from the variable array to the arithematic stack. The value in location 1 is moved to the arithmatic stack (in this case, a 2 is moved). The next atom, 0.0625, is a numeral; therefore, it is pushed onto the arithemetic stack directly. Finally, the >= symbol pops out two values from the arithmetic stack and compares them. If a TRUE condition results from the comparison, the next expression will be evaluated. Conditional expressions are ANDed together. And if all conditional expressions are true, the action expressions will be evaluated.

A few more operators are worth discussing here. All the mathematical operators are standard. However, stack operators have their own special meaning in PKI. The operator

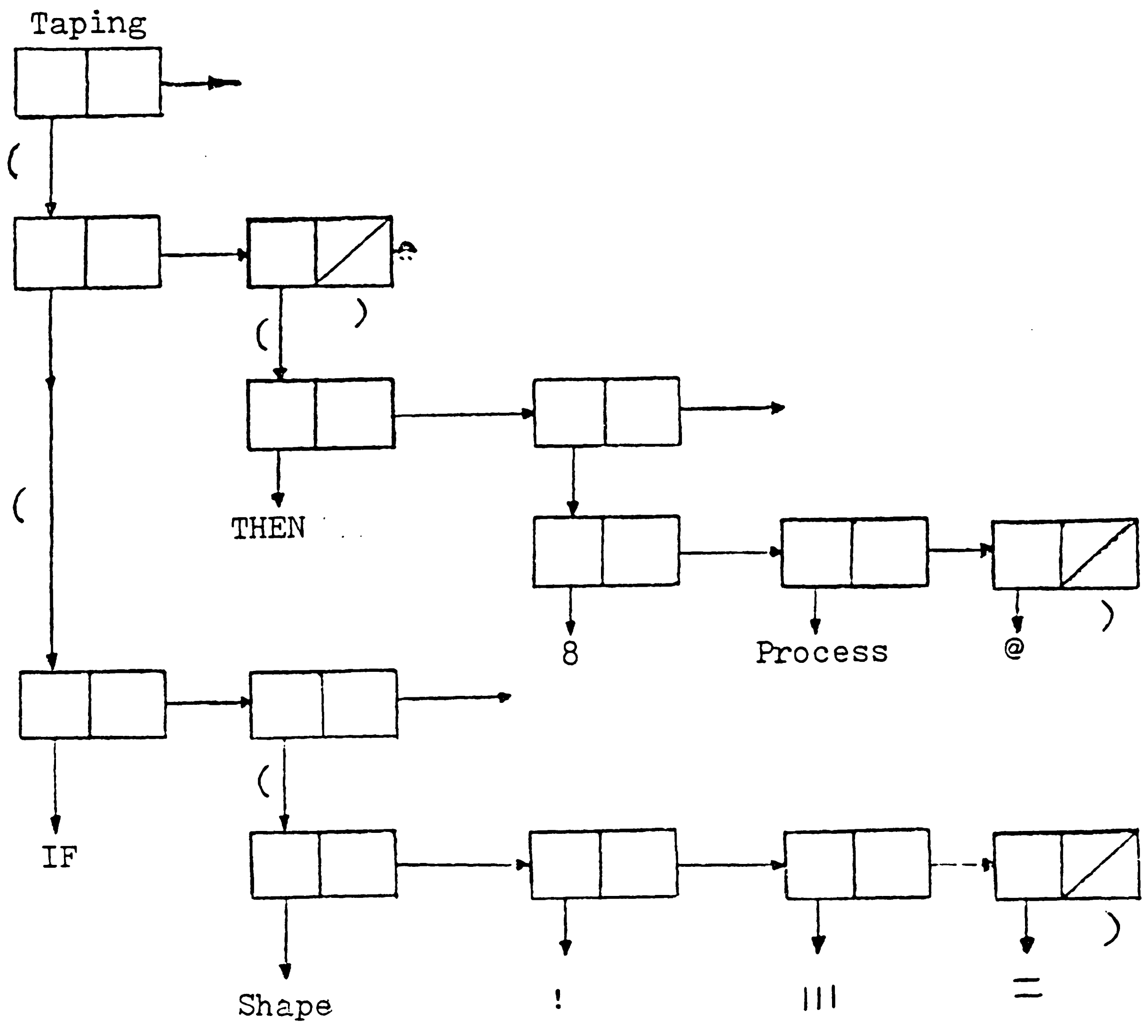


Figure 6.11 Data Structure

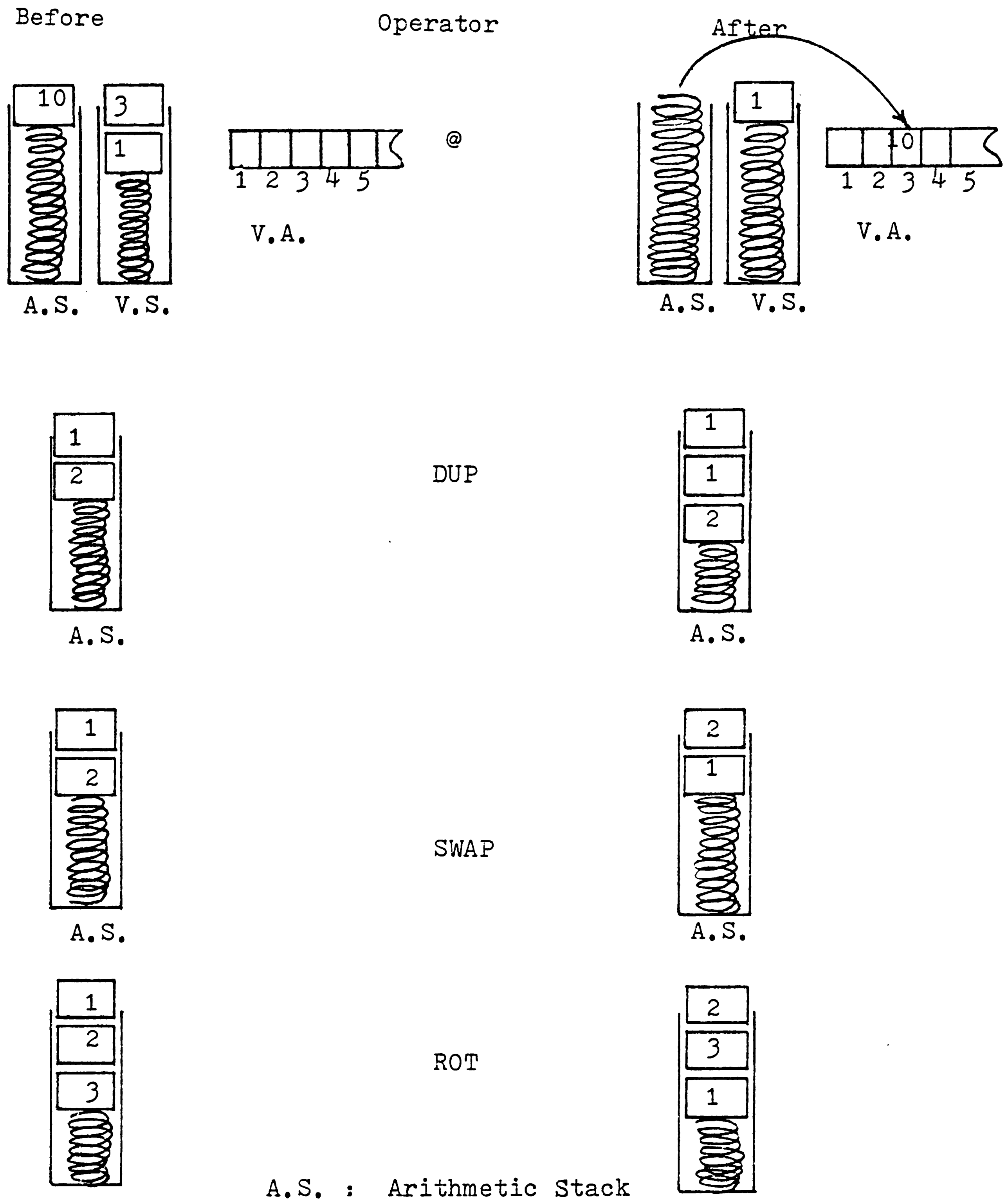
" @ " is very different then "!". " @ " removes the first item in the arithmetic stack and then changes the part condition by placing the value in the variable stack (Figure 6.13). This operation is useful for changing pre-conditions of a surface. DNP can duplicate the top item on the arithmetic stack. It is a handy feature when a variable is used several times in an expression. "SWAP" <u>swaps</u> the top two items on the arithmetic stack. "ROT" <u>rotates</u> three items on the arithmetic stack. By combining these operators with mathematical operators, statements describing process cabilities can be easily written. Examples using PKI will be illustrated in the next section.

6.4.2 <u>Process Description with PKI</u>

Process capabilities can be represented by their boundaries. For example, there are maximum and minimum hole diameters which can be drilled by twist drilling (due to tool availability). There are also boundaries on length, tolerances and surface finish. A typical process boundary for twist drilling can be written as:

Length < 12.0 d (length of the hole must be less than 12 times the diameter)

0.0625 < d < 2.000 (tool size is restricted between



A.S. : Arithmetic Stack

V.S. : Variable Stack

V.A. : Variable Array

Figure 6.12 Stack Operators

1/64" and 2.0" diameter)

$t_p \geqslant 0.007\ \sqrt{d}$ (tolerance (plus) as a function of drill size)

$t_n \geqslant 0.007\ \sqrt{d} + 0.003$ (tolerance minus)

straightness $\geqslant 0.0005\ (\frac{\ell}{d})^3 + 0.002$

roundness $\geqslant 0.004$

parallelism $\geqslant 0.001\ (\frac{\ell}{d})^3 + 0.003$

true position $\geqslant 0.008$

SF $\geqslant 100$

Using backward planning, the above process can "change a hole to a flat surface". A machined surface is finished after a process, if its condition is within the original boundaries. In order to translate these boundaries into PKI expressions, they can be written:

```
(LEN ! 12.0 DIA!* <=)
(DIA ! 0.0625 >=)
(DIA ! 2.000 <=)
(TLP ! DIA ! 0.5 ** 0.007 * >=)
(TLN ! DIA ! 0.5 ** 0.007 * 0.003 + >=)
(STRAIGHTNESS ! LEN ! DIA !/3 ** 0.005 * 0.002 + >=)
(ROUNDNESS ! 0.004 >=)
(PARALLELISM ! LEN ! DIA ! / 3 ** 0.001 * 0.003 + >=)
(TRUE ! 0.008 >=)
(SF ! 100 >=)
```

Shape also must be included, since drilling only creates round holes (Refer to Table 6.2 for shape code)

```
          (SHAPE ! 111 =)
```

The above expressions constitute the conditional portion of a statement. Actions are then specified to include the process identification, tool diameter and also mark the end of the search for each process.

```
          (1 PROCESS      @        )
          (DIA ! DTL      @        )
          (FREE)
```

The first expression puts process identification (1) into a process data file. DIA ! DTL @ retrieves the current hole diameter and saves it on the process data file as the desired tool diamter. FREE terminates the search.

Because the evaluation of expressions is sequential, conditional expressions should be arranged so that the more important an expression is, the earlier it is stated. Although there is no difference in the search results, the efficiency of search is affected. Figure 6.14 shows a complete set of statements for twist drilling input using PKI.

A finishing process normally does not start until a roughing process have been ompleted. In order to describe this, actions are required which include expressions to

```
;
;  TWIST DRILLING (CODE 1)
;
( ( IF
    (SHAPE ! 111 = )
    ( LEN ! 12.0 DIA ! * <= )
    ( DIA ! 0.0625  >= )
    ( DIA ! 2.000 <= )
    ( TLP ! DIA ! 0.5 ** 0.007 * >= )
    ( TLN ! DIA ! 0.5 ** 0.007 * 0.003 + >= )
    ( STRAIGHTNESS ! LEN ! DIA ! / 3. ** 0.0005 * 0.002 + >= )
    ( ROUNDNESS ! 0.004 >= )
    ( PARALLELISM ! LEN ! DIA ! / 3. ** 0.001 * 0.003 + >= )
    ( TRUE ! 0.008 >= )
    ( SF  ! 100 >= )
  )
  ( THEN
    ( 1 PROCESS @ )
    ( DIA ! DTL @ )
    ( FREE )
  )
)
```

Figure 6.13 Twist Drilling Input Using PKI

change the surface conditions between processes and while not signaling a FREE expression. For example, a finish boring process can be expressed as shown in Figure 6.15. (DIA ! .002 - DIA @) implies that finish boring can remove a 0.002 layer of material. Other expressions such as (0.002 TLP @), (1 TRUE @) etc., change the hole conditions so that a roughing process can be selected during the next iteration. More details can be found in The TIPPS Users Manual (Chang 1982).

6.5 TIPPS OPERATION

The TIPPS system and its structure has been detailed in the previous sections. The logical continuation is to demonstrate how TIPPS works. One can start using TIPPS after the process knowledge base has been prepared. The hardware needed for running the TIPPS system includes, 150k bytes of memory, a disk system and a graphics terminal with a crosshair cursor. The current version of TIPPS is divided into three parts: the TIPPS executive, PARSEL and REPORT. The TIPPS executive contains the graphics display, surface marking and process selection modules. All of the modules can be envoked by typing commands while in the executive mode. Both PARSEL and REPORT are independent of the executive and can be evoked under the operating system of the computer.

```
;
;
;  FINISH BORING ( CODE 4 )
;
( ( IF
    ( SHAPE ! 111 = )
    ( DIA ! .3750 >= )
    ( DIA ! 10 <= )
    ( TLP ! .0003 >= )
    ( TLN ! .0003 >= )
    ( STRAIGHTNESS ! .0002 >= )
    ( ROUNDNESS ! .0003 >= )
    ( PARALLELISM ! .0005 >= )
    ( LENGTH ! 10 DIA ! * <= )
    ( TRUE ! .0001 >= )
    ( SF ! 8 >= )
  )
  ( THEN
    ( 4 PROCESS @ )
    ( DIA ! DTL @ )
    ( DIA ! .002 - DIA @ )
    ( .002 TLP @ )
    ( .002 TLN @ )
    ( 1 STRAIGHTNESS @ )
    ( 1 ROUNDNESS @ )
    ( 1 PARALLELISM @ )
    ( 1 TRUE @ )
    ( 50 SF @ )
  )
)
```

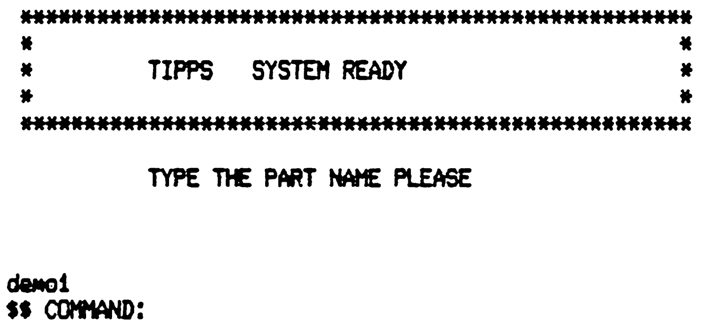
Figure 6.14 <u>Finish Boring Knowledge</u>

In order to generate a process plan, the following logical steps are required:

1. Input and display the CAD model,
2. Identify machined surfaces,
3. Select processes,
4. Select process parameters, and
5. Generate the report.

6.5.1 Input and Display of a CAD Model

The input for the CAD model is created by typing the file name in which the model is stored when running the TIPPS executive (See Figure 6.16). A typical CAD input model is shown in Figure 6.17. After a model has been input, it can be displayed in either 2-D or 3-D. A plot command puts the system in the graphics mode. A menu prompt from the user for the desired plot is then required. Hidden lines are not removed from either the 2-D or 3-D plots. The user can scale a 2-D plot to find coordinate information directly from the screen. In a 3-D environment, the user can view the object from any direction, but the display is always scaled to fit into the display window on the screen. Figure 6.18 shows two views of the same object. Different views help the user in the next task.

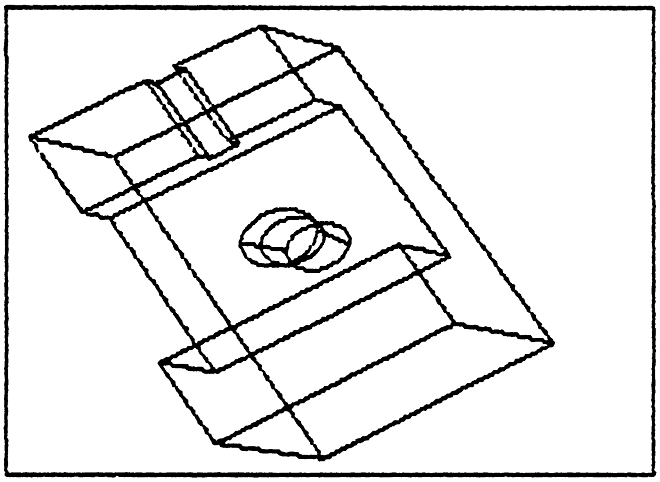


```
*****************************************************
*                                                   *
*        TIPPS   SYSTEM READY                       *
*                                                   *
*****************************************************

         TYPE THE PART NAME PLEASE


demo1
$$ COMMAND:
```

Figure 6.15 Entering the File Name

```
TEST PART - BRACKET P-10001
 100   2   200
 110   1     2     1     2     3     4     5     5     7    -8
 110   2     4     9    10   -11    -6     1
 110   3     4    13   -12    -9    -5
 110   4     4   -14   -13    -4   -15     2
 110   5     8    21   -20   -19   -18   -17   -16   -15    -3
 110   6     4    -2    23   -22   -21     3
 110   7     4   -24    17   -26    25     3
 110   8     4    18    28    27   -26
 110   9     4    29   -23    19    30     4
 110  10     4   -20    30   -31   -22
 110  11     8    -1    33   -32   -25   -27   -29   -31   -23
 110  12     8   -10    12    14    16    24    32    36   -34
 110  13     4     7    35   -34   -11
 110  14     4    -8    33    36   -35
 210   1     1     2
 210   2     2     3
 210   3     3     4
 210   4     4     5
 210   5     5     6
 210   6     6     7
 210   7     7     8
 210   8     1     8
 210   9     6     9
 210  10     9    10
 210  11     7    10
 210  12     9    11
 210  13     5    11
 210  14    11    12
 210  15     4    12
 210  16    12    13
 210  17    13    14
 210  18    14    15
 210  19    15    16
 210  20    16    17
 210  21     3    17
 210  22    17    18
 210  23     2    18
 210  24    13    22
 210  25    21    22
 210  26    14    21
 210  27    20    21
 210  28    15    20
 210  29    19    20
 210  30    16    19
 210  31    18    19
 210  32    22    23
 210  33     1    23
 210  34    10    24
 210  35     8    24
 210  36    23    24
 310   1     0     0     0
 310   2     0     0     3
 310   3     2     0     3
 310   4     2     0     2
 310   5     6     0     2
 310   6     6     0     4
 310   7     8     0     4
 310   8     8     0     0
 310   9     6     4     4
 310  10     8     4     4
 310  11     6     4     2
 310  12     2     4     2
 310  13     2     4     3
 310  14     2    25     3
 310  15     2    25    26
 310  16     2     2    26
 310  17     2     2     3
 310  18     0     2     3
 310  19     0     2    26
 310  20     0    25    26
 310  21     0    25     3
 310  22     0     4     3
 310  23     0     4     0
 310  24     8     4     0
 310  25     4     2     2
 130   1     2    25    12     5    10    10     5
 130   0     1    25    10     1    10    10     6
 500   1     5     5    50     5     5   100     5
 500   2     5     5    50     5     5   100     5
 500   3     5     5    50     5     5   100     5
 500   4     5     5     5     5     5   100     5
 500   5     5     5   120     5     5     1     5
 500   6     5     5   120    10     1    10     5
 999
```

Figure 6.16 A CAD Input Model



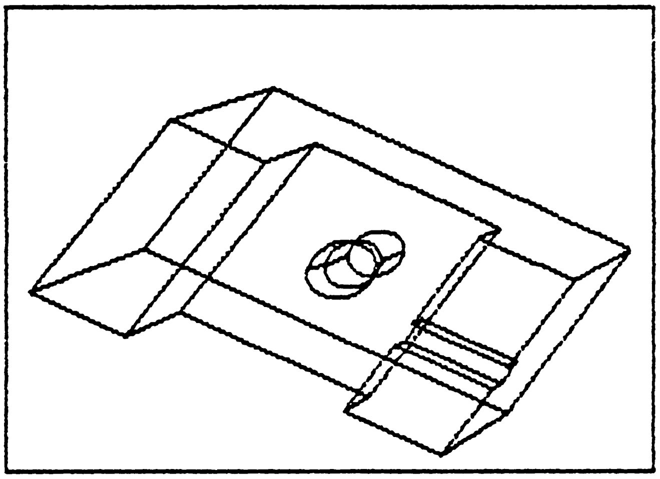


Figure 6.17 View an Object from Different Direction

6.5.2 Surface Identification

There are two steps in identifying a machined surface. One needs to tell the computer what the shape of a machined surface is and which faces are included in that surface. Both of these steps can be done interactively in TIPPS. The shape is identified by choosing items from a menu. The user need not know the code for different surfaces. The menu, in table format, is shown in Figure 6.19.

The linking of faces to form a machined surface is done interactively with a crosshair cursor. The cursor is controlled by either thumb wheels or joysticks-depending on the hardware. The reference point (the center of a face) is first plotted on the display. By locating the cursor on the reference point of a face and typing a 'Y', the face will be linked and saved in the surface file. A relationship matrix will be created to show the relationships of surfaces. A surface is said to cover another surface, if it needs to be machined before the other surface. The rule used to determine surface relationships for process sequencing is based on the bottom face surface height. Because this does not always provide the correct surface relationships, an interactive modification routine is available to change the surface identifications and relationships. A surface editor is also available to add and delete surfaces as well as faces

1st Digit		2nd Digit		3rd Digit	
1	Cylindrical	1	external	1	simple
				2	groove
				3	spline
		2	internal	4	keyway
				5	thread
				6	straight chamfer
				7	inner fillet
				8	outer fillet
2	Non-Cylindrical	0		1	flat
				2	step
				3	straight slot
				4	dovetail
				5	V bottom
				6	T or Y slot
				7	pocket
				8	square hole
				9	curved surface

Figure 6.18 Surface Type Code Table

of a surface. The editing is also performed direction on a CRT display.

6.5.3 Select Processes

Process selection in TIPPS requires only a single command. The command 'PROSS' will initiate the process selection module. The selected processes will be stored in the process file and later displayed on the CRT screen in sequence. Figure 6.4 shows an example of this display.

6.5.4 Select Process Parameter

The process parameter selection module is a separate program. Appropriate parameters are selected from built-in tables. The parameter selection module is evoked by typing the module name 'PARSEL' and then the part name. Process parameters as well as machining time are saved in the parameter file and displayed on the CRT screen.

6.5.5 Report Generation

The function of a report generator is to print a human readable report of the results. The report includes part name, part number, material code, planner, planning data, etc. The contents include surface identification, dimensions, processes, tool diameter, feed, speed and machining time. Figure 6.21 shows the report generation procedure (lower case words are typed by the user and uppercase words by the system). A process plan for the part in Figure 6.17 is shown in Figure 6.22 (see Figure 6.20 for surface identification information).

There are several aspects of the process plan shown in Figure 6.22 that are worth mentioning. First of all, the plan, as shown, was generated automatically. Secondly, the plan can be improved upon, significantly, with a bit of "massaging". For instance, because of the way TIPPS generates the process sequence, the sequencing of operations does not take tool changes or fixturing changes into account. By simply inspecting the process plan, one can see that if operations 9 and 10 are interchanged; a tool change (set up) can be eliminated. The processing time reflected in the plan does not include set-up or tool change time. These times (tool change and set-up), however, are system dependent. If the part were being machined on a CNC Machining center, only one set-up would be required; and depending on the machine, the tool change time might be negligable. On the other hand, in a manual system, set up and tool change may contribute the largest portion of the processing time.

```
RELATION MATRIX:
 0 0 0 0 0
 1 0 1 1 1
 1 0 0 1 1
 0 0 0 0 1
 1 0 0 0 0
CHANGE ? (Y/N),
n
```

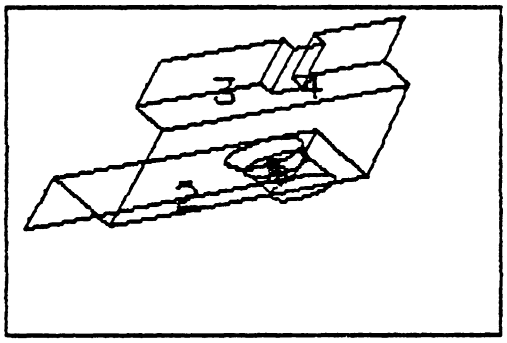


Figure 6.19 Marked Surfaces

```
CMS

report

 ***********************************************
 *                                             *
 *        PROCESS PLAN REPORT GENERATOR        *
 *                                             *
 ***********************************************

          TYPE THE PART NAME PLEASE


t3d
YOUR NAME PLEASE :
t. c. chang
PLANNING DATE :
august 13, 1982
R;
```

Figure 6.20 Report Generation

```
+------------------------------------------------------------------------------------------------------------------+
|                                                                                                                  |
|                                          P R O C E S S   P L A N                                                |
|                                                                                                                  |
| PART NAME: TEST PART - BRACKET                                    PART NUMBER: P-10001                           |
| MATERIAL:              2                                          PLANNER:     T. C. CHANG                       |
|                                          DATE :AUGUST 13, 1982                                                   |
|                                                                                                                  |
+------------------------------------------------------------------------------------------------------------------+
| OP |   SURFACE     |    L    |    D    |         | PROCESS  | TOOL   |  FEED   | SPEED  |  TIME  | REMARKS|
| NO |  ID  | TYPE   |    L    |    W    |    H    |          | DIA    | (IPR)   | (RPM)  | (MIN)  |        |
+------------------------------------------------------------------------------------------------------------------+
|    |      |        |         |         |         |          |        |         |        |        |        |
|  1 |  2   | FLAT   |  4.000  |  2.000  |  0.0    |R F MILL  | 2.040  | 0.02091 | 117.87 |  1.62  |        |
|  2 |      |        |         |         |         |F F MILL  | 2.040  | 0.02362 | 254.42 |  0.67  |        |
|  3 |  3   | FLAT   |  4.000  |  2.000  |  0.0    |R F MILL  | 2.040  | 0.02091 | 117.87 |  1.62  |        |
|  4 |      |        |         |         |         |F F MILL  | 2.040  | 0.02362 | 254.42 |  0.67  |        |
|  5 |  4   | S SLO  |  2.000  |  0.500  |  0.400  |R F MILL  | 0.500  | 0.00886 | 375.04 |  0.60  |        |
|  6 |      |        |         |         |         |F F MILL  | 0.500  | 0.01181 | 840.65 |  0.20  |        |
|  7 |  5   | STEP   |  4.000  |  4.000  |  2.000  |R P MILL  | 4.000  | 0.00472 | 168.54 | 25.12  |        |
|  8 |  1   | HOLE   |  1.200  |  0.840  |  0.0    |TWST DRL  | 0.840  | 0.01931 | 323.00 |  0.08  |        |
|  9 |      |        |         |         |         |CHAMF B   | 1.200  | 0.00546 | 161.68 |  0.57  |        |
| 10 |  1   | HOLE   |  1.200  |  0.993  |  0.0    |TWST DRL  | 0.993  | 0.02139 | 272.99 |  0.17  |        |
| 11 |      |        |         |         |         |RGH BORE  | 0.998  | 0.00506 | 164.79 |  1.20  |        |
| 12 |      |        |         |         |         |F. BORE   | 1.000  | 0.00506 | 227.36 |  0.87  |        |
|    |      |        |         |         |         |          |        |         |        |        |        |
+------------------------------------------------------------------------------------------------------------------+
|                                                      TOTAL PROCESSING TIME :    33.38 MINUTES                    |
|                                                                                                                  |
+------------------------------------------------------------------------------------------------------------------+
```

Figure 6.21 A Process Plan

6.5.6 Testing Results

For any machined surface, the processes required depends on the workpiece shape and surface conditions. Using TIPPS, immediate feedback can be obtained when surface condition specification changes. When used in conjunction with a CAD system, the design specification for a surface condition can be checked before issueing the design to production. A designer can avoid specifying unnecessarily tight tolerances and surface finishes.

Table 6.4 shows the effects of surface finish to process selection and machining time. The example part (Figure 6.18) was used for illustrative purposes. Five test runs were performed with different surface finish specifications being given to the various surfaces. The processes required for each surface are listed in the table. Since surface 2 and 5 require the same processes, only one set of processes is included in the Table. From the Table, a total of 29.87 minutes of machining time is required when all five surfaces have a 125 micro-inch surface finish. The total time increased to 48.02 minutes when the surface finish became 20 micro-inches. This represents a 60% increase in processing time. Table 6.5 shows how a change in tolerance effects a process plan. Such information can be provided to the designer for possible modification of the design. After the modification, a process plan can be generated with no addition effort.

Table 6.4 Processes and Total Machining Time under Different Surface Finish

	Surface		Tests				
	No	type					
Surface Finish	1	hole	125	60	30	30	20
	2	flat	125	125	30	20	20
	3	flat	125	125	30	20	20
	4	S. slot	125	125	30	20	20
	5	step	125	125	30	20	20
Processes	1	Hole chamfer	T.D. C.B.	T.D. C.B.	T.D. C.B.	T.D. C.B.	T.D. C.B.
		body	T.D.	T.D. R.B.	T.D. R.B. F.B.	T.D. R.B. F.B.	T.D. R.B. F.B. G.
	2 3 4 5	flat flat S. slot step	R.F.M.	R.F.M.	R.F.M. F.F.M.	R.F.M. F.F.M. G.	R.F.M. F.F.M. G.
Total time(min)			29.87	30.98	34.85	46.72	48.02

T.D.:	twist drill	R.F.M.:	rough face mill
C.B.:	chamfer bore	F.F.M.:	finish face mill
R.B.:	rough bore	G. :	grind
F.B.:	finish bore		

TABLE 6.5

<u>Processes Under Different Tolerances</u>

Hole Body

Tolerance $\pm$	0.01	0.005	0.0005
Processes	Twist Drill	Twist drill Rough bore	Twist Drill Rough bore Finish bore

Chapter VII

CONCLUSIONS

In this thesis, process planning system design guidelines as well as background information for process planning are provided. A process planning system is the interface between design and manufacturing. In the past two decades, much attention has been given to both design and manufacturing automation, individually. Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) systems have evolved separately. Our current inability to communicate between CAD and CAM was realized, and CAD and CAM integration work has started. So called "integrated CAD/CAM systems" became available on the commericial market in the past decade. However, these systems are nothing but CAD/APT systems, which save the geometry definition during part programming. This is simply a lower level integration. A higher level integration, CAD/process planning, is still not available.

Automated process planning is the interface between CAD and CAM. CAD is the upstream portion in the design/production information flow process, and CAM is the downstream portion. Process planning determines how to execute the operations necessary to fabricate a design in a manufacturing system. Part programming can be started only after process planning has been completed. Problems and potential solutions associated with CAD/process planning integratin issues are also addressed in this thesis. After reading this dissertation, the reader should have a complete picture of process planning systems approaches and structures. The author would like to conclude this dissertation with some discussions concerning the past and the future of process planning systems.

7.1 THE PAST

When the concept of computer-aided process planning was first conceived in the 1960's, group technology was spanned as process planning's first off-spring. Because of the complexity required to develop automatic planning systems, researchers were unable to develop intelligent computer software to generate process plans automatically. Assisting a process planner and saving planning time and effort was the main focus. Retrieving process plans for a similar component, then manually making appropriate modifications was the approach used. The term variant planning describes such an approach. In the first part of Chapter 4, the detailed steps in developing a variant process planning system was discussed. In Chapter 5, a review of several commerically available variant process planning systems were provided.

Although the variant approach cannot generate process plans automatically, the ease of developing and using the systems makes it attractive. Most process planning systems in the field, at the present time are still variant in nature.

Based on the experience of practitioners, a variant process planning system can save up to 90% of the process planning time. The plans generated are always consistent with those of similar components. However, many problems and deficiencies also exist. The most significant problem is the inability of these systems to provide plans for new components (where no similar component has been planned before). The process plan may not be consistent when compared with components in a different family. The success of the final process plan depends on the ability of the planner who modifies the plan which has been retrieved. An alternative approach which can avoid these problems is more desirable.

The generative process planning concept arose at the inception of variant process planning. However, difficulties in quantifying process capabilities are probably the major reason for the delay in generative planning development. Most generative process planning systems use "canned process plan" segments to construct process plans. Either a detailed GT code or a descriptive language is used to describe the component to be planned. A process plan is constructed using canned process plan segments which are recalled from the data base during the planning cycle. Although this type of process planning synthesis is not the most desirable, it is feasible and drastically reduces process planning decision making requirements.

Over the years some variant process planning systems have evolved into generative systems. Several commerical systems are in this category. They are by nature variant; however, some of the auxiliary functions of process planning such as parameter selection, tool selection, etc., can be generated automatically. Such systems are normally called (by their designers) generative process planning systems. Although not completely generative, they are one step closer to this goal.

Only in recent years, has some effort in integratingCAD with process planning systems been started. Without integration, a process planning system is part of the CAM function, and is independent of the CAD system. Process planning input is prepared by interpreting an engineering drawing or output from a CAD system. Several process planning systems use CAD for cutter path verification and tool fixture layout only. Most so called "CAD/process planning systems" use CAD as a display tool. None of the commercial systems use the data in a CAD databse to generate process plans. Manual interpretation of a drawing is still necessary. However, the TIPPS system, described in Chapter 6, is the first system which uses CAD data directly.

In the past, the main emphasis has been on developing a working process planning system. A temporary solution - variant process planning has proven successful. However, generic deficiencies associated with this approach make it less than ideal. Therefore, the generative approach is being explored. Although a true generative system is yet to come, a satisfactory system can be expected in the near future.

7.2 FUTURE DEVELOPMENT

As is always the dream, people prefer to build more automated systems thant exist today. In the past, we envisioned an unmanned factory working today. Currently several such subsystems exist and our goal for tomorrow is to integrate the entire industry (automated from design to final manufacturing). Process planning is a bridge between design and manufacturing. A process planning system which minimizes human decision making and data preparation is therefore desirable.

In Chapter 4, difficulties associated with the development of a generative process planning system were discussed.

The goal for the future is to overcome those difficulties and develop something better than what exists today. In order to achieve this goal, it is believed that the research direction in process planning will be toward the following:

1. Develop a unified data base for both CAD and process planning.

2. Study process, machine and tool behavior, building better mathematical (or descriptive) models which can be used in process planning systems.

3. Use artificial intelligence techniques in process planning.

4. Build a process capability knowledge base.

5. Develop techniques for marking machined surfaces automatically.

Ultimately, a generative process planning system can be a substitute for a process planner/manufacturing engineer in the design-manufacturing process. Knowledge concerning shop operations must be captured and saved in a unified design/manufacturing data base in order to be used for both design and planning. Such information cannot only assist in manufacturing planning, but also provide feedback to the designer.

The concept of design for manufacturing can be realized by using such feedback.

During the building of the data base, there are two important considerations that need to be included: process knowledge and facility information. Facility information simply describes what the system has, such as machine and tool inventory data. The process knowledge as discussed in Chapter 3, includes process capabilities. At this moment, satisfactory quantitative models and data for different processes are still not available. The most fundamental knowledge concerning a process is the tool life equation. Very few sound tool life equations have been identified. Other information concerning process capability is equally scarce. It is necessary to have complete process capability information before a true generative process planning system can be developed.

An efficient way to store and use process knowledge can determine the success of a system. The major advantage of Artificial Intelligence, versus the conventional approach, is the elegance of representing data and deducing conclusions from the data. It is most effective when qualitative decision making is involved. In process planning, a lot of information does not have a quantitative description, the AI approach can greaterly improve such this representation and decision making problem. A "self-learning" AI system, in the future, would be most interesting.

The AI approach discussed above is used in process, machine, etc. - selection decision making. Another important approach is in machined surface marking. This is the determination of which surfaces need to be machined in order to change a raw material to a designed shape. Only after such a marking process has been automated can a true automated process planning system be constructed.

7.3 CLOSING REMARK

In this chapter, a summary of the past and the future of the computer-aided process planning was presented. A lot more work is required before one can proudly say he has solved this difficult yet practical problem. It is the wish of the author that this thesis will enlighten intelligent people involved with this problem and finally complete this task.

REFERENCES

Allen, D. K., "Implications for Manufacturing Process Taxonomy in Process Selection," presented at the International CAM Congress, Hamilton, Ontario, Canada, McMaster University, May 14-16, 1974

Allen, D. K., "GENERATIVE PROCESS PLANNING using the DCLASS Information System," Monograph no. 4, Computer Aided Manufacturing Laboratory, Brigham Young University, Provo, Utah, 1979

Allen, D. K., "Computer Aided Process Planning OPCODES and Work Elements for Machined Parts," Final Report for CAM-I, Inc., Arlington, TX, Dec. 2, 1978.

Allen, D. K. and P. R. Smith, "Computer Aided Process Planning," Computer-Aided Manufacturing Laboratory, Brigham Young University. October 15,1980.

Armorego, E. J. and R. H. Brown, The Machining of Metals, Prentices Hall Inc., N.J., 1968.

Baer, A., C. Eastman and M. Henrion, "Geometric Modelling a Survey," CAD Vol. 11, No. 5, September 1979.

Barash, M. M., E. Bartlett, I. I. Finfter and W. C. Lewis, "Process Planning Automation- A Recursive Approach," The Optimal Planning of Computerized Manufacturing Systems, report No. 17 School of Industrial Engineering, Purdue University, West Lafayette, Indiana, 1980.

Barnes, R. D., "Group Technology Concepts Relative to the CAM-I Automated Process Planning (CAPP) System, presented to the Executive Seminar on Coding, Classifiction and Group Technology for Automated Planning, St. Louis, MO, p. 13

Belyakov, I. T., et al., "Identification Systems for Selecting an Optimum Technological Process," Machines and Tooling Vol. 43, No. 7, 1972.

Berra, P. B., "Investigation of Automated Planning and Optimization of Metal Working Process Processes," Ph.D. Thesis, Purdue University, Lafayette, Indiana, 1968.

Berra, P. B and M. M. Barash, "Investigation of Automated Planning and Optimization of Metal Working Processes," Report No. 14, Purdue Laboratory for Applied Industrial Control, July, 1968.

Bezier, "Numerical Control: Methematics and Applications," Translated by A. R. Forrest and A. F. Pankhnrst, 1972.

Boothroyd, G., Fundamentals of Metal Machining and Machine Tools, McGraw-Hill, 1975.

Bruyevich, N. G., B. Ye. Chelishchev, "Problems of Automation of Technological Planning - 1. Formal Description of Technological Planning," Engineering Cybern, V. 12 n. 5, pp. 154-162, Sep-Oct 1974.

Budde, W., "EXAPT in NC Operation Planning," Numerical Control Society Proc. 10 th Annual Tech. Meeting, 1973.

Burbidge, J. L., "Production Flow Analysis," The Production Engineering, April/May 1971.

Burbidge, J. L., The Introduction of Group Tecnology, Wiley, 1975.

CAM-I, "Functional Specifications for an Advanced Factory Management System," Computer Aided Manufacturing-International, Inc. Arlington, Texas, March 1979.

CAM-I "Functional Specification for an Experimental Planning System XPS-1" Computer Aided Manufacturing-International, Inc. Arlington, Texas, Oct 1980.

Carter, W. A. "Computer Aided Process Planning," Conf. on Compututer-Aided Manufacturing, National Engineering Laboratory (NEL), East Kilbride, Glasgow, Scotland, Jun 20-22, 1978.

Challa, K. and Berra, P., "Automated Planning and Optimization of Machining Procedures - A Systems Approach," Computers and Industrial Engineering, Vol. 1, pp. 35-46, 1976.

Chang, T. C., "Interfacing CAD and CAM - A Study of Hole Design," Unpublished Master's Thesis, Virginia Polytechic Institute & State University, Blacksburg, Virginia, 1980.

Chang, T. C. and R. A. Wysk, "Interfacing CAD/ Automated Process Planning," AIIE Transactions, September, 1981.

Chang, T. C., R. A. Wysk, R. P. Davis and B. Choi, "Milling Parameter Optimization Through a Discrete Variable Transformation," International Journal of Production Research, June, 1982.

Chisolm, A. W. J., "Design for Economic Manufacture," CIRP Annals, Vol. 22, 1973.

Claytor, R. N., "CAM-I's CAPP System," SME Tech. Paper, Series MS 1976.

Coelho, P. L. F., The Machining Economics Problem in a Probabilistic Manufacturing Environment, Unpublished M.S. Thesis, Virginia Polytechnic Institute & State University, Blacksburg, Virginia, 1980.

Collins, D., "Computer Aided Process Standardization," Paper published by TNO, Waltham, Massachusetts, 1977.

Cook, N. H. and K. L. Chanderamani, "Investigation on the Nature of Surface Finish and its Variation with Cutting Speed," Journal of Engineering for Industry, Vol. 86, No. 134, 1964.

Davis, R., B. Buchanan, and E. Shortliffe, "Production Rules as a Representation for a Knowledge-Based Consultation Program," Artificial Intelligence, Vol. 8, No. 1, 1977.

Descotte, Y. and J.-C. Latombe, "GARI: A problem Solver that Plans how to Machine Mechanical Parts," IJCAI 7, Vancouver, Canada, pp. 766-772, August 1981.

DeVor, R. E., W. J. Zdeblick, V. A. Tipnis and S. Buescher, "Development of Mathematical Models for Process Planning of Machining Operations NAMRC - VII, 6th North American Metalworking Research Conference Proceedings, SME, 1978.

Doran, J. M. and N. W. Sechrist, "Computer-Aided Process Planning and Work Measurement," Numerical Control Society Annual Meeting Tech Conf, 15th, Chicago, IL, Apr 9-12 1978.

Dunn, M. S. and W. S. Mann, "Computerized Production Process Planning," Proceedings 15th Numerical Control Society Technical Conference, Chicago, IL, April 1978.

Eary, D. F. and G. E. Johnson, Process Engineering for Manufacturing, Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

ElGomayel, J. and M. R. Abou-Zeid, "Piece Part Coding and the Optimization of Process Planning," Paper presented at the North American Metal Research Conference, May 1975.

Eversheim, W. and H. Fuchs, "Integrated Generation of Drawings, Process-Plans and NC-tapes," SME Tech Paper Series MS79-178, 1979. see also, Advanced Manufacturing Technology, P. Blake, ed. North-Holland, IFIP, 1980.

Eversheim, W., H. Fuchs and K. H. Zons, "Automatic Process Planning with Regard to Production by Application of the System AUTAP for Control Problems," Computer Graphics in Manufacturing Systems, 12th CIRP International Seminar on Manufacturing Systems, Beograd, 1980.

Eversheim, W., B. Holz and K. H. Zons, "Application of Automatic Process Planning and NC-Programming," Proceedings AUTOFACT WEST, Society of Manufacturing Engineers, Anaheim, CA, pp. 779-800, Nov 1980.

Fiacco, A. V. and G. P. McCormick, Nonlinear Programming, John Wiley & Sons, N.U., 1968.

GE, Operation Manual for the Carboloy Machinability Computer: Manual No. MC-101-B, Detroit, General Electric Company, Metallurgical Products Department, 1957.

Gilbert, W. W. and W. C. Truckenmiller, "Nomograph for Determining Tool Life and Power When Turning with Single-Point Tools," Mechanical Engineering, Vol. 65, pp. 893-98, 1943.

Grays, J. C., "Compound Data Structures for Computer Aided Design: A Survey," Proceedings ACM National Conference, Thompson Books, Washington, D.C. 1967.

Groover, M. P., "A Survey on the Machinability of Metals," SME Tech. Paper. MR76-269, SME, Dearborn, MI, 1976.

Groover, M. P., A. M. Gunda and R. J. Johnson, "Determination of Machining Conditions by a Self-Adaptive Procedure," Proceedings, 4th North American Metalworking Research Conference, 1976.

Haan, R. A., "Group Technology Coding and Classification Applied to NC Part Programming," Numerical Control Society Proc. 14th Ann. Tech. Meet., Pittsburgh, Pennsylvania March 1977.

Halevi, G., The Role of Computers in Manufacturing Processes, John Wiley & Sons, New York, 1980.

Halevi, G. and R. Weill, "Development of Flexible Optimum Process Planning Procedure," CIRP Annuals 1980, also, Manufacturing Technology, 29 Jan 1980, pp. 313-317.

Halevi, G., "Production Planning Module of All Embracing Technology," Proc. AUTOFACT West, Nov. 1980, Anaheim, CA., pp. 45-465, 1980.

Hasselt, R. V. and W. T. Oudolf, "Computer Aided Work Planning for Simple Sheet Components," CIRP Annals, Vol. 22, 1973.

Hati, S. and S,. Rao, "Determination of the Optimum Machining Conditions -- Deterministic and Probabilistic Approaches," Journal of Engineering for Industry, Vol. 98, Feb., 1976.

Hayes, G. M. Jr., R. P. Davis and R. A. Wysk, "A Dynamic Programming Approach to Machine Requirements Planning," AIIE Transactions, Vol. 13, No. 2, June 1981.

Hewgley, R.E. Jr., H.P. Prewett,Jr. "Computer Aided Process Planning at the Oak Ridge Y-12 Plant: a Pilot Project," Third Annual TNO User's Meeting, Arlington, TX, May 2-3, 1979.

Horvath, M. "Semi-Generative Process Planning for Part Manufacturing," SME Tech Paper Series MS 79-153, 1979.

Houtzeel, A., "Integrating CAD/CAM Through Group Technology" Numerical Control Society Proc Annu Meet Tech Conf 18th , Dallas, TX, May 1981, pp 430-444.

Houtzeel, A., "Computer Assisted Process Planning - A First Step Towards Integration," Proceedings AUTOFACT WEST, Society of Manufacturing Engineers, Anaheim, CA, Nov 1980, pp. 801-808.

ISO, Tool Life Testing with Single-Point Turning Tools, ISO, 5th Draft Proposal, ISP.TC 29/WG22 (Secretariat 37) 91, March 1972.

Iwata, K. et al., "Optimization of Cutting Conditions for Multi-pass Operations Considering Probabilistic Nature in Machining Processes," Journal of Enginee4ring for Industry, Vol. 99, Feb., 1977.

Iwata, K., Y. Kakino, F. Oba and N. Sugimura, "Development of Non-part Family Type Computer Aided Production Planning System CIMS/PRO," Advanced Manufacturing Technology, P. Blake, ed. North-Holland, IFIP, 1980.

Jackson, R. H. "Automated Process Planning - Key to Automating Manufacturing Support," Numerical Control Society proc. 16th Ann. Meet. Tech. Conf., Los Angeles, CA, Mar 25-28 1979

Japan, "Group Technology," Japan Society for the Promotion of Machine Industry, March 1980.

Kaczmarek, J., Principles of Machining by Cutting, Abrasion and Erosion, Voellnagel, A. and E. Lepa (trans.), Peter Peregrinus Ltd., Stevenage, England, 1976.

Kakino, Y. et al., "A new Method of Parts Description for Computer-aided Production Planning," Advances in Computer-Aided Manufacturing, North-Holland, 1977.

Kerry, T. M., "Integrating NC Programming with Machining and Group Technology," Numerical Control Society Proc Annu Meet Tech Conf 12th, 1975, pp.149-162.

Kikino, Y., F. Ohba, T. Meriwaki and K. Iwata, "A New Method of Parts Description for Computer-Aided Production Planning," Advances in Computer-Aidied Manufacturing, North-Holland, 1977.

King, J. R., 'Machine-Component Group Formation in Group Technology," 5th International Conference on Production Research, Amsterdam, The Netherlands, 12-16 August, 1979.

Kotler, R. A., "Computerized Process Planning - part 1" Army Man Tech Journal, Vol 4, No. 4. pp 28-36, 1980.

Kotler, R. A., "Computerized Process Planning - Part 2" Army Man Tech Journal, Vol 4, No. 5, pp. 20-29.

Krag, W. B., "Toward Generative Manufacturing Technology," Numerical Control Society Proc. 15th Ann. Meet. Tech. Conf., Chicago, IL, Apr 9-12 1978

Kyttner, R., N. Shtcheglov and A. Kimmel, "A Complex Computer-Aided Process Planning and Optimization for Machine Production," SME Tech paper series MS 79-158 1979 see also, Advanced Manufacturing Technology, P. Blake, ed. North-Holland, IFIP, 1980.

Lacoste, J. P. and R. Rothenberg, "Communication in Computer Aided Desgn and Computer Aided Manufacturing," in Computer Languages for Numerical Control, J. Havatny, Ed., (Proc. 2nd IFIP/IFAC Int'l PROLOMAT Conf., 1973), North-Holland, NY 1973, pp. 163-171.

Link, C. H., "CAPP - CAM-I Automated Process Planning System," Numerical Control Society Proc. 13th Ann. Tech. Meet., Cincinnati, OH, March 19 76.

Machover, C. and R. E. Blanth, eds, The CAD/CAM Handbook, computervision Corp., Bedford, MA, 1980.

Mann, W. S., M. S. Dunn, Jr. and S. J. Pflederer, Computerized Production Process Planning, United Technologies Research Corp. Rept #R77-942625-14, Nov. 1977.

McDaniel, H., Decision Table Software - A Handbook, Brandon/ Systems Press, Princeton, N.J., 1970.

Merrihew H. W., "Computerizing Process Planning for Machine Tools," Automation (Cleveland), V 17, n6 June 1970, pp. 56-59.

Metcut, Machining Data Handbook, Machinability Data Center, Cincinnati, 1980.

Metzner, J. R., B. H. Barnes, Decision Table Language and Systems, Academic Press, 1977.

Montalbano, M., Decision Tables, Science Research Associates, Chicago, 1974.

Myolopoulos, J., "An Overview of Knowledge Representation," Proc. Workshop on Data Abstraction, Databases, and Conceptual Modeling, June 1980.

Nagel, R. N., W. W. Braithwaite, and P. R. Kennicott, Initial Graphics Exchange Specification IGES Version 1.0, National Bureau of Standards, March 1980.

Nau, D. S., "Expert Computer Systems, and Their Applicability to Automated Manufacturing," Tech. Report., Industrial Systems Division, National Bureau of Standards, 1981.

Newman, W. M., R. F. Sproull, Principles of Interactive Computer Graphics, 2nd Edition, McGraw-Hill, 1979.

Niebel, B. W., "Mechanized Process Selection for Planning New Designs," ASTME paper No. 737, 1965.

Nilsson, N. J., Principles of Artificial Intelligence, Tioga Publishing Company, Palo Alto, California, 1980.

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

Parsons, N. R. (Ed.), N/C Machinability Data Systems, SME, Dearborn, MI., 1971.

Pedersen, J. T., B. Haugrud, and O. Bjorke, "General Approach to Technological Planning," Numerical Control Society Proc Ann. Meet. Tech. Conf., Chicago, IL, April 1972, pp 92-103.

Philipson, R. H. and A. Ravindron, "Application of Goal Programming to Machinability Data Optimization," Journal of Mechanical Design, Vol. 100, April, 1978.

Phillips, R. H., A Computerized Process Planning System Based on Component Classification and Coding, Ph.D. Thesis, Purdue University, Lafayette, Indiana, 1978

Pilederer, S. J. and W. S. Mann, "Principles of Computer Process Planning," SME prepr 760914 for meet. Nov 29 - Dec 2, 1976.

PII "PII Press release - 4-15-80," Planning Institute, Inc., Arlington, TX, 1980.

Planning Institute, Inc., "PI-CAPP" Publication of the Planning Institute, Inc., Arlington, TX, 1980

Planning Institute, Inc., "CAM-I's CAPP System: A General Interface for Interactive/Generative Process Planning," Publication of the Planning Institute, Inc., Arlington, TX, 1980.

Porazynski, R. J., "Using Group Technology Concepts in Manufacturing," AIIE System engineering Conference Proc. Kansas City, Mo. Nov 2-4 1977

Requicha, A. A. G., "Representations of Rigid Solid Objects," Computer Aided Design Modelling, Systems Engineering, CAD Systems, CREST Advanced Course, Darmstadt, Sep 1980, Ed. J. Encarnacao, Springer-Verlag, Berlin.

Requicha, A. A. G. and H. B. Voelcker, "Geometric Modelling of Mechanical Parts and Machining Processes," COMPCONTROL '79, Aopron, Hungary, November 1979.

Rolt, L. T. C., "Short History of Machine Tools," MIT Press, Cambridge, Mass., 1967.

Rose, D. W., Coding for Manufacturing COFORM, Unpublished M.S. Thesis, Purdue University, W. Lafayette, Indiana, 1977.

Scarr, A. J. T., Metrology and Precision Engineering, McGraw-Hill, London, 1967.

Scheck, D. E., "New Directions in Process Planning," SME Tech Paper MM75-908 for Meet. Ft. Lauderdale, FL., Feb 26-28, 1975.

Scheck, D. E., "Feasibility of Automated Process Planning," Ph.D. Thesis, Purdue University, 1966.

Schaffer, G., "GT via Automated Process Planning," American Machinist, May 1980, pp. 119-122.

Schilperoot, B. A., "Classification, Coding and Automated Process Planning," Published in CAM-I Proceedings #P-76-MM-01, Nov. 1975.

Scott, R. B., "Regenerative Shop Planning," American Machinist, Vol. 109, March 28, 1966.

Silberg, B., "DETAB/65 in Third-Generation COBOL," GIGPLAN Notices , 6, No. 8, September 1971.

Smart, H. G., "Group Technology and the Least Cost Method," Proceedings of AUTOFACT WEST, Society of Manufacturing Engineers, Anaheim, CA, November 1980, pp. 743-778.

Smith, B., J. Albus and A. Barbera, "Glossary of Terms for Robotics," Automation Technology Program, National Bureau of Standards, July 1981.

Spur, G., "Automation of Manufacturing Planning," Paper presented at the CIRP Conference, Chicago, IL, 1974

Stout, K. J. and G. Halevi, "A Computerized Planning Procedure for Machined Components," Production Engineer, Vol. 56, No. 4, April 1977, pp. 37-42.

Subbarao, P. and C. Jacobs, "Application of Nonlinear Goal Programming to Machine Variable Optimization," 6th NAMRC Proceedings, May, 1978.

Tempelhof, K. H., "A System of Computer-Aided Process Planning for Machine Parts," SME Tech Paper Series MS 79-154, 1979. Also in Advanced Manufacturing Technology, P. Blake, ed. North-Holland, IFIP, 1980.

Throop, J. W., "Computer Assisted Speed and Feed Selection for Automated Process Planning," CAM-I Standards Committee Report #R-76-SC-02.

Tipnis, V. A., S. A. Vogel and C. E. Lamb, "Computer-Aided Process Planning System for Aircraft Engine Rotating Parts," SME Tech Paper Series MS 79-155, 1979. Also in Advanced Manufacturing Technology, P. Blake, ed. North Holland, IFIP, 1980.

Tipnis, V. A., S. A. Vogel and H. L. Gegel, "Economic Model for Process Planning," NAMRC-IV, University of Florida, Gainesville, April 16-19, 1978. In Proceedings of the Sixth North American Metalworking Research Conference (Society of Manufacturing Engineers) 1978, pp. 379-387.

Tipnis, V. A., M. Field, and M. Y. Friedman, "Development and Use of Machinability Data for Process Planning Optimization," SME Tech Paper MD75-517, Chicago, IL, Feb 11-13, 1975.

TNO, "The Miturn Programming System for Lathes," Metaalinstituut TNO, December 1974, The Netherlands, Editor J. A. Jochems.

TNO, "Introduction to MIPLAN," Organization for Industrial Research, Inc., Waltham, MA, 1981.

Trucks, H. E., Designing for Economical Production, SME, 1974.

Tulkoff, J., "CAM-I Automated Process Planning (CAPP) System," Presented at the 15th Numerical Control Society Technical Conference, Chicago, IL, April 1978.

Tulkoff, J., "Lockheed's GENPLAN" Presented at the 18th Numerical Control Society Technical Conference, Dallas, TX, May 1981,pp 417-421.

Van Dyck, F., M. M. Tseng and O. D. Lascoe, "Study of Computerized Process Planning with Continuous and Discrete Data Base," AIIE Transactions, Vol. 8, No. 3, Sep 1976, pp. 320-327.

Voelcker, H. B. and A. A. G. Requicha, "Geometric Modeling of Mechanical Parts and Processes," Computer Vol. 10, No. 12, December 1977.

Vogel, S. A., "Integrated Process Planning at General Electric's Aircraft Engine Group," Proceedings of AUTOFACT WEST, Society of Manufacturing Engineers, Anaheim, CA, Nov 1980, pp. 729-742.

Vogel, S. A., "Metcut Machinability Process for NC," First Annual Conference on Computer Graphics in CAD/CAM Systems, Massachusetts Institute of Technology, April 9-11, 1979, pp. 5-15.

Vogel, S. A. and E. J. Adlard, "The AUTOPLAN process Planning System" Numerical Control Society Proc. 18th Ann. Meet. Tech. Conf., Dallas, TX, May 1981, pp. 422-429.

Vogel, S. A. and E. J. Adlard, "The AUTOPLAN Process Planning System," Metcut Research Associates, Inc., Ohio.

Vogel, S. and V. A. Tipnis, "The Concepts and Applications of the Metcut Machinability Processor for NC Machining," Numerical Control Society Proc. 14th Ann. Tech. Meet., March 1977, pp. 365-377.

Wandmacher, R. R., "Group Technology Concepts and Computer Aided Process Planning," SME Prepr, 750944, 1975.

Warner and Swasey, CUTS, A brochure on Computer Utilized Turning System, 1973.

Williams, R., "A Survey of Data Structures for Computer Graphics Systems," Computer Survey, Vol. 3, No. 1, March 1971.

Wysk, R. A. "An Automated Process Planning and Selection Program: APPAS," Ph.D. Thesis Purdue University, Lafayette, IN, 1977. ASME paper 78-WA Prod-13, 1978.

Wysk, R. A., "Process Planning Systems," MAPEC Module, Compiled by School of Industrial Engineering, Purdue University, 1979.

Wysk, R. A., T. C. Chang and R. P. Davis, "Analytical Techniques in Automated Process Planning," MAPEC module, Compiled by School of Industrial Engineering, Purdue University, 1980

Wysk, R. A., D. M. Miller and R. P. Davis, "The Integration of Process Selection and Machine Requirements Planning," Proceedings, AIIE 1977 Systems Engineering Conference, 1977.

Appendix A

TIPPS USER'S GUIDE

1. Introduction

TIPPS is an acronym for Totally Integrated Process Planning System. It is a generative process planning system with a CAD interface. TIPPS reads part description from a CAD database, then interacts with a process planner to mark surfaces which require machining. A process plan is generated automatically after machined surfaces have been marked.

A detailed discussion of the TIPPS system structure and design philosophy can be found in Chang and Wysk, TIPPS-Totally Integrated Process Planning System. In this manual we will present the following three major aspects of TIPPS.

1. How to use TIPPS to generate a process plan.

2. How to describe process capabilities in TIPPS.

3. TIPPS system flowchart and data format.

A complete source program listing is also attached. TIPPS is written in standard FORTRAN 66. The Tektronix PLOT-10 subroutine library is called in the system to generate graphic displays. In order to use the system, the minimum hardware requirements are:

1. 150k bytes of usable main memory,

2. disk system, and

3. a graphic terminal with crosshair cursor capability.

The system was originally run on a IBM 4341 computer under the VM/CMS operating system. The capability to access multiple sequential files is necessary. Files needed are listed in the appendix of this manual.

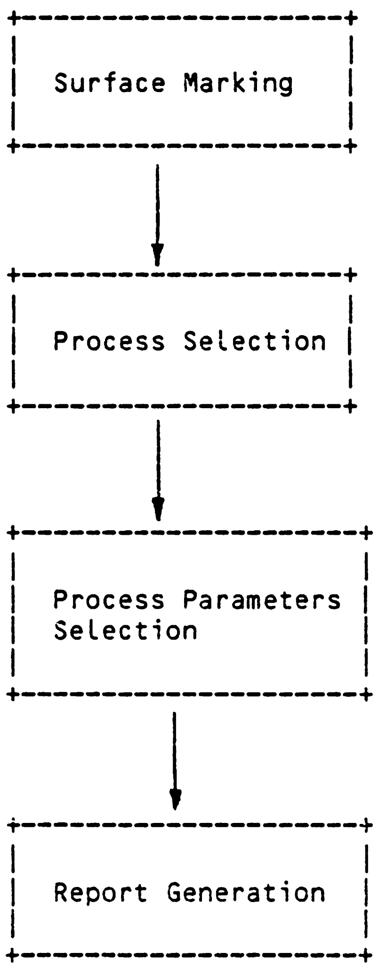
1.1 Format of the Manual

This user's manual is divided into three major parts. Part one is the TIPPS operating manual. For a user of the system, part one provides enough information to generate a process plan. Part two discusses the PKI - process knowledge description language. It is important only for the system maintenance person. During the installation of the TIPPS system, the process knowledge base should be updated to comply with the manufacturing system. PKI is the language needed for the updating. The third part of this manual includes the file structures and subroutine calling sequences of the system. Only when there are required modifying the TIPPS system is desired. Part three covers the most important information about the system itself.

2. TIPPS Operating Manual

The TIPPS system structure is shown in Figure 2.1. In order to generate a process plan, we need a part description which is in the TIPPS CAD data format. The data format is discussed in part three of this manual. Let's assume that we already have translated the CAD data of a part into the appropriate format, and the process knowledge data is also available. Now we can start the planning procedure.

The planning procedure can be divided into four stages. They are: 1. surface marking, 2. process selection, 3. process parameters selection, and 4. report generation.



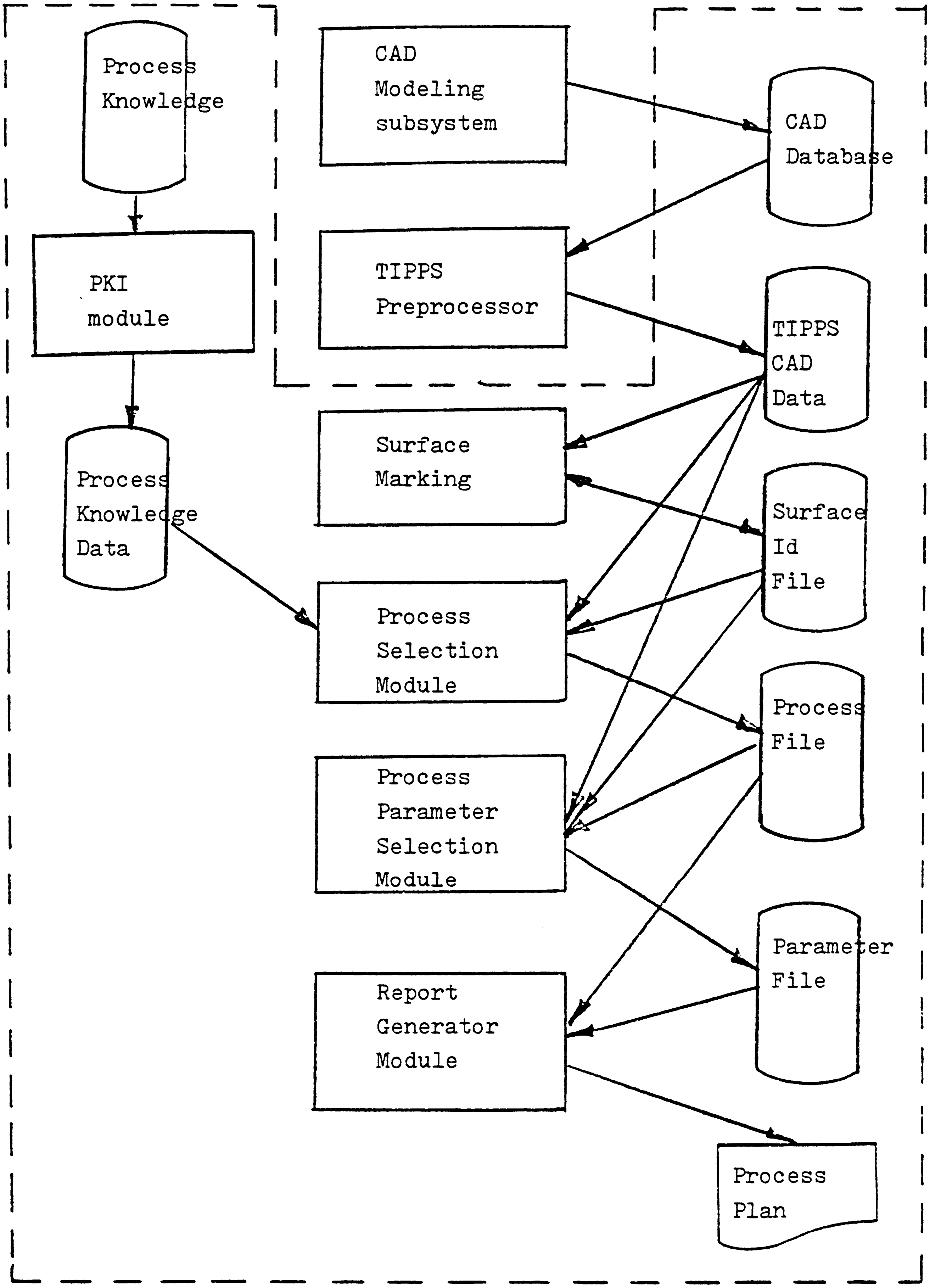


Figure 2.1 TIPPS structure

In the current TIPPS system, both the surface marking and process selection modules are included in one program called TIPPS. The process parameters selection module is called PARSEL, and the report generator is called REPORT. Each program can be invoked by typing the program name under CMS. In the following sections we will discuss how to use each of the programs.

2.1 Surface Marking and Process Selection

As mentioned, both the surface marking and process selection modules are included in one program - TIPPS. To invoke this program the syntax is: (commands are typed in lower case letters, system prompts are typed in upper case letters.)

```
tipps

*********************************************
*                                           *
*   TIPPS SYSTEM READY                      *
*                                           *
*********************************************


   TYPE THE PART NAME PLEASE
p-name

COMMAND:
```

where   p-name is the part id.

After TIPPS returns "command", different commands can be used to manipulate the display, marking surfaces and generate processes. TIPPS accepts commands in several levels. On the top level functional commands are accepted and interpreted. A functional command will change the system environment into a functional environment. Under such an environment, the system either prompts a user for the selection of actions or a waits a subcommand. The selection of command structure depends on the use. Basic functional commands include:

1. plot	--	display 2-D or 3-D drawing of the component.
2. scale	--	change the scale of a 2-D drawing.
3. sid	--	mark machined surfaces.
4. sed	--	edit marked surfaces.
5. process	--	process and cutter selection.
6. save	--	save surface marking or process selection data.
7. reload	--	reload surface marking or process selection data.
8. mode	--	select either graphics or non-graphics mode.
9. erase	--	erase the screen.
10. log	--	log out of the TIPPS system.
11. find	--	find the coordinate on a 2-D screen.
12. show	--	list marked surfaces, for non-graphics mode.

Only the first three letters in a command are significant. In the following sections, we will discuss each command separately.

2.1.1 Plot

The function of the plot command is to display either a 2-D or a 3-D drawing on a CRT screen. It is a tool to show the picture of a component being planned. After entering the plot command the system prompts:

```
PLOT TYPE?  2:2-D, 3:3-D
```

If a 2 is typed, a 2-D drawing will be displayed and the system returns to the command mode. If instead of 2, a 3 is typed, the system prompts:

SPECIFY VIEW POINT.

The user then specifies the view by typing the x,y,z location of the view point. It is assumed that the user is standing at the view point and looking toward the origin (0,0,0). In order to center the picture on the screen, the drawing is scaled and centered to fit into the display window. Therefore, the view point only specifies the view direction. The direction of the view is (-x,-y,-z) on a left hand Cartesian coordinate system (Figure 2.2). The real effect of the view point is to create a view direction passing the center of an object. The object is always scaled to the size of the display window.

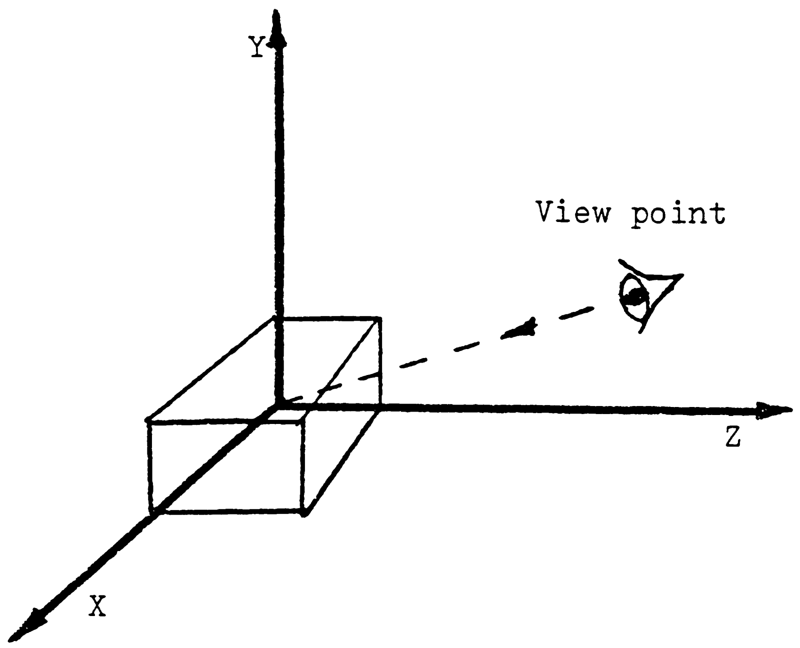


Figure 2.2 View point

2.1.2 Scale

The scale command allows a user to enlarge or reduce a portion of the drawing on a 2-D display (Figure 2.3). It is worth mentioning again that a 3-D drawing is always scaled to the display window size by the system. In 3-D, no local enlargement of a drawing is allowed. The scale command works only after a 2-D drawing has been displayed. The sequence of using this command is,

Step 1. type

```
scale,n
```

where n is the scaling factor.

A crosshair cursor appears on the screen after this command is entered.

Step 2. Move the cursor to the desired picture center on either the front view or the top view of the drawing.

Step 3. Type any key to inform the system of the location. At this time the system will beep once to acknowledge the receipt of the location data.

Step 4. Move the cursor to another view of the drawing, and repeat Step 3. If the user neglects to change the view, the system will prompt the user for the correct view. After both views have been entered, the picture will be redrawn.

Step 5. A "0" key typed during the crosshair cursor appearance will force the system out of the scale mode without changing anything.

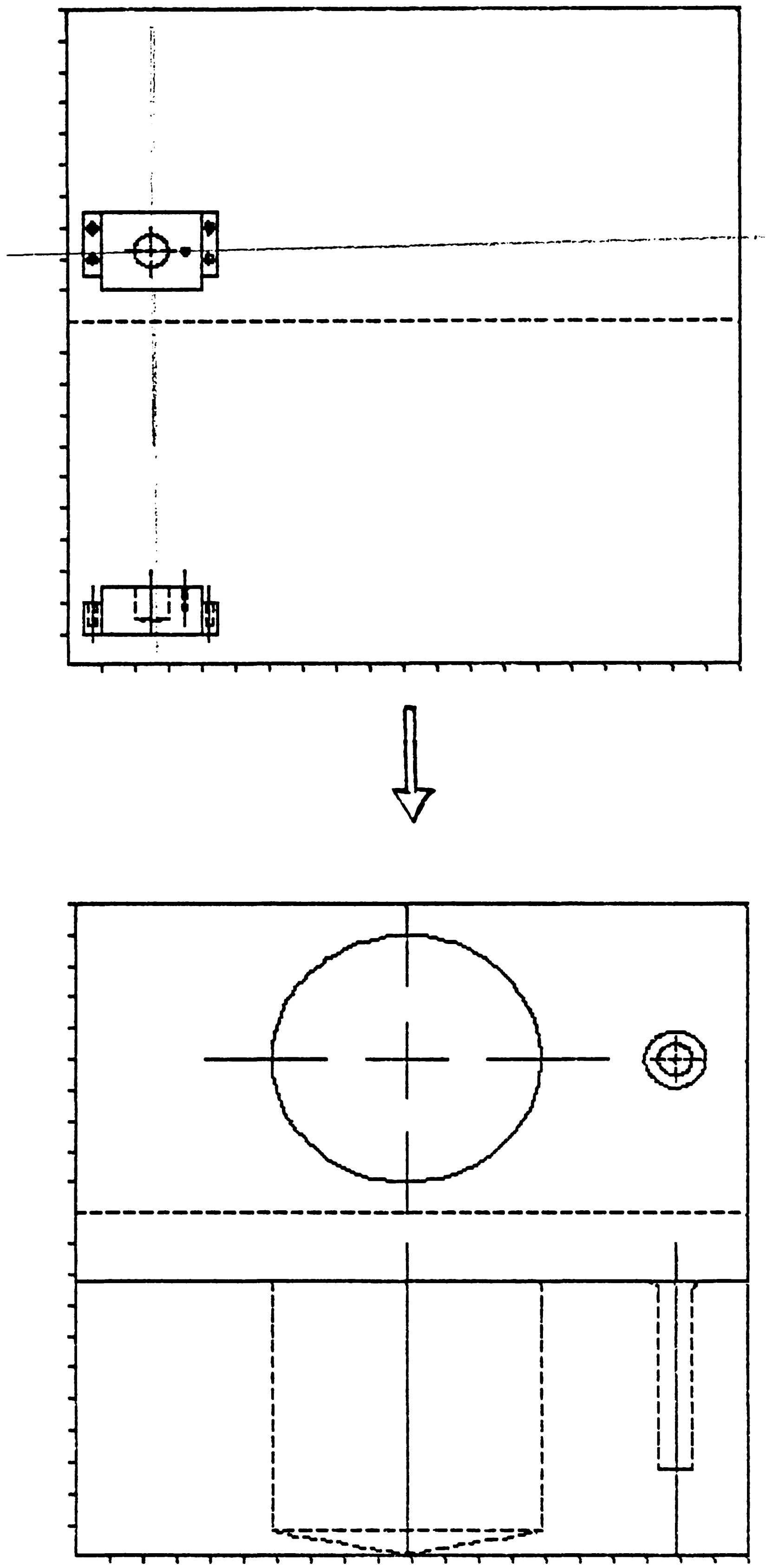


Figure 2.3 Scale a drawing

2.1.3 Sid

The sid command is used to mark machined surfaces. After "sid" is typed, the system goes into a surface marking mode. In order to mark a surface, first the surface type need to be identified. Then, all the associated faces need to be linked (Figure 2.4).

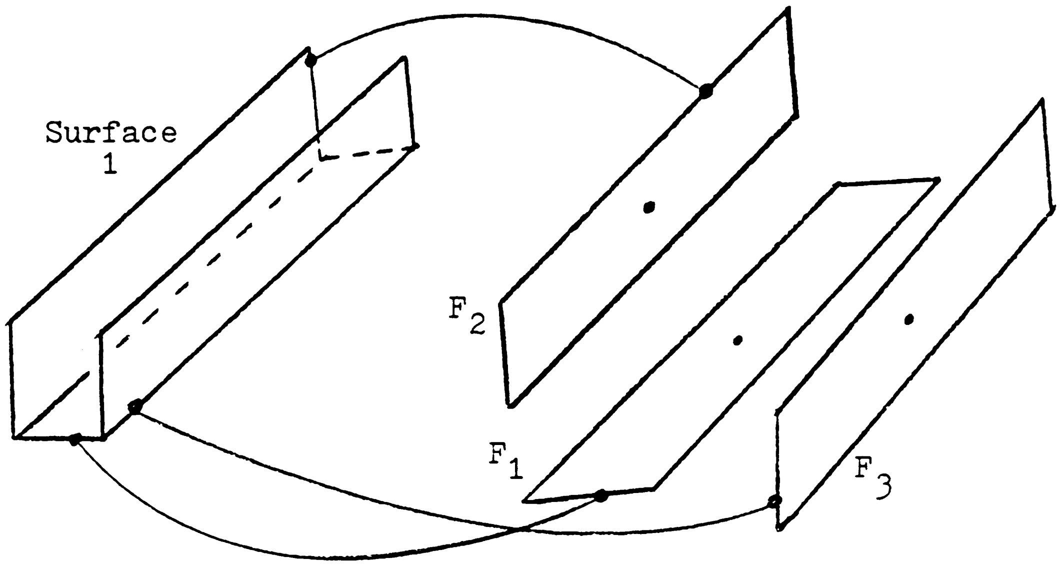


Figure 2.4 Surface linking

The following steps are required to mark a machined surface.

Step 1. Select surface type from the menu displayed by the system. After menu selection is completed, a graphics cursor will appear on the screen.

Step 2. Move the cursor to the center of the bottom face of the surface being selected. The center of a face is identified by a dot displayed on the drawing. Type any key to inform the system of the selection. If the system is able to find the face, it will redraw the face and return a beep to acknowledge the user mark.

Step 3. If the face found is the face desired, type a "Y" key. Otherwise type a "n" key. When the "Y" key is typed, the system will beep twice signaling that it is ready for the next face. Repeat step 2 until no other face is included in the surface. Otherwise go to step 5.

Step 4. In case a "n" key is typed, the system will keep searching for the face. At this time a user can move the cursor around to find the true center. Repeat step 2.

Step 5. type "O" key to terminate the face linking for a surface. During the appearance of a graphic cursor the "Q" key will exit the face linking without saving the current surface information.

Step 6. Upon terminating or exiting, the system will prompt the user for the next surface. If all the surfaces have been marked, enter a no and go to the next step. Otherwise, go back to step 1.

Step 7. The system will prompt for saving the marking information to a disk file. If a "yes" is typed, current marking will substitute the previous marking.

The surface relationship matrix is constructed automatically after surfaces have been marked.

2.1.4 Sed

Sed is a surface marking and relationship editor. Under this command previously marked surfaces and relationships can be edited. Some parts ares of sed are menu driven while other part command driven. The first thing sed does is displays a dotted 3-D drawing of the part. It then initiates the following procedures:

Step 1. If an individual surface need not be modified, answer the system prompt with a "no". The system will go to step 9. Otherwise next.

Step 2. One surface will be displayed on the screen with solid lines or curves. The system will prompt "change? y/n". If a "n" is typed, the system will repeat the next step until all surfaces have been displayed. If a "Q" is typed, the system will return to the TIPPS command mode. After all surfaces have been displayed, the system will go to step 8.

Step 3. The system is ready to accept a command. Commands available at this point include, remove, out, delete and add. We will discuss these commands in step 4 through 7.

Step 4. Remove implies to remove the current surface from the marked surface file. After this command the system will return to step 2.

Step 5. Out simply returns to step 2.

Step 6. The delete command deletes a face from the current surface. The add command is opposite to the delete command. The system handles both commands in the same manner as illustrated in the next step.

Step 7. First the system prompts for the surface type. if there is no change, type <return>. Otherwise enter the new surface type code (refer to the appendix).

At this moment the cursor will come up. Locate the cursor to the face being deleted or added and type any key. If no change is desired, type a "O" key. When the system locates the face from its data file, the system will return a beep to acknowledge this action to the user. The face found will be redrawn. If the surface found by the system is the correct surface, type a "Y" key, otherwise type any key other than "O" and "Y". Now the system will either attempt to find the correct surface or return to step 3.

Step 8. the system prompts "ADD NEW SURFACES? Y/N". If yes, the system will enter sid mode (see 1.1.3). If no, go to next step. If "QUIT", exit from the editor without change anything.

Step 9. the screen will be cleared, only the marked surfaces and surface numbers will be displayed. The relationship matrix will also be printed on the screen.

relationship matrix

```
0  1  0  1

0  0  0  0

1  0  0  0

0  0  0  0
```

s1 proceeds s2 and s4

s3 proceeds s1

The system prompts "CHANGE ? Y/N". If "N" is typed, go to next. Otherwise, the system will wait for the change. The change is represented by three integer numbers, s1,s2,r and separated by a comma. Where s1, s2 are surface numbers and r is the relationship. r=1, if s1 must be machined before s2. If s2 must be machined before s1 then r=-1. s1,s2,1 equivalent to s2,s1,-1. When it does not matter which surface needs to be machined first, r=0. s1 and s2 are also row number and column number respectively in the matrix. When the change is completed, type a null <return>. The system will go to the next step.

Step 10. the system prompts "SAVE?", a "Y" will save the change in disk file, otherwise only the information in the buffer is changed.

2.1.5 Process

The "process" command is used to generate processes for the component. Processes selected will be printed on the screen and also saved in a disk file for the parameter selection and final report generation. If the sequence of processes is depends on the surface relationship matrix, no additional subcommand is needed.

2.1.6 Save

The save command is used to save the current work in disk files to the system buffer (working area). It is useful when the previous marked surfaces have been temporarily modified and the testing shown positive. Three different types of information can be saved, they are surface marking data, CAD data and process data. The system will provide the menu.

2.1.7 Reload

The reload command is opposite to the save command. It reloads information from the disk files.

2.1.8 Mode

The mode command is provided in order to use the system on a non-graphics terminal. The syntax of this command is,

```
mode,n
```

where n is the mode code.

```
n=0 graphics mode
 =1 non-graphics mode
```

When the graphics mode is turned off, no picture will be drawn under the sid or sed commands. However these two commands still can be used. The plot command overrides the mode command, it always turns the graphics mode "on".

2.1.9 Erase

The erase command clears the CRT screen.

2.1.10 Log

The log command terminate the execution of the TIPPS system, and return the control to the CMS.

2.1.11 Find

The find command is used to find a coordinate on a 2-D drawing. It is not used very often, however, it provides a tool to find the location

6,2

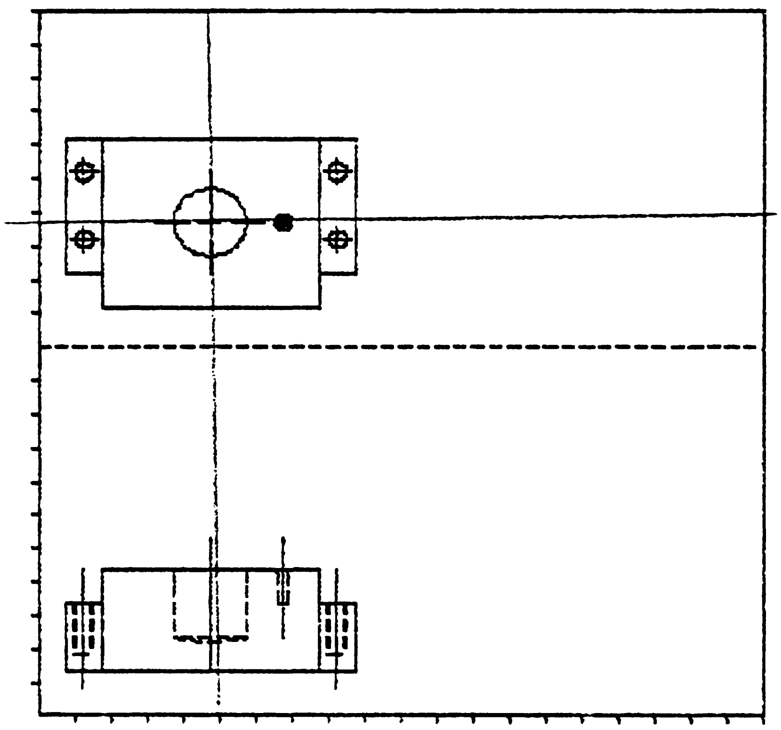


Figure 2.5 Finding a location coordinate

of a surface on the component (Figure 2.5).

The find command will generate a graphics cursor. Locate the cursor on the desired location and type any key other than "0" will print the coordinate on the screen. When "0" is typed the system returns to the command mode.

2.1.12 Show

The show command is used under non-graphics mode to list all the marked surfaces and surface numbers.

2.2 Parameters Selection

The parameter selection program is called PARSEL. It selects the most appropriate process parameters, and calculates machining time. There is no subcommand available in PARSEL. in order to run PARSEL, follow the following sequence.

```
parsel
*********************************************
*                                           *
*   PARAMETER SELECTION SYStEM READY        *
*                                           *
*********************************************

         TYPE THE PART NAME PLEASE
```

p-name After the part name is entered, the system will print out the selected process parameters and estimated machining time. All the information generated is saved in a file for report generation.

2.3 Report Generation

The report generation is accomplished by a report generator - REPORT. The only function of the report generator is to prepare a human readable process plan. In order to include the planning date and the planners name, the system will prompt the user on the date and name. However, REPORT can be modified to read date from the system directly. The syntax of calling the report generator is:

```
    report

       **************************************************
       *                                                *
       *       PROCESS PLAN REPORT GENERATOR            *
       *                                                *
       **************************************************

                TYPE THE PART NAME PLEASE

p-name
YOUR NAME PLEASE
your name
PLANNING DATE :
xxx xx, xxxx
```

A copy of the final process plan will be sent to the printer.

2.4 EXAMPLE

In this section an example will be shown. Through the step by step procedure an user should be able to learn how to use commands listed in the previous sections. The example component is shown in Figure 2.6. This component is made of low carbon steel. The design information is specified in the CAD data file. The file is called "demo".

2.4.1 Log In

First we need to log in the computer system, then enter the following command.

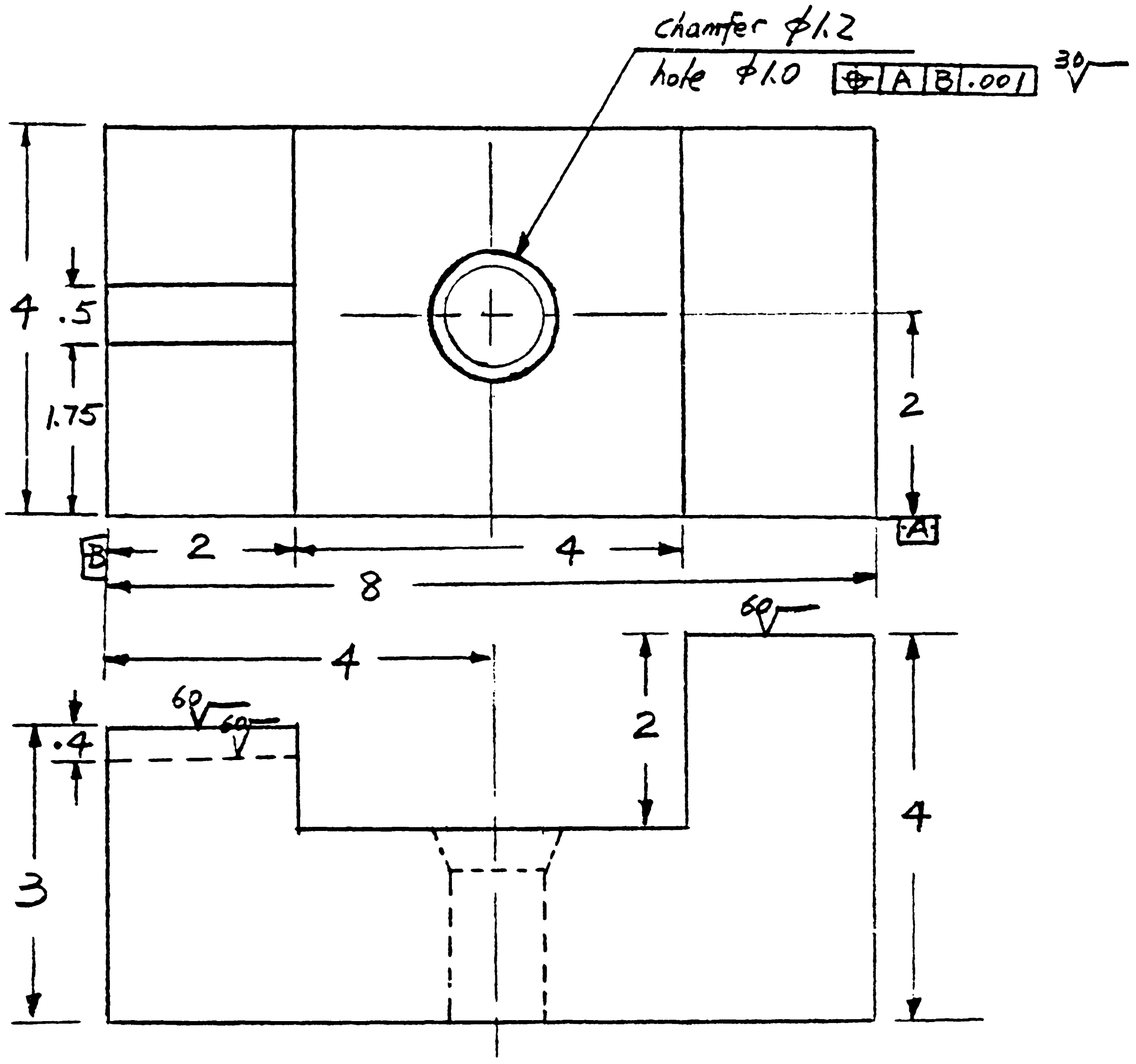
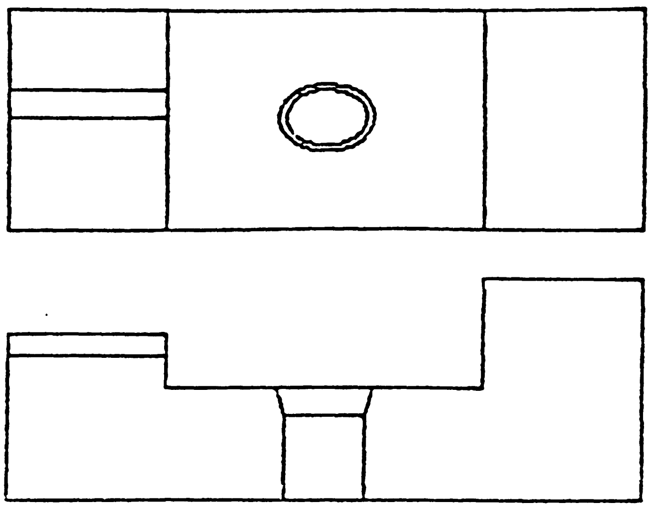


Figure 2.2 The component

```
     ***********************************************
     *                                             *
     *             TIPPS SYStEM READY              *
     *                                             *
     ***********************************************

                TYPE THE PART NAME PLEASE

 demo

 COMMAND:
```

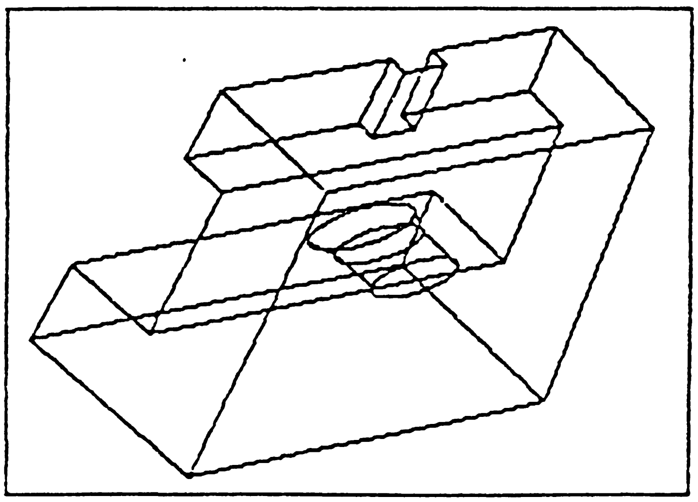
2.4.2 Plotting

```
     plot

     PLOT TYPE?   2:2-D, 3:3-D

     ?

     2
```



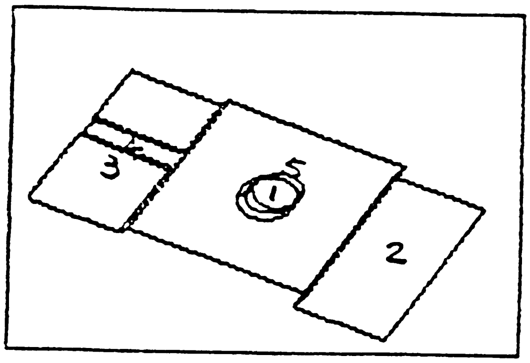
```
plot

PLOT TYPE?   2:2-D, 3:3-D

?

3

SPECIFY VIEW POINT.
```

```
?

80,50,60
```



The above session will plot the component - demo on the screen. Since the sid command is difficult to show, it will not be shown here. The surface modification part of the sed command also will not be shown for the same reason.

```
COMMAND:
sed
REVIEW INDIVIDUAL SURFACE? (Y/N)
n


RELATION MATRIX:
 0 0 0 0 0
 1 0 1 1 1
 1 0 0 1 1
 0 0 0 0 1
 1 0 0 0 0
CHANGE ? (Y/N)
n
```



```
SAVE THE WORK? (Y/N)
Y
COMMAND:
```

2.4.3 Process Selection

```
process

 ****  SURFACE :  2 ****

PROCESS:  R  F  MILL
PROCESS:  F  F  MILL

 ****  SURFACE :  3 ****

PROCESS:  R  F  MILL
PROCESS:  F  F  MILL

 ****  SURFACE :  4 ****

PROCESS:  R  F  MILL
PROCESS:  F  F  MILL

 ****  SURFACE :  5 ****

PROCESS:  R  P  MILL

 ****  SURFACE :  1 ****

PROCESS:  TWST DRL
PROCESS:  CHAMF  B

 **** NEXT HOLE SEGMENT ****
PROCESS:  TWST DRL
PROCESS:  RGH  BORE
PROCESS:  F.  BORE

 COMMAND:
log
TIPPS SYSTEM LOGOFF
WORK IS SAVED
R;  T=2.31/3.28  21:20:46N
```

2.4.4 Parameter Selection

The above example shows how to use TIPPS to generate process for a component. In the following section the parameter selection module will be demonstrated.

```
 parsel
 ************************************************
 *                                              *
 *        PARAMETER SELECTION SYSTEM READY      *
 *                                              *
 ************************************************

          TYPE THE PART NAME PLEASE

 demo

 ****  SURFACE :  2 ****

PROCESS:  R  F  MILL
  TOOL DIA: 2.04  FEED : 0.021  SPEED : 117.87
  MACHINING TIME : 1.623
PROCESS:  F  F  MILL
  TOOL DIA: 2.04  FEED : 0.024  SPEED : 254.42
  MACHINING TIME : 0.666

 ****  SURFACE :  3 ****

PROCESS:  R  F  MILL
  TOOL DIA: 2.04  FEED : 0.021  SPEED : 117.87
  MACHINING TIME : 1.623
PROCESS:  F  F  MILL
  TOOL DIA: 2.04  FEED : 0.024  SPEED : 254.42
  MACHINING TIME : 0.666

 ****  SURFACE :  4 ****

PROCESS:  R  F  MILL
  TOOL DIA: 0.50  FEED : 0.009  SPEED : 375.04
  MACHINING TIME : 0.602
PROCESS:  F  F  MILL
  TOOL DIA: 0.50  FEED : 0.012  SPEED : 840.65
  MACHINING TIME : 0.201

 ****  SURFACE :  5 ****

PROCESS:  R  P  MILL
  TOOL DIA: 4.00  FEED : 0.005  SPEED : 168.54
  MACHINING TIME : 25.117

 ****  SURFACE :  1 ****

PROCESS:  TWST DRL
  TOOL DIA: 0.84  FEED : 0.019  SPEED : 323.00
  MACHINING TIME : 0.080
PROCESS:  CHAMF  B
  TOOL DIA: 1.20  FEED : 0.005  SPEED : 161.68
  MACHINING TIME : 0.566

 **** NEXT HOLE SEGMENT ****
PROCESS:  TWST DRL
  TOOL DIA: 1.00  FEED : 0.021  SPEED : 272.99
  MACHINING TIME : 0.171
PROCESS:  RGH  BORE
  TOOL DIA: 1.00  FEED : 0.005  SPEED : 164.79
  MACHINING TIME : 1.199
PROCESS:  F.  BORE
  TOOL DIA: 1.00  FEED : 0.005  SPEED : 227.36
  MACHINING TIME : 0.869


***  TOTAL MACHINING TIME = 33.38 MIN.
R; T=0.54/0.86 21:21:00N
```

2.4.5 Report Generation

The following example shows how to generate a process plan.

```
report

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

*                                                *

*          PROCESS PLAN REPORT GENERATOR         *

*                                                *

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


          TYPE THE PART NAME PLEASE

demo

YOUR NAME PLEASE :

T. C. Chang

PLANNING DATE :

July 22, 1982
```

```
+-----------------------------------------------------------------------------------------------------------------------+
|                                                                                                                       |
|                                                                                                                       |
|                                               P R O C E S S   P L A N                                                 |
|                                                                                                                       |
|                                                                                                                       |
| PART NAME: TEST PART - BRACKET                                                    PART NUMBER: P-10001                |
| MATERIAL :           2                                                            PLANNER:     T. C. CHANG            |
|                                              DATE :AUGUST 13, 1982                                                    |
|                                                                                                                       |
+-----------------------------------------------------------------------------------------------------------------------+
| OP |   SURFACE    |    L    |    D    |          | PROCESS  |  TOOL  |  FEED   | SPEED   |  TIME   | REMARKS|
| NO |  ID  | TYPE  |    L    |    W    |    H     |          |  DIA   | (IPR)   | (RPM)   |  (MIN)  |        |
+-----------------------------------------------------------------------------------------------------------------------+
|    |      |       |         |         |          |          |        |         |         |         |        |
|  1 |  2   | FLAT  |  4.000  |  2.000  |   0.0    |R F MILL  | 2.040  | 0.02091 | 117.87  |   1.62  |        |
|  2 |      |       |         |         |          |F F MILL  | 2.040  | 0.02362 | 254.42  |   0.67  |        |
|  3 |  3   | FLAT  |  4.000  |  2.000  |   0.0    |R F MILL  | 2.040  | 0.02091 | 117.87  |   1.62  |        |
|  4 |      |       |         |         |          |F F MILL  | 2.040  | 0.02362 | 254.42  |   0.67  |        |
|  5 |  4   | S SLO |  2.000  |  0.500  |  0.400   |R F MILL  | 0.500  | 0.00886 | 375.04  |   0.60  |        |
|  6 |      |       |         |         |          |F F MILL  | 0.500  | 0.01181 | 840.65  |   0.20  |        |
|  7 |  5   | STEP  |  4.000  |  4.000  |  2.000   |R P MILL  | 4.000  | 0.00472 | 168.54  |  25.12  |        |
|  8 |  1   | HOLE  |  1.200  |  0.840  |   0.0    |TWST DRL  | 0.840  | 0.01931 | 323.00  |   0.08  |        |
|  9 |      |       |         |         |          |CHAMF B   | 1.200  | 0.00546 | 161.68  |   0.57  |        |
| 10 |  1   | HOLE  |  1.200  |  0.993  |   0.0    |TWST DRL  | 0.993  | 0.02139 | 272.99  |   0.17  |        |
| 11 |      |       |         |         |          |RGH BORE  | 0.998  | 0.00506 | 164.79  |   1.20  |        |
| 12 |      |       |         |         |          |F. BORE   | 1.000  | 0.00506 | 227.36  |   0.87  |        |
|    |      |       |         |         |          |          |        |         |         |         |        |
+-----------------------------------------------------------------------------------------------------------------------+
|                                                                   TOTAL PROCESSING TIME :     33.38 MINUTES    |
|                                                                                                                       |
+-----------------------------------------------------------------------------------------------------------------------+
```

3. PKI - The Process Knowledge Description Language

The process knowledge description language - PKI is designed to define the capability of a manufacturing system. It is not an appropriate way to design a single process knowledge database for all the manufacturing systems. Each installation of a process planning system should be tailored to fit into its specific manufacturing system. In order to be able to fit into different manufacturing systems, a process planning system must have the capability to describe the processes externally. The PKI language is designed to achieve this requirement.

The PKI language provides the user a tool to modify the process capability information easily without the full knowledge of the entire TIPPS program structure. The PKI is a process oriented language. It has mathematical and stack operators. Mathematical operators includes arithmetic and logic operators. Meaning: However in PKI all variables are predefined by the system, no user defined variable is allowed. Each process is defined by a statement. A statement is closed by a pair of paratheses. In a statement, there are two major portions, "IF" and "THEN". Both of them are closed by paratheses. Under each portion, there are expressions. Paratheses in a statement must match. A PKI statement has the format:

```
(
  ( IF
        ( expression 1 )
        ( expression 2 )
                  .
                  .
        ( expression n )
  )
  ( THEN
        ( expression n+1 )
```

```
              ( expression n+2 )
                      .
                      .
              ( expression n+m )
          )
        )
```

where:

```
<expression>      := <atom><atom>..<atom>

<atom>            := <literal atom>|<numeral>

<literal atom>    := <system variable>|<operator>

<numeral>         := integer or real number

<system variable>:= DIA, LENGTH, TLP, TLN, etc.

<operator>        := <math. operator>|<stack operator>

<math. operator> := <arith. oper>|<logic oper>

<stack operator> := @, !, DUP, SWAP, ROT

<arith. oper>     := +, -, *, /, **

<logic operator> := =, >, <, ><, >=, <=
```

Legal atoms can be found in Table 3.1.

Expressions 1 to n are condition expressions and n+1 to n+m are action expressions. A condition expression can either be true (T) or be false (F). Expressions 1 through n are "ANDed" together. Only when all of them are T, the statement is true, and actions are taken. Several logic operators can be put in a single condition expression, they are "ORed" together. Therefore complex decision can be made. A typical condition expression can be written as (DIA ! 2.00 >). The meaning of this expression is, retrieve the value of diameter and test whether it is greater than 2.000. The stack operator "!" implies retrieve the variable data value. Since all the operations are done on the stack, it is always necessary to retrieve a variable value and put on the stack before using. Expressions follow a post-fix convention. More detail will be discussed later.

Table 3.1
PKI operators and system variables

Code	Reserved Words	Meaning
-1	>	greater than
-2	<	smaller than
-3	+	add
-4	-	subtract
-5	*	multiply
-6	/	divide
-7	=	equal to
-8	>=	greater equal
-9	<=	less equal
-10	**	exponent
-11	><	not equal
-12	@	save
-13	!	fetch
-14	NEG	negation
-15	ABS	absolute value
-16	DUP	Duplicate
-17	SWA	swap
-18	ROT	rotate
-19		reserved
-20		reserved
-21	DIA	diameter
-22	LENGTH	length
-23	TLP	positive tolerance
-24	TLN	negative tolerance
-25	FLATNESS	flatness
-26	TRUE	true position
-27	ROUNDNESS	roundness
-28	ANGULARITY	angularity
-29	PARALLELISM	parallelism
-30	STRAIGHTNESS	straightness
-31	SHAPE	surface shape
-32	SF	surface finish
-33	WIDTH	surface width
-34	HEIGHT	surface height
-35	THEN	
-36	IF	
-37	FEED	feed rate
-38	SPEED	cutting speed
-39	PROCESS	process selected
-40	FREE	surface completed
-41	DLT	tool diameter
-42	MACHINE	machine selected
-43		reserved
.		
-50		reserved

In the current TIPPS process selection module, it sequentially search through the knowledge base for the appropriate process. The system always takes the first process which can be used. The process statements should be arranged according to their priorities. The most frequently used and lest costly process should be put at the top.

The process knowledge must be compiled before TIPPS can use it. The process knowledge is stored in a file called - "PROCESS DATA". To compile it, type "PKI PROC". The compiled object code is stored in file "PROC KNOW". A listing file - "PROC LISTING" shows all the diagnostics.

3.1 Expressions

An expression is a functional element in a statement. There are two classes of expressions, condition and action. In a condition expression, there exist at least one logic operator. Condition operators include =, >, <, >=, <= and ><. When there are more than one logic operators in an expression, they form several sub-expressions. Each expression or sub-expression upon evaluating yields either a "T" or a "F". Sub-expressions in an expression are ored together. There is no limitation on the number of condition expressions in a statement. However all of them must be in an expression starts with a "if". An expression with several sub-expressions is shown in the following:

```
( SHAPE ! DUP 201 = DUP 202 = DUP 203 = 207 = )
```

Meaning:

If shape equals to 201 or 202 or 203 or 207.

In an expression atoms are separated by a null space. In the above example, 201, 202, 203 and 207 are shape code. The shape code can be found in Table 3.2. Since every condition operator will clear the tow number compared on the stack, "DUP" is required in order to save a number for the next comparison. This will be explained in more detail in the next section.

Table 3.2 Surface code

```
                                      +--  simple 101
                                      |
                                      +--  groove 102
                   +-- external ---+
                   |                  +--  spline 103
                   |                  |
                   |                  +--  keyway 104
                   |                  |
                   |                  +--  thread 105
                   |
   cylindrical  -------+
                   |                  +--  simple 111
                   |                  |
                   |                  +--  groove 112
                   |                  |
                   +-- internal ---+--  spline 113
                                      |
                                      +--  keyway 114
                                      |
                                      +--  thread 115
                  +--  flat                    201
                  |
                  +--  step                    202
                  |
                  +--  straight slot           203
                  |
                  +--  dovetail                204
  non-cylindrical ---+
                  +--  V bottom                205
                  |
                  +--  T or Y slot             206
                  |
                  +--  pocket                  207
                  |
                  +--  square hole             208
                  |
                  +--  curved surface          209
```

Action expressions are different than condition expressions. No condition operator is used in action expressions. An action expression can do the following functions:

1. save a process code (Table 3.3) in the process data file.

2. save a tool diameter in the process data file.

3. change the dimension or tolerances of a machined surface.

4. terminate the search process.

Table 3.3 Process name code

Code	Process	Abbreviation
1	twist drill	TWST DRL
2	spade drill	SPD DRL
3	rough bore	RGH BORE
4	finish bore	F BORE
5	rough ream	RGH REAM
6	finish ream	F REAM
7	gun drill	GUNDRL
8	standard tap	STD TAP
9	rough face mill	R F MILL
10	finish face mill	F F MILL
11	rough peripheral mill	R P MILL
12	finish peripheral mill	F P MILL
13	rough end mill	R E MILL
14	finish end mill	F E MILL
15	grinding	GRINDING
16	lapping	LAPPING
17	honing	HONING
18	chamfer bore	CHAMF B

Through the above functions, a satisfied statement ("T" in condition portion can output the corresponding process code and also alter the current surface condition. For example in a reaming statement, the actions include,

1. save the process code - 5.

2. save the current hole diameter as the tool diameter.

3. change the hole diameter to be .002" smaller.

(Note, a backward planning method is employed in the system, a hole is being filled from its final shape to a flat surface.) Then these actions can be written as:

```
( Then
     ( 5 PROCESS @ )
     ( DIA ! DTL @ )
     ( DIA ! 0.002 - DIA @ )
                 .
                 .
)
```

Before the system found a terminate word - "FREE", it will repeatly search the knowledge base. In case none of the statement can be satisfied, an error message is printed. It is important to eliminate endless loop in the knowledge base. In the next section we will discuss how to write an expression.

3.2 PKI Operators

An expression is a combination of atoms. All the literal atoms in PKI are restricted to system words and operators. A post-fix convention is used in PKI. In the post-fix convention, an operator is always placed after operands. e.g. "DIA !", ! is the operator and dia is the operand. In most of textbooks, equations are normally written in an in-fix convention. e.g.

```
x := 5 + 3
```

The same equation in a post-fix convention would be (note, in PKI := is not used)

```
5  3  +  x  :=
```

The second important fact of PKI is the stack operation. All the computations are done on an arithmetic stack (it is called the stack in the following). Whenever a numeral is encountered, it is pushed onto the stack. A system variable however will be pushed onto a second stack - variable stack. Its value needs to be fetched from the appropriate location and pushed onto the stack. The fetch operation is not automatic, a fetch operator "!" needs to be indicated. i.e. "DIA !", value of DIA is fetched and pushed onto the stack. The variable on the variable stack is an address which is used to point the location of the variable value. A general rule is, whenever a variable is used, its value must be fetched by "!" operator. However, if in the same expression a variable is used more than once, Operator "DUP" can be used to save the value on the stack for the second use of the variable. For example, DIA square can be written as "DIA ** 2" or "DIA * DIA". In PKI, the second formula can be expressed as "DIA ! DUP *". It is interpreted as fetch DIA and push onto the stack, then duplicated the first value on the stack (now we have to DIA values on the stack). Finally multiple the two values on the stack and save the result.

```
( DIA ! DUP * )

 step    atom evaluated    variable    stack
---------------------------------------------

                                       ---------+
  1           DIA          DIA=2.0              |
                           LEN=3.5     ---------+
                              .

                              .
                                       ---------+
  2            !                       2.0|  |  |
                                       ---------+

                                       ---------+
  3           DUP                      2.0 |2.0 |
                                       ---------+
                                          ---------+
     4              *                     4.0|  |  |
                                          ---------+
```

In the following, the function of different operators will be discussed.

3.2.1 Mathematical Operators

Mathematical operators always take some operands out-off the stack, after the operation, it push the result back to the stack. Conceptually, a mathematical operator can be thought as a assembly line worker who takes a few items from the top of a box, then those items are assembled. The final assembly is throw back the box. The assembly will become the top item in the box. However, internally an easier method is used which makes the system very efficient.

In the system, mathematical operators include arithmetic operators and logic operators. There are six arithmetic operators - +, -, *, /, **. A function ABS is also available. They can satisfy most of the computational needs. Following table will show how these operator change the stack.

Prior to the operation the stack reads:

```
---+---+---+------+
 x | y | z |......|
---+---+---+------+
```

```
 Operator                  Stack (after operation)
-------------------------------------------------

                       ------+---+---------+
    +                   x+y  | z |.........|
                       ------+---+---------+

                       ------+---+---------+
    -                   y-x  | z |.........|
                       ------+---+---------+
                       ------+---+----------+
  *                     x*y  | z |.........|
                       ------+---+----------+

                       ------+---+----------+
  /                     y/x  | z |.........|
                       ------+---+----------+

                       ------+---+----------+
  **                    y**x | z | ........|
                       ------+---+----------+

                       ------+---+----------+
  ABS                  ABS(x)| y | z |.....|
                       ------+---+----------+
```

There are six logic operators. Logic operators do not push the result back to the stack, instead it set or reset a logic flag. When the result is "T", the flag equals to 1, otherwise it equals to 0.

The stack prior to the operation reads:

```
 ---+---+---+--------+
  5 | 4 | 9 | ..... |
 ---+---+---+--------+
```

```
 Operator          Stack (after)        Flag
------------------------------------------------

                    ---+-----------+
    >                9 | ........ |        0
                    ---+-----------+

                    ---+-----------+
    <                9 | ........ |        1
                    ---+-----------+

                    ---+-----------+
    =                9 | ........ |        0
                    ---+-----------+

                    ---+-----------+
    >=               9 | ........ |        0
                    ---+-----------+
                          ---+----------+
      <=                   9 | ........ |          1
                          ---+----------+

                          ---+----------+
      ><                   9 | ........ |          1
                          ---+----------+
```

Note, the original stack is created by the expression (9 4 5).

3.2.2 Stack Operator

The concept of stack operation has been discussed earlier. Here the definitions of some stack operators will be shown.

The stack prior to the operation reads:

```
                  ---+---+---+-------+
                   x | y | z | ..... |
                  ---+---+---+-------+
```

```
         Operator            Stack
     --------------------------------------------

                             ---+---+---+---+------+
         DUP                  x | x | y | z | .... |
                             ---+---+---+---+------+

                             ---+---+---+----------+
         SWA                  y | x | z | ........ |
                             ---+---+---+----------+


                             ---+---+---+----------+
         ROT                  y | z | x | ........ |
                             ---+---+---+----------+
```

Another two stack operators ! and @ are different than the above three. The function of ! has been discussed previously. @ is in opposite to !. It's function is shown in the following diagram.

Prior to the operation:

```
          variable stack
          -----+-------+
           DTL |...... |
          -----+-------+        DTL (memory location)
                              +-------+
                              |       |
          the stack           +-------+

          ---+---+---+-------+
           x | y | z | ..... |
          ---+---+---+-------+
```

after the @ operation:

```
          variable stack
          -----+-------+
           DTL |...... |
          -----+-------+        DTL (memory location)
                              +-------+
                              |   x   |
          the stack           +-------+
          ---+---+-----------+
           y | z | ..........|
          ---+---+-----------+
```

3.2.3 <u>Process Description with PKI</u>

Process capabilities can be represented by their boundaries. A typical process boundary for twist drilling can be written as:

```
l  <  12.0 d
0.0625 < d  < 2.000
t  >  0.007  d
t  > 0.007  d  + 0.003
straightness  >  0.0005 ( l / d )  + 0.002
roundness  >  0.004
parallelism  > 0.001   ( l / d ) + 0.003
true position  >  0.008
surface finish > 100
```

Such process boundaries can be written in PKI as:

```
( IF
( ( IF
      ( SHAPE ! 110 = )
      ( LENGTH !  12.0  DIA ! * < )
      ( DIA ! 0.0625 < )
      ( DIA ! 2.0000 > )
      ( TLP ! 0.007 DIA ! * > )
      ( TLN ! 0.007 DIA ! * 0.003 + > )
      ( STRAIGHTNESS !  0.0005  LENGTH ! DIA ! / *
                    0.002 + > )
      ( ROUNDNESS ! 0.004 > )
      ( PARALLELISM ! 0.001  LENGTH ! DIA ! / *
                    0.003 + > )
      ( TRUE ! 0.008 > )
      ( SF ! 100 > )
  )
  ( THEN
      ( 1 PROCESS @ )
      ( DIA ! DTL @ )
      ( FREE        )
  )
)
```

A complete set of processes described in PKI can be found in the appendix.

3.3 Process Knowledge Base Structure

The entire process knowledge base is stored in a single linked list. Each cell is a two tuple (a,b), which takes two integer words. "b" is always either a pointer or a terminator (0). "a" can be a pointer, an operator, a variable, or an address of a constant in the constant value array. Constant is stored outside the linked list in order to avoid the confusion between constant value and pointers. An atom takes exactly one cell. "a" is for the atom itself and "b" is for the linking to other atoms.

```
          +-----+-----+
  ------>| DIA |     |--->
          +-----+-----+
```

--> represent pointer

DIA is represented by a code which has a negative value (refer to Table 3.1). A negative "a" always imply it is not a pointer. When the "a" value is less than zero but greater than -50, it is a system word or operator. If the "a" value is smaller than -50, then it is a constant. The constant is stored in the constant array at the location (-50-a). An expression consists of some linked atoms with a terminator. For example "(DIA ! 2.0 <)" is an expression, its structure is:

```
+-----+--+
|     |  |
+-----+--+
   |
   | (
   |
+-----+--+  +--+--+  +--+--+  +--+--+
| DIA |  |-->| ! |  |-->|2.0|  |-->| < | 0|  )
+-----+--+  +--+--+  +--+--+  +--+--+
```

The left and right parentheses represent pointers. "(" represents a pointer "a" which points to the expression body. ")" represents a terminator in "b". An expression can be considered as a branch in a tree. The leftest pointer -> in the above example is the left parenthesis in the expression (DIA ! 2.0 <). The last zero in the expression structure is the right parenthesis. A complete statement can be represented by a tree. Where the first "(" is the root pointer.

```
+---+--+
|   |  |----> to next statement
+---+--+
  | (
  |
+---+--+       +----+--+   +---+--+   +--+--+
|   |  |------>|THEN|  |-->|   |  |-->|  | 0|  )
+---+--+       +----+--+   +---+--+   +--+--+
  |                          |          |
  | (                        |      action 2
  |                      action 1
  |
+---+--+    +---+--+   +--+--+   +--+--+   +--+--+
| IF|  |--->|   |  |-->|  |  |-->|  |  |-->|  | 0|  )
+---+--+    +---+--+   +--+--+   +--+--+   +--+--+
              |          |         |         |
              |          |         |     condition 4
              |          |     condition 3
              |      condition 2
          condition 1
```

the above structure represents:

```
      (
          ( IF
                 ( condition 1 )
                 ( condition 2 )
                 ( condition 3 )
                 ( condition 4 )
          )
          ( THEN
                 ( action 1 )
                 ( action 2 )
          )
      )
```

The entire process knowledge base looks like a vine. Statements - small branches, come out of the main vine. The branch closest to the root of the vine is of the most important. It gets the nutrient first. The following diagram shows the structure of the process knowledge base.

```
  +--+--+   +--+--+   +--+--+   +--+--+   +--+--+
  | 0|  |-->|  |  |-->|  |  |-->|  |  |-->|  |  |-->
  +--+--+   +--+--+   +--+--+   +--+--+   +--+--+
   |          |         |         |         |
   |          |         |         |     statement
   |          |         |         |    for rgh bore
   |          |         |      f. ream
   |          |     rgh ream
   |      spade drl
   |
twst drl
```

The entire process knowledge base is stored in a one dimensional integer array and a real array. The integer array stores the linked list, and the real array stores the constant values. The size of the arrays are determined by the number of expressions in the knowledge base.

4. TIPPS DATA AND FILE STRUCTURES

In this part, detailed data and file structures are presented.

4.1 CAD Data Input Format

An 80 column card image file (p-name DATA) is used to store the CAD data. Data can be created by another CAD system then translated into this format through a postprocessor. In the file, the first record is used to store the header information. The header information consists of the part name and part number. Each takes twenty columns. From the second to the end of the file, geometric and technological data are stored.

Each input record (one card) is divided into fields.

Column number	Contains	
1 - 5	code	I5
6 - 9	field 1	F4.2
11 - 15	field 2	F5.2
16 - 20	field 3	F5.2
21 - 25	field 4	F5.2
26 - 30	field 5	F5.2
31 - 35	field 6	F5.2
36 - 40	field 7	F5.2
41 - 45	field 8	F5.2
46 - 50	field 9	F5.2

In the following, different record types are discussed.

4.1.1 100 Material

Syntax:

100 material type code BHN number

material type:

1x		cast iron
	11	grey cast iron
	12	ductile cast iron
	13	malleable cast iron
	14	alloys
2x		steel
	20	carbon steel
	21	alloy steel
	22	stainless steel
30		special purpose steel
40		aluminum
	41	aluminum alloy
50		brass
60		other materials

4.1.2 <u>110</u> <u>Plane Face</u>

Syntax:

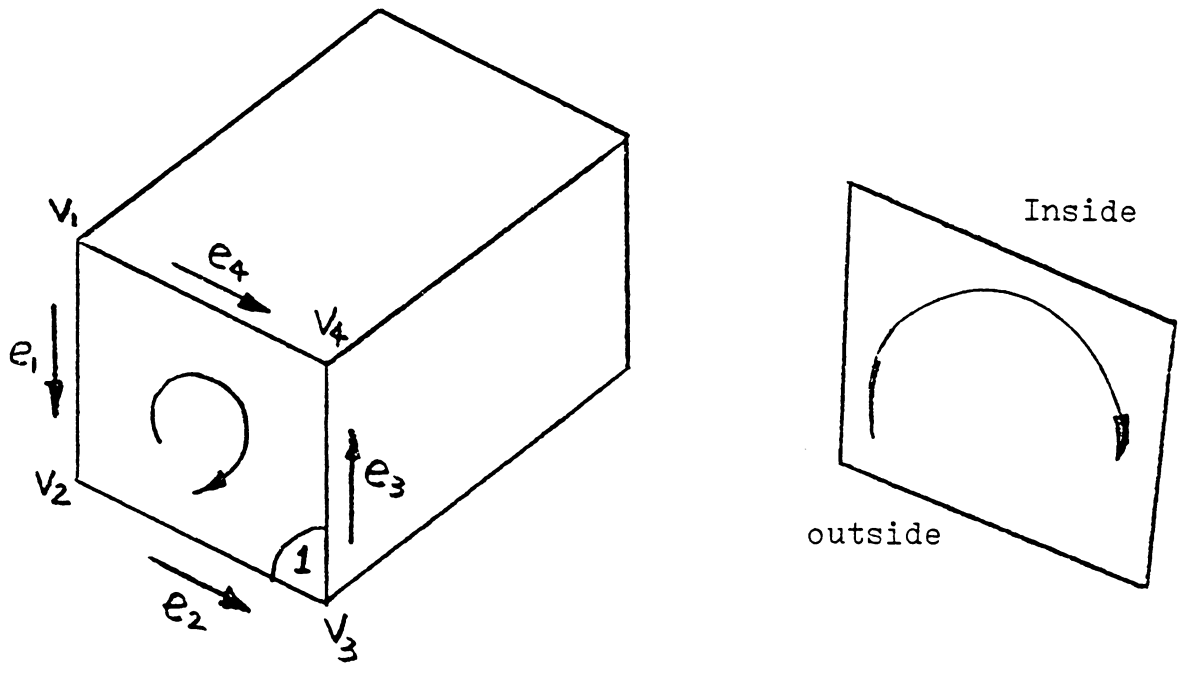
110 I N P_1 P_2 ... P_n $P_{f.g.}$

I : face number

N : number of edges

P_i: pointer locates the edge $1 \leq i \leq n$

Edges are arranged in sequence. Right-hand rule is applied to show the direction of inside or outside of the component. When the direction of a edge is reversed, minus sign is used on P_i to indicate the correct direction (Figure 4.1).



Code	I	N	$P_1P_2P_3P_4P_{f.g.}$			
110	1	4	-1	4	-3	-2

Figure 4.1 Face representation

$P_{f.g.}$: pointer locates the geometric tolerance.

4.1.3 <u>120</u> <u>Cylindrical Face (Figure 4.2)</u>

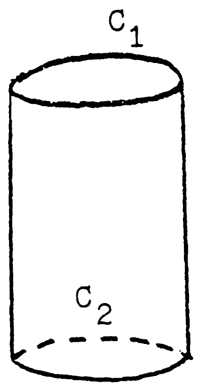
Syntax:

120 I UC BC $P_{f.g.}$

I : face number

UC : points to the upper circle

BC : points to the bottom circle



Code	I	UC	BC
120	1	1	2

Figure 4.2 cylindrical face

4.1.4 130 Hole

Syntax:

130 I TYPE Ph DIA L + - P_{fg}

I : hole number. for the top hole segment only.

Following hole segments have value 0.

TYPE : hole type

1 : simple hole segment

2 : simple linear chamfer

3 : inner fillet

4 : outer fillet

5 : thread

Ph : pointer to the reference point.

(the top center point).

DIA : diameter.

For thread, this is the thread series

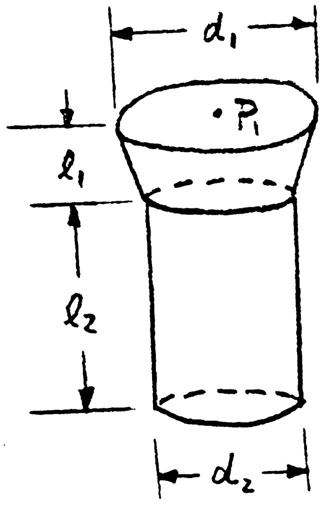
1 : NC

2 : NF

L : length (inch)

+,- : tolerance on length (.001 inch)

For thread, field for + is the number of threads per inch.



Hole #1

Code	I	TYPE	Ph	DIA	L	+	-	P_{fg}
130	1	2	1	d_1	L_1	10	10	
130	0	1	1	d_2	L_2	10	10	

Figure 4.3 Hole

4.1.5 140 - 200 Reserved

4.1.6 210 Edge (Line segment)

Syntax:

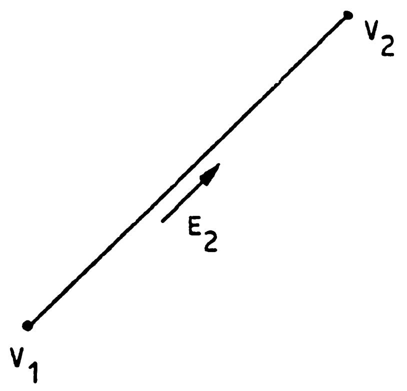
210 I V_1V_2

I : edge number

V_1 : vertex 1 (point 1)

V_2 : vertex 2

Direction of the edge is from V_1 to V_2.



<u>Code</u>	<u>I</u>	V_1	V_2
210	2	1	2

Figure 4.4 Edge

4.1.7 <u>220</u> <u>Circle</u>

Syntax:

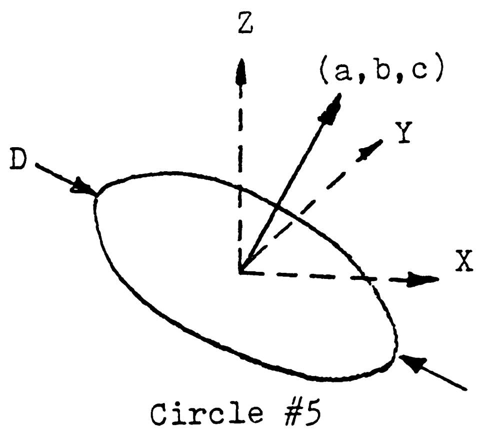
220 I P DIA a b c

I : circle number

P : center

DIA: diameter

a,b,c : unit vectors show the orientation of the circle.



Code	I	P	DIA	a	b	c
220	5	1	d	a	b	c

Figure 4.5 Circle

4.1.8 <u>230</u> <u>Spline</u>

Syntax:

230 I TYPE N P_1, . . P_n

I : spline number

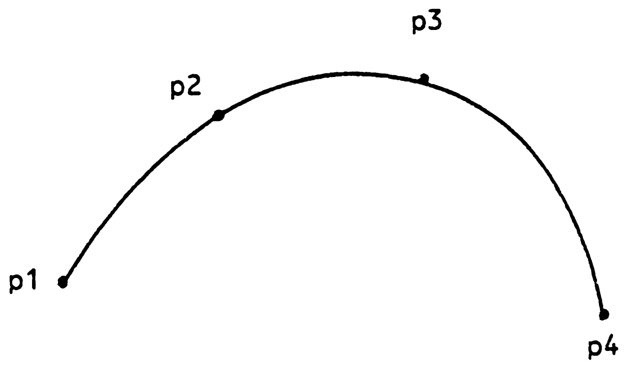
TYPE: types of spline

=1 : Bezier

=2 : B-spline

N : number of control points

$P_1..P_n$: control points



Code	I	TYPE	N	P_1	P_2	P_3	P_4
230	1	1	4	1	2	3	4

Figure 4.6 Spline

4.1.9 240 - 300 Reserved

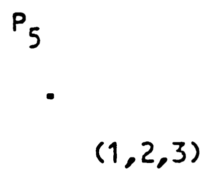
4.1.10 310 Vertex (point)

Syntax:

310 I X Y Z

I : vertex number

X,Y,Z : coordinate



code	I	X	Y	Z
310	5	1.0	2.0	3.0

Figure 4.7 Vertex

4.1.11 500 Technological Information

Syntax:

```
500  I  +  -  SF  T.P.  R   PARA   STRA
                   F    A
```

I : tolerance number

+,- : dimension tolerance

SF : surface finish

T.P.: true position

F : flatness

R : roundness

A : angularity

PARA : parallelism

STRA : straightness

A complete input data set for the example illustrated in part two is listed in the following.

Figure CAD data

```
TEST PART - BPACKET P-10001
   100   2  200
   110   1    8    1    2    3    4    5    6    7   -8
   110   2    4    9   10  -11   -6    1
   110   3    4   13  -12   -9   -5
   110   4    4  -14  -13   -4  -15    2
   110   5    8   21  -20  -19  -18  -17  -16  -15   -3
   110   6    4   -2   23  -22  -21    3
   110   7    4  -24   17  -26   25    3
   110   8    4   18   28   27  -26
   110   9    4   29  -28   19   30    4
   110  10    4  -20   30  -31  -22
   110  11    8   -1   33  -32  -25  -27  -29  -31  -23
   110  12    8  -10   12   14   16   24   32   36  -34
   110  13    4    7   35  -34  -11
   110  14    4   -8   33   36  -35
   210   1    1    2
   210   2    2    3
   210   3    3    4
   210   4    4    5
   210   5    5    6
   210   6    6    7
   210   7    7    8
   210   8    1    8
   210   9    6    9
   210  10    9   10
   210  11    7   10
   210  12    9   11
   210  13    5   11
   210  14   11   12
   210  15    4   12
   210  16   12   13
   210  17   13   14
   210  18   14   15
   210  19   15   16
   210  20   16   17
   210  21    3   17
   210  22   17   18
   210  23    2   18
   210  24   13   22
   210  25   21   22
   210  26   14   21
   210  27   20   21
   210  28   15   20
   210  29   19   20
   210  30   16   19
   210  31   18   19
   210  32   22   23
   210  33    1   23
   210  34   10   24
   210  35    8   24
   210  36   23   24
   310   1    0    0    0
   310   2    0    0    3
   310   3    2    0    3
```

```
310    4    2    0    2
310    5    6    0    2
310    6    6    0    4
310    7    8    0    4
310    8    8    0    0
310    9    6    4    4
310   10    8    4    4
310   11    6    4    2
310   12    2    4    2
310   13    2    4    3
310   14    2    25   3
310   15    2    25   26
310   16    2    2    26
310   17    2    2    3
310   18    0    2    3
310   19    0    2    26
310   20    0    25   26
310   21    0    25   3
310   22    0    4    3
310   23    0    4    0
310   24    8    4    0
310   25    4    2    2
130    1    2   25    12     5   10    10      5
130    0    1   25    10    1    10    10      6
500    1    5    5   50     5     5  100      5
500    2    5    5   50     5     5  100      5
500    3    5    5   50     5     5  100      5
500    4    5    5   5      5     5  100      5
500    5    5    5  120     5     5  10       5
500    6    11   11  20    100  100   100     5
999
```

4.2 Internal Data Structure

In this section the data structure used in the program will be presented. The arraies which store the corresponding data will also be explained.

4.2.1 CAD Data

PFACE(j): face pointer array

j: face number

NF: number of faces

FACE(j) : face data link list, 5 tuple.

1 : number of edges

2 to N+1 : edge pointers

N+2 : pointer to geometric tolerance

LIN(i,j) : edge data array

LIN(1,j) : starting vertex

LIN(2,j) : ending vertex

j : edge number

NL : number of edges

POINT(i,j) : vertices data array

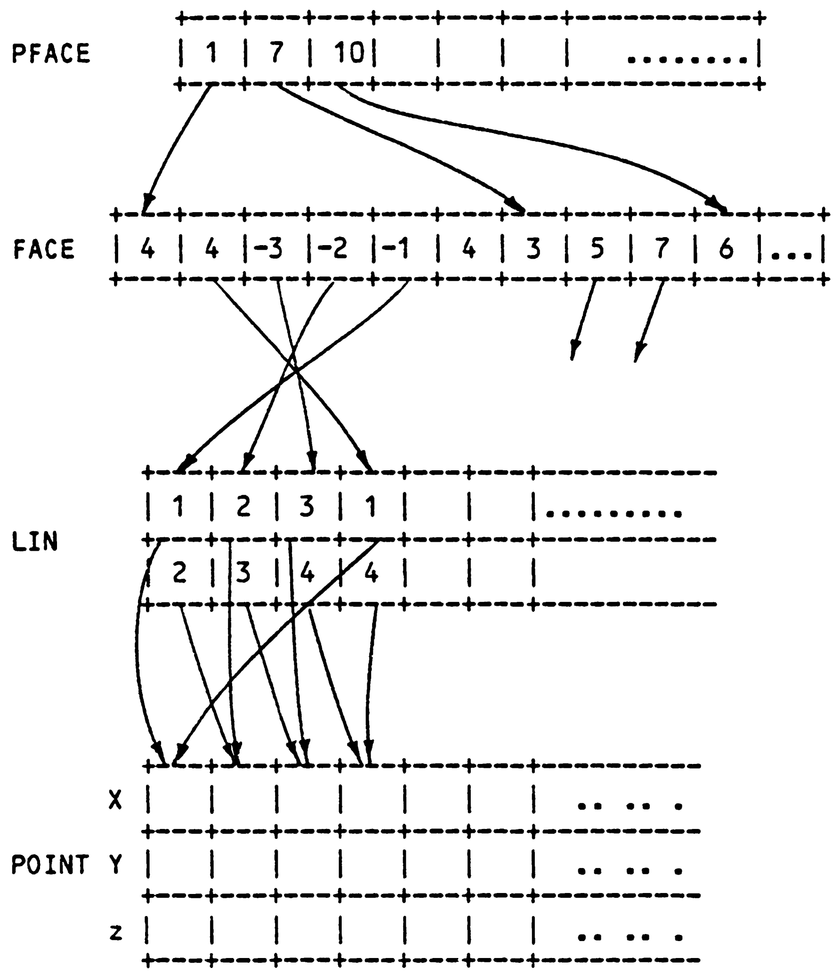
POINT(1,j) : x

POINT(2,j) : y

POINT(3,j) : z

j : vertex number

NP : number of vertices



HL(i) : hole pointer array

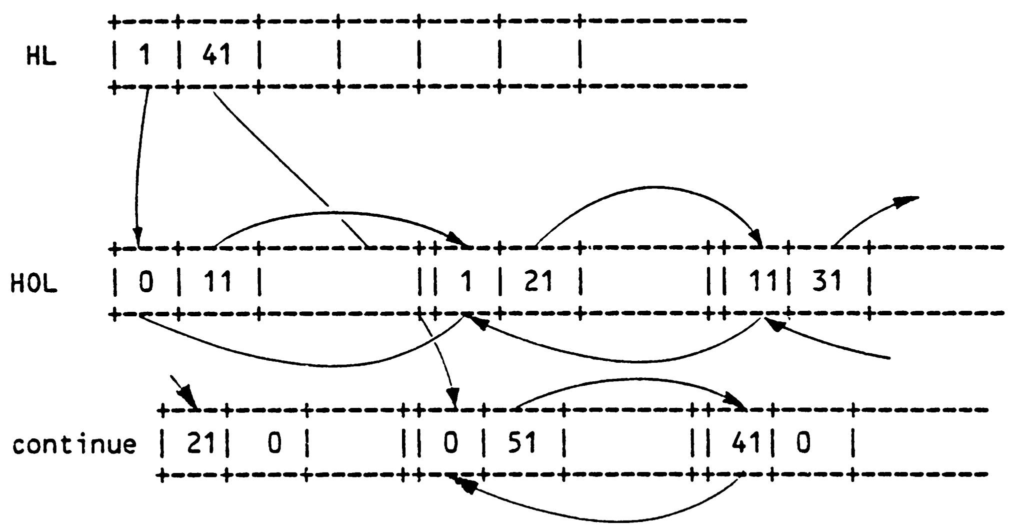
i : hole number

IH : number of holes

HOL : hole data linked list, 10 tuple field

1	:	backward pointer
2	:	forward pointer
3	:	hole type
4	:	reference point pointer

5	:	diameter
6	:	length
7	:	positive tolerance
8	:	negative tolerance
9	:	geometric tolerance pointer



4.2.2 Surface ID

IPSUR(1,j) : surface type

IPSUR(2,j) : pointer to nsur

j : surface number

NSUR : linked list which stores the surface information each recode is a four tuple.

field one: backward pointer

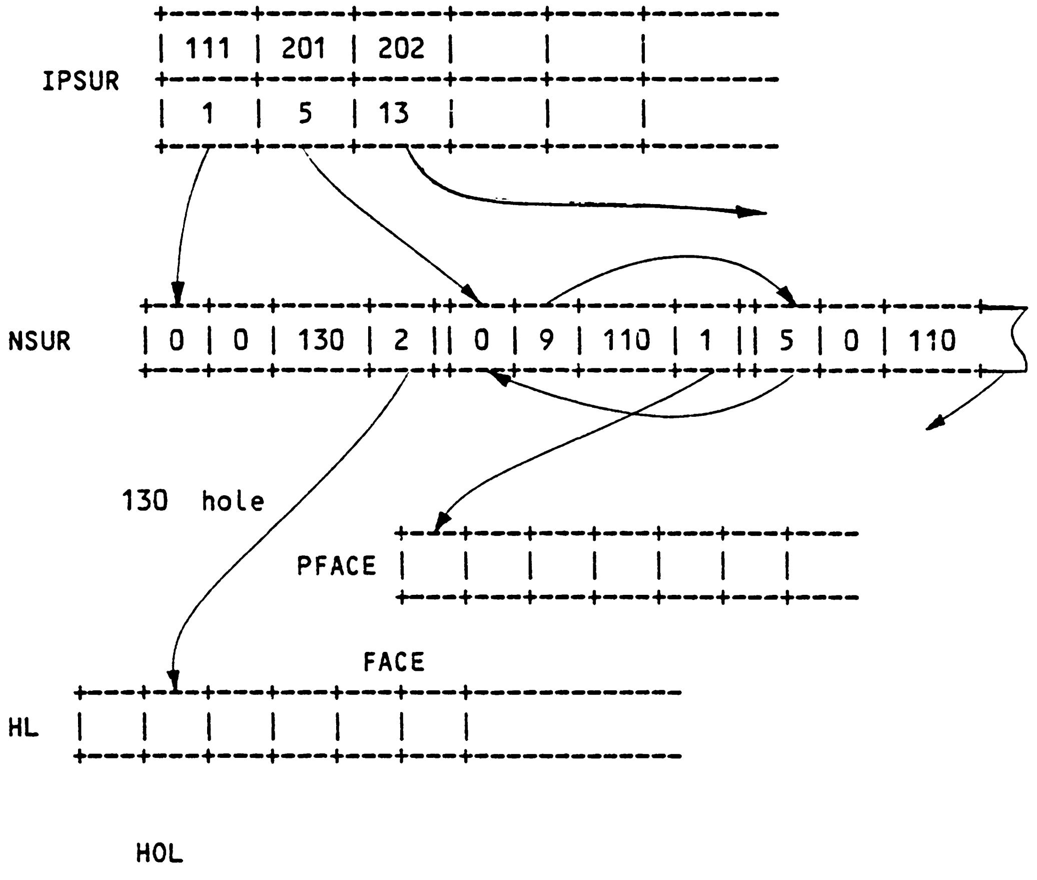
field two: forward pointer, to next face in the same machined surface.

field three: face type, see CAD data.

field four: pointer to CAD data.

$$
\begin{array}{cc|c|c|c|}
 & & 1 & 2 & 3 \\
\cline{3-5}
 & 1 & 0 & 1 & 0 \\
\cline{3-5}
\text{IRSUR} & 2 & 0 & 0 & 0 \\
\cline{3-5}
 & 3 & 1 & 0 & 0 \\
\cline{3-5}
\end{array}
$$

relationship matrix
(s_3 must be done before s_1; s_1 must be done before s2)



4.2.3 Process Data

IPPRC(i) : process pointer array

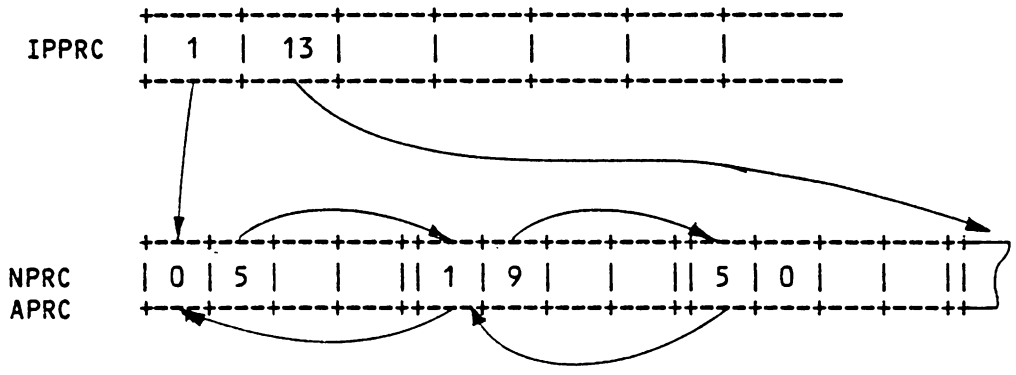
i : surface number

NPRC, APRC : process linked list. they are equivalent. 4 tuple

field

1 :	backward pointer
2 :	forward pointer
3 :	process code
4 :	tool diameter

A minus process code is used to separate the processes for two hole segments.



4.2.4 Parameter File

IPARA(i) : process parameters pointer array

i : surface number

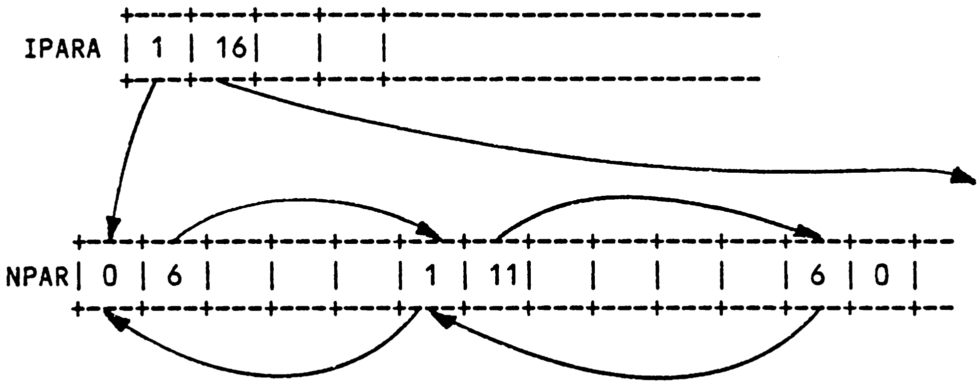
NPAR, APAR : equivalent parameter linked list, 5 tuple

field

1 :	backward pointer
2 :	forward pointer
3 :	feed

4 : speed

5 : time



4.3 Data Files Associated with Each Program

Following are list of data files needed for each program. IBM CMS exec files for each program is also listed in the appendix. The same information can be found in both place.

4.3.1 TIPPS

p-name : user assigned part name.

UNIT NO	FILE NAME	FILE TYPE	RECORD LENGTH	I/O
4	p-name	DATA	80	I
5	(terminal)			
6	(terminal)			
8	p-name	SID	80	0
10	PRNAM	DATA	80	I

11	p-name	PROC	80	O/I

p-name DATA : see Section 4.1

p-name SID : see Section 4.2.2

Output by SSID in a unformated file. input by LDSUR.

p-name PROC : see Section 4.2.3

Output by SPROC, input by LDPRO.

PRNAM DATA : see Table 3.3

Stores process codes and abbreviations. Input by RPCN.

4.3.2 PAESEL

UNIT NO	FILE NAME	FILE TYPE	RECORD LENGTH	I/O
2	TOOL	DATA	80	I
4	p-name	DATA	80	I
5	terminal			
6	terminal			
8	p-name	SID	80	I
10	prnam	DATA	80	I
11	p-name	PROC	80	I
12	p-name	LISTING	131	O
13	p-name	PAR	80	O
17	PROC	KNOW	80	I

TOOL DATA : Format f10.5

field

1	CCNCA	clearance required for tool approach
2	CCNCT	clearance required for tool breakthrough
3	TTLFDR	drill tool life
4	TTLFBR	bore tool life
5	TTLCNG	machine tool change time
6	TTLFRM	ream tool life
7	TTLCEF	operator tool change time
8	SSTEP	step down size

p-name PAR : see Section 4.2.4

PROC KNOW : see part 2.

4.3.3 REPORT

UNIT NO	FILE NAME	FILE TYPE	RECORD LENGTH	I/O
4	p-name	DATA	80	I
5	terminal			
6	terminal			
8	p-name	SID	80	I
10	prnam	DATA	80	I
11	p-name	PROC	80	I
12	p-name	LISTING	131	0
13	p-name	PAR	80	0

4.3.4 <u>PKI</u>

UNIT NO	FILE NAME	FILE TYPE	RECORD LENGTH	I/O
3	PROC	KNOWL	80	O
5	PROC	DATA	80	I
6	PROC	LISTING	130	O

PROC DATA : process knowledge description file.

4.4 Subroutine Calling Sequence

Refer to the program listing for program descriptions. Each box in the diagram represents a subroutine. The top one represents a main program. When the system is loaded, be sure to include all the subroutines in. The system should be divided into four parts: TIPPS, PARSEL, REPORT and PKI. However, further division is also possible. In the following diagrams, TIPPS is represented by four diagrams. The main purpose of separting them is to show the differences among those functions. In realaty they are all in one program.

4.4.1 TIPPS Initialization

```
                     +----------+
                     | TIPPS    |
                     +----------+
                           |
       +----------+--------+---------+---------+---------+
       |          |        |         |         |         |
  +--------+  +-------+    |    +--------+ +------+  +-------+
  | SETPRC |  | SETSID|    |    | GINPT  | | FIND |  | PKINPT|
  +--------+  +-------+    |    +--------+ +------+  +-------+
                           |
         +-----------------+------------------+
         |                 |                  |
     +-------+ +-----------------------+ +--------+
     | COMPR | |    PLOT-10 LIBRARY    | | SEENUM |
     +-------+ |                       | +--------+
               |  INITT  ERASE TSEND   |
               |  NEWPAG NEWLIN        |
               +-----------------------+
```

4.4.2 2-D GRAPHICS

```
                     +----------+
                     | TIPPS    |
                     +----------+
                           |
                           |
      +-------------+------+------------+---------------+
      |             |                   |               |
      |             |                   |               |
  +--------+    +--------+         +-------+       +---------+
  | BOUND  |    |  P2D   |         | SCALPT|       |  RNG2D  |
  +--------+    +--------+         +-------+       +---------+
     |   |        |    |            |    |
 +-----+ +--------+-----------------+    +---------+
 | TIC | |             |                           |
 +-----+ |             |                +----------+-----------+
         |       +--------+             |          |           |
         |       |  V2D   |        +-------+   +-------+  +------+
         |       +--------+        | SEENUM|   | COMPR |  | FIND |
         |         |  |  |         +-------+   +-------+  +------+
         |         |  |  |
         +---------+  |  +---------+
         |            |            |
         |            |       +--------+
         |            |       | PHOLE  |
         |            |       +--------+
         |            |         |  |  |
         |            +---------+  |  +---+
         |            |            |      |
         |     +-----------+  +------+    |
         |     | WINDOW    |  | PCIR |    |
         |     +-----------+  +------+    |
         |                                |
         +----------------+---------------+
                          |
                          |
               +---------------------+
               |  PLOT-10 LIBERARY   |
               |                     |
               | NEWPAG     DCURSR   |
               | TSEND      MOVEA    |
               | MOVABS     DRAWA    |
               | DRWABS     DASHA    |
               | CSIZE      SWINDO   |
               | ANMODE     VWINDO   |
               +---------------------+
```

4.4.3 2-D GRAPHICS

```
                 +---------+
                 | TIPPS   |
                 +---------+
                       |
                       |
                 +----------+
                 |  P3D     |
                 +----------+
                       |
     +---------+-------+--------+-----------+
     |         |       |        |           |
     |         |       |        |           |
+---------+ +------+   |   +--------+       |
| PERSP   | | BND  |   |   | V3D    |       |
+---------+ +------+   |   +--------+       |
               |       |     |   |          |
               +-------+-----+   |          |
               |                 |          |
               |                 |          |
               |            +--------+      |
               |            |  P3DH  |      |
               |            +--------+      |
               |             |    |         |
               +-------------+    |         |
               |                  |         |
               |            +---------+     |
               |            |  P3DC   |     |
               |            +---------+     |
               |              |    |        |
               +--------------+    +--------+
               |                            |
      +-----------------------+       +---------+
      |    PLOT-10 LIBERARY   |       | TRANS   |
      |                       |       +---------+
      | NEWPAG  TSEND SWINDO  |
      | VWINDO  MOVABS MOVEA  |
      | DRAWA   DASHA         |
      +-----------------------+
```

4.4.4 Surface Marking and Editing

```
                     +-----------+
                     |  TIPPS    |
                     +-----------+
                          |
    +---------------------+------------------+
    |                                        |
    |                                  +----------+
    |                                  | SIDED    |
    |                                  +----------+
    |                                    |     |
    +------------------------------------+     |
    |                                          |
+----------+                                   |
|  SID     |                                   |
+----------+                             +-----+----+
  |    |                                 |          |
  |    |                                 |    +-----+-+
  |    +--------+                        |    |       |
  |    |        |                        | +------+   |
  |    |        +-------+------+---------+ | BND  |   |
  | +---------+ |       |      |         | +------+   |
  | | RELTON  | |       |  +-------+     |      +--------+
  | +---------+ |       |  | GID   | +------+   | PSUR   |
  |      |      |       |  +-------+ | DREF |   +--------+
  |      |      |   +---------+      +------+        |
  | +---------+ |   | CREF    |          |    +--------+
  | | CHYT    | |   +---------+          |    | P3DC   |
  | +---------+ |      |   |             |    +--------+
  |             |      |   +-------------+         |
  |        +--------+  +------+          |         |
  |        |  P3D   |         |          |         |
  |        +--------+     +------+       |         |
  |  (SEE 3-D GRAPHICS)   | TRANS|       |         |
  |                       +------+       |         |
  |                                      |         |
  +--------------------------------------+---------+
                                         |
                   +----------------------------+
                   |  PLOT-10 LIBERARY          |
                   |                            |
                   | HOME  ERASE  TSEND  NEWLIN |
                   | CARTN MOVABS VCURSR SWINDO |
                   | MOVEA VWINDO DRAWA  AOUTST |
                   | BELL  CZAXIS ANMODE        |
                   +----------------------------+
```

4.4.5 Process Selection

```
                                +------------+
                                |   TIPPS    |
                                +------------+
                                      |
                                      |
                                +------------+
                                |   PROCSL   |
                                +------------+
                                      |
        +-------------+---------+-----+----------------+
        |             |         |                      |
   +--------+    +---------+    |               +---------+
   |  FAS   |    | LDSUR   |    |               |  OUTPUT |
   +--------+    +---------+    |               +---------+
                                |
                         +-----------+                |
                         |  PKS      |                |
                         +-----------+                |
                                |                     |
              +--------+--------+---------+           |
              |        |        |         |      +--------+
          +--------+   |   +--------+ +-------+  | LAS    |
          | PUSH   |   |   | POP    | | FETCH |  +--------+
          +--------+   |   +--------+ +-------+
                       |
                  +---------+
                  |  STKOP  |
                  +---------+
```

4.4.6 Parameter Selection

```
                          +------------+
                          |   PARSEL   |
                          +------------+
                                |
      +--------+-------+--------+--------+-------+--------+
      |        |       |        |        |       |        |
  +---------+  |  +----------+  |   +--------+   |  +--------+
  |  SPAR   |  |  |  INTERP  |  |   | BORING |   |  |  LWH   |
  +---------+  |  +----------+  |   +--------+   |  +--------+
               |        |       |        |       |        |
        +--------+ +--------+   |   +--------+   |  +--------+
        | SPROC  | |  LAS   |   |   | BORPAR |   |  |  XYZ   |
        +--------+ +--------+   |   +--------+   |  +--------+
                                |                |
     +--------+-------+---------+-------+   +----+--------+
     |        |       |         |       |   |             |
  +--------+  |  +---------+    | +-------+ |       +--------+
  | GINPT  |  |  |  RPCN   |    | | LDSUR | |       |  HINF  |
  +--------+  |  +---------+    | +-------+ |       +--------+
     +--------+------+   +------+------+  +--------+
     |        |      |   |      |      |  | MILPAR |
  +--------+  | +------+ |  +-------+  |  +--------+
  | LDPRO  |  | | TAPAR| |  | SETPAR|  |
  +--------+  | +------+ |  +-------+  |
              |          |             |
       +---------+  +--------+   +----------+
       |  PARAM  |  | TOLSZ  |   |  RMPAR   |
       +---------+  +--------+   +----------+
```

4.4.7 Report

```
                       +----------+
                       |  REPORT  |
                       +----------+
                            |
                            |
  +------+------+------+----+---+------+-------+-------+
  |      |      |      |        |      |       |       |
+------+ | +--------+  |  +---------+  |   +--------+  |
| LAS  | | | SURNAM |  |  | GINPT   |  |   | HINF   |  |
+------+ | +--------+  |  +---------+  |   +--------+  |
         |             |               |               |
   +---------+         |               |       +-------+----+
   |         |     +-------+    +--------+     |            |
+------+ +-------+ |  LWH  |    | LDPRO  |   +------+ +------+
| HINF | | RPAR  | +-------+    +--------+   | LDSUR| | RPCN |
+------+ +-------+     |                     +------+ +------+
                   +-------+
                   | XYZ   |
                   +-------+
```

4.4.8 Process Knowledge Interpretor

```
                                 +-----------+
                                 |  PKI      |
                                       |
                                       |
                   +-------------------+---------------+
                   |                                   |
            +------------+                      +----------+
            |  PROIN     |                      |  PLIST   |
            +------------+                      +----------+
                   |                                   |
                   +------------------+                |
                   |                  |                |
            +------------+      +---------+            |
            |  SEENUM    |      | STORE   |            |
            +------------+      +---------+            |
                   |              |    |               |
                   |              |    +--+---------+--+
             +-----------+  +--------+    |         |
             |  FIND     |  | ICOMP  |    |     +--------+
             +-----------+  +--------+    |     |  PUSH  |
                                          |     +--------+
                                     +---------+
                                     |  POP    |
                                     +---------+
```