Order Number 9011041

Computer-aided simultaneous engineering for components manufactured in small and medium lot-sizes

Subramanyam, Sridhar, Ph.D.

University of Illinois at Urbana-Champaign, 1989

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

COMPUTER-AIDED SIMULTANEOUS ENGINEERING FOR COMPONENTS MANUFACTURED IN SMALL AND MEDIUM LOT-SIZES

BY

SRIDHAR SUBRAMANYAM

B.Tech., Indian Institute of Technology, 1983
M.S., University of Iowa, 1985

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1989

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

AUGUST 1989

WE HEREBY RECOMMEND THAT THE THESIS BY

SRIDHAR SUBRAMANYAM

ENTITLED COMPUTER-AIDED SIMULTANEOUS ENGINEERING

FOR COMPONENTS MANUFACTURED IN SMALL AND MEDIUM LOT-SIZES

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

Stephen Lu
Director of Thesis Research

A. L. Addy
Head of Department

Committee on Final Examination†

Stephen Lu
Chairperson

James H. Garrett, Jr.

David Kuhn

R. E. Miller

† Required for doctor's degree but not for master's.

O-517

Abstract

Simultaneous Engineering addresses the issue of developing the lowest cost design of a part by concurrently taking into consideration different product life-cycle concerns during the product development process. When only functional, structural, and machining life-cycle concerns are considered, Simultaneous Engineering entails concurrent product and process design. The objective of this research is to study and propose a methodology for the simultaneous product and process design of components manufactured in small and medium lot-sizes. For such components, Simultaneous Engineering is the process of ensuring that these parts are manufacturable for the lowest possible cost in specially designed facilities such as manufacturing cells.

A computer-based design environment encapsulating the expertise of process planners has been developed to cooperatively assist designers in developing manufacturable product designs. Model-based reasoning is the fundamental methodology employed to develop the framework of this design environment. Explicit models to describe product designs and manufacturing facilities have been developed to support the reasoning process. A combination of feature-based, geometric, and process performance models is proposed to satisfy the basic requirements. The reasoning subsystem is based on the Multiple Cooperative Knowledge Sources (MCKS) paradigm, with explicit separation of domain and control knowledge. Domain knowledge sources deal with the product and process refinement activities, and a control knowledge source deals with managing the concurrency between these two tasks. Appropriate justifications are provided in this research for the modeling and reasoning mechanisms chosen. The design environment has been tested and validated by

implementing it for the simultaneous product and process design of Bearing Cages. Several observations made about the general characteristics of the design environment, its impact on the product and process designer, and the unique nature of the knowledge acquisition task are discussed in detail and suitable recommendations are provided for future research directions.

Acknowledgements

Completion of this thesis would not have been possible without the help and encouragement of numerous people.

I wish to thank my advisor Professor Stephen Lu for his support, supervision, constant encouragement, and personal involvement in this research work. I am also grateful to the other three members of my thesis committee, Professors Richard DeVor, James Garrett, and David Knapp, for sharing their expertise in suggesting research directions, providing strong guidance, and unflinching criticism.

This study was made possible through research grants from Allied Signal Aerospace, Caterpillar Inc., and National Science Foundation.

I would like to thank Karl Arnold and the process planners at the Kansas City Division of Allied Signal Aerospace for their assistance in providing insights into the Operation Planning task.

Thanks are also due to John Armstrong, Jim Carter, Tim Applegren, Ron Owden, and Donna Murr from the Artificial Intelligence Group at Caterpillar Inc., for helping make the two summers I spent with them very pleasant and productive.

My sincere appreciations are due to the assistance provided in conducting this research by many process planners and product designers from Caterpillar Inc. at its East Peoria, Davenport, and Mossville plants. Among them Cliff Cirillo and Karl Nurstad deserve special mention. I am grateful to them for taking time off from their busy schedule to enable me to complete the knowledge acquisition process required for this research. Many of the concepts in this thesis are a direct result of the insight they provided into the product and process

design tasks. I am indebted to them for the generous assistance they provided.

I would like to thank members of the Knowledge-Based Engineering Systems Research Laboratory (KBESRL) for their help, ideas, discussions, and comaraderie, all of which made KBESRL a great place to work and conduct research. Allen Herman and Bob Wilhelm helped in using the constraint-based system developed by them, and Jim Thompson provided selfless computer support.

I am grateful to John Sutherland, Mike Kuhl, Mani Subramani, and Guangming Zhang for helping me understand the mechanistic and empirical process performance models and patiently answering all the questions I had about the models.

I would like to thank June Kempka, Suzanne Faikus, Tammy Lawhead, and Celia Daniel from the Mechanical and Industrial Engineering Publications Office for their help in putting this thesis together.

Though miles away, the good wishes and unassuming trust of my parents as well as my parents-in-law was a constant source of guidance and solace.

Finally, thanks to my loving wife Jyothi who provided great comfort at times of doubt. She has had to persevere through my idiosyncracies while having to work on her own doctoral program. She did it with commendable grace and style. It is to her that I dedicate this work.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 Background

One of the challenges in engineering is the development of useful, reliable, and economical products. In the past, engineers utilized their broad expertise gained through years of experience in carrying out product development. They incorporated large safety factors into those areas of product development where they lacked detailed expertise. This centralized product development practice was technically sound and economically feasible. With increases in the complexity of products and manufacturing processes brought about by technological innovation, this product development process soon evolved into being distributed among a team of engineers, each with different responsibilities and narrow fields of expertise. Within this distributed product development practice, product development advanced serially, with many iterations occurring within and among engineers from marketing, product design, product evaluation, process design, production, quality control, product service, and maintenance.

Today, within such a distributed product development environment, fast changing and highly competitive economies are forcing industries world-wide to seriously consider various ways to reduce product development time and cost. Manufacturing costs form a major com-

ponent of the total cost of a product. A number of different measures can be taken to reduce this cost. For example, cost reductions can be obtained by standardization, automation, new materials, cutting tools and manufacturing processes, improved material handling and assembly techniques, and more effective factory layout schemes. Some of the reductions in manufacturing cost are highly dependent on the design of a part and can be fully achieved only when it is taken into consideration while the part is being designed. However, in the distributed product development environment, the primary goal of designers often is the development of part designs to meet structural and functional requirements. Manufacturing requirements are either not considered by designers or are taken into consideration on an ad-hoc basis. Manufacturing costs have, therefore, continued to form a large component of the total cost of a product, in spite of cost-saving improvements in the manufacturing arena, e.g., the development of Computer-Controlled Machines, Flexible Manufacturing Systems, and Robotic Cell Assembly Systems.

Recognizing this apparent anomaly in the product development process, considerable attention has been directed recently to integrating the tasks of the engineering design and manufacturing planning departments. The concept of Simultaneous Engineering has been proposed as an improved product development practice that can concurrently integrate a wide spectrum of product life-cycle concerns to reduce product development time and cost and achieve higher quality. This concept calls for parallel interaction and true cooperation among various product development engineers. Current product development practice is a sequential, iterative, and distributed approach. In contrast, the Simultaneous Engineering concept requires a parallel, interactive, and cooperative team approach. This concept has become an important subject of study in both academic and industrial circles, and almost every major industrial organization has special task forces to deal with this critical issue [1-6].

1.2 Research Motivation

Even though tremendous gains can be achieved by implementing the Simultaneous Engineering concept, pursuits in this direction have encountered numerous difficulties. These difficulties need to be overcome before full benefits of the Simultaneous Engineering concept can be realized. Firstly, the degree of concurrency that can be achieved between design and manufacturing tasks has been a highly debatable research issue. Secondly, the task of developing design guidelines that suitably incorporate manufacturing considerations has proved quite problematic. Identifying the manufacturing concerns of a particular domain has been very time consuming and requires several man-hours of work from a group of people. The diversity of the manufacturing arena and the lack of a systematic procedure or proper basis for establishing these manufacturing concerns has served only to aggravate this task. Thirdly, suitable means have not been found to record the new design guidelines. The frequent revisions that these guidelines undergo because of changes at the manufacturing end have made it less than satisfactory to record them in engineering standards manuals or in design and manufacturing reference manuals. Even without incorporating these guidelines, the manuals are large and cumbersome to use and have been constantly ignored by designers as a useful design aid. Lastly, it has been difficult to develop effective and logical means to ensure that these guidelines are properly followed and implemented in practice. In the absence of an alternate scheme, designers are often entrusted with the additional responsibility of ensuring that the guidelines are properly used. This practice has proved to be a tremendous burden on designers, forcing them to work very much below their capabilities.

This research is mainly motivated by the realization that there is a definite need to study the Simultaneous Engineering concept in greater detail and to propose viable solutions for implementing it. For example, a portion of the scenario during the implementation of the

Simultaneous Engineering concept for the design of the Gear Group Transfer Case shown in Fig. 1.1 might be as follows:

> ... Due to alignment considerations, the designer specifies a tolerance of 0.0125 mm on the straight inner diameter (that seats the bearing) of Bearing Cage-A. A manufacturing engineer on studying the part from a manufacturing standpoint indicates to the designer that the designer has just increased the cost of manufacturing the part by adding an extra grinding operation. The designer then reconsiders his functional requirements for the design of the part and decides to loosen the tolerance, yet maintain the alignment requirements by using shims.
>
> ... Assuming that a similar type of product development activity takes place while designing Bearing Cage-B, this time an assembly engineer, on studying the part from an assembly standpoint, indicates to the designer that assembling such a part on an assembly line using shims would be very difficult (increases assembly cost) because of the size of the part. In this case, since the Bearing Cage is used to align two bearings, alignment considerations are very critical, and the designer is left with no other alternative but to specify a tight tolerance on the inner diameter that seats the bearings ...

Although the above scenario is hypothetical, it shows the main emphasis of the Simultaneous Engineering concept (interaction between product development participants as early as possible) and the need for a systematic approach to achieve it. The main motivation for this research is derived from the dire need to enable such scenarios to become a reality in the product development arena.

1.3 Research Scope and Objectives

1.3.1 Scope

The y-axis of the graph shown in Fig. 1.2 lists some of the concerns (in addition to functional and structural concerns) that can be addressed within the scope of the Simultaneous Engineering concept. The x-axis of the graph shows the different kinds of commodities for which some or all of these concerns can be relevant. Due to the wide scope of the Simultaneous Engineering concept, it is extremely difficult to concurrently consider and

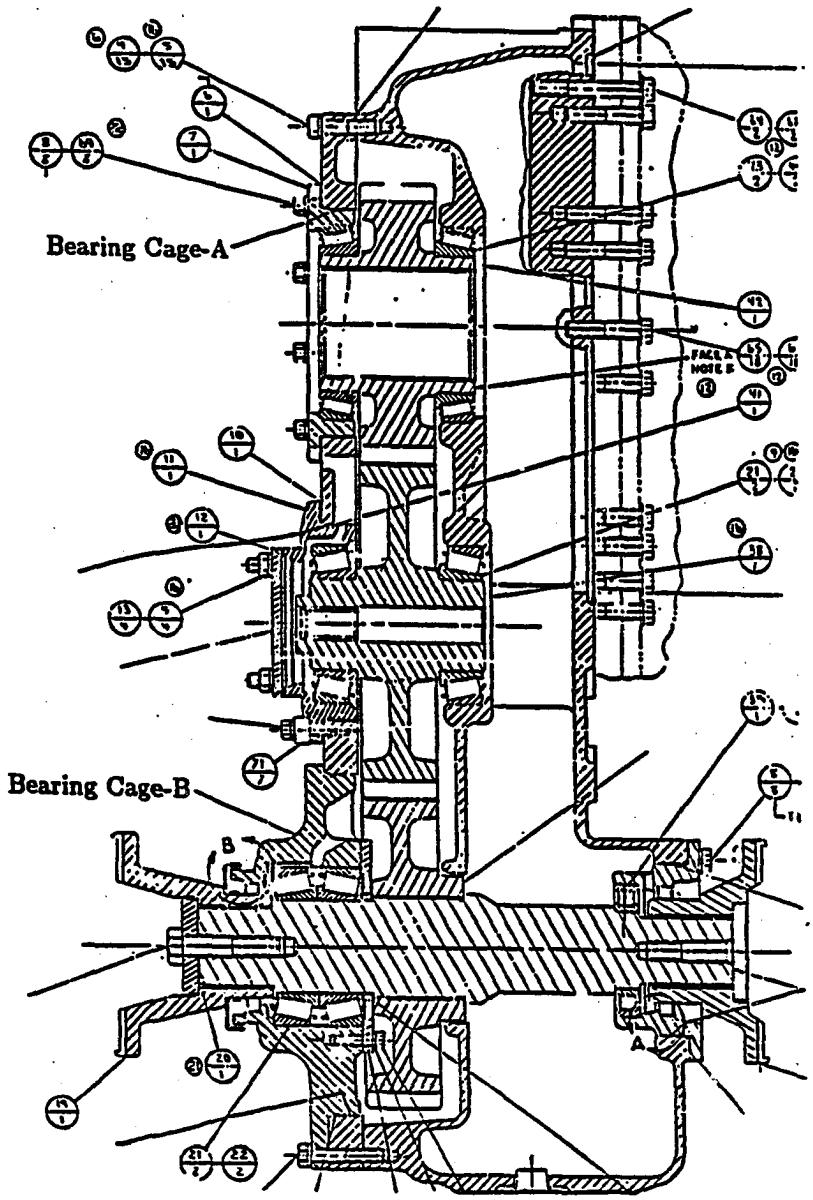

Figure 1.1: Gear Group Transfer Case Design

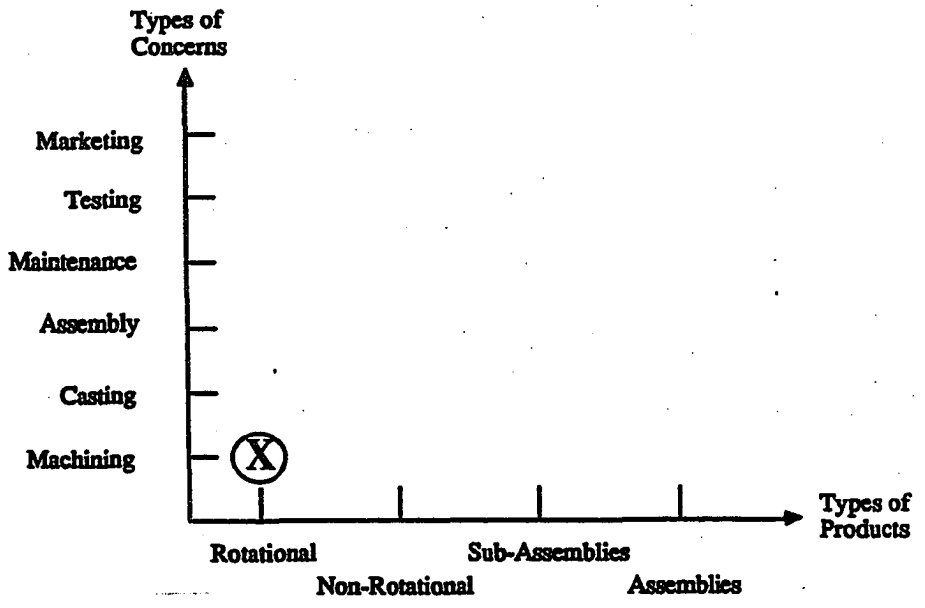

Figure 1.2: Research Scope

propose solutions for all the problems encountered during its implementation. The radical and unique nature of this concept demands that only a subset of the research issues posed by this concept be addressed at any given time. In order to study, understand, and tackle some of the fundamental issues raised, it is important to restrict the range of "manufacturing concerns" and "commodities" considered. In this research, manufacturing concerns from the machining area are considered and these issues are dealt with in the context of developing the design of rotational components. When restricted to functional, structural, and machining-related concerns, Simultaneous Engineering entails concurrent product and process design. Henceforth, the term "Simultaneous Engineering" is used in this restricted sense.

1.3.2 Objectives

Simultaneous Engineering is a fairly recent research issue and people are still experimenting with various ways to implement this concept. To get a better understanding of this concept for different domains, research in this area tends to be problem-driven. This research has taken a similar approach and the main objective is to conduct a detailed study and propose a methodology[1.1] to implement the Simultaneous Engineering concept for rotational components manufactured in small and medium lot-sizes. The following major research issues are addressed as part of this work:

- What is the degree of concurrency that can be realistically achieved between product and process design?
- What are machining-related concerns, and how can they be systematically identified and classified?
- What are the changes introduced by the Simultaneous Engineering concept in the traditional approach to product and process design?

1.4 Fundamental Contributions

The fundamental contribution of this research is the development of an Artificial Intelligence (AI) based design environment to implement the Simultaneous Engineering concept for components manufactured in small and medium lot-sizes. In developing this approach, the importance of the Model-Based Reasoning concept for supporting the concurrent product and process design activity is demonstrated. Multiple product and manufacturing facility models and the Multiple Cooperative Knowledge Sources paradigm are established as key

[1.1] A term used throughout this thesis to describe a procedure.

components of the design environment. This research also exemplifies the efficacy of using AI-based systems to complement (rather than replace) the activities of human experts in engineering applications. It is shown that the AI-based design environment leads to an efficient restructuring of the process planning task and to significant improvements in the productivity of the process planner.

1.5 Organization of this Thesis

Chapter 2 presents a literature survey of relevant and current research in Simultaneous Engineering, Computer-Aided Design, Computer-Aided Process Planning, and Part Representation schemes. A review of AI tools and techniques used in this research is also presented in this chapter. Chapter 3 describes the methodology proposed to address the Simultaneous Engineering concept for components manufactured in small and medium lot-sizes. Chapters 4 and 5 describe in greater detail relevant aspects of a computer-based design environment developed to support the approach proposed for Simultaneous Engineering. The design environment has been tested and validated by implementing it for the simultaneous product and process design of Bearing Cages. Chapter 6 presents the salient points of this implementation. Important observations about the design environment, conclusions and recommendations are presented in Chapter 7.

Chapter 2

LITERATURE SURVEY

2.1 Introduction

A review of work in the area of Simultaneous Engineering is presented in Section 2.2. Literature in the areas of Computer-Aided Design (CAD) and Computer-Aided Process Planning (CAPP) is reviewed in Sections 2.3 and 2.4, respectively. The main intention in reviewing literature in these two areas is to study the extent to which work has addressed any of the relevant aspects of the Simultaneous Engineering concept. Work in the area of Mechanical Part Representation is summarized in Section 2.5. A brief review of AI concepts, tools, and techniques used in this research is presented in Section 2.6.

2.2 Simultaneous Engineering

In Chapter 1 it was indicated that Simultaneous Engineering encompasses a wide variety of concerns and has a very broad scope. This characteristic is also reflected in the varied approaches undertaken to address different aspects of this concept.

2.2.1 Design for Assembly

Design for Assembly (DFA) is a major sub-area in which much work has been done recently [7-11]. Guidelines have been developed to enable designers to develop better sub-assembly and assembly designs so that "good assembly practices are designed into a product rather than planned into a production line." Various schemes enabling designers to reduce the number of parts in an assembly and making it easier to assemble the remaining parts have been proposed [7]. Modularization, function integration, function distribution, parametric design, and dimensional standardization are representative examples of guidelines that fall within the scope of good DFA principles. Several case studies have been reported wherein assemblies have been extensively redesigned to the point of making assembly costs an insignificant portion of the total cost of an assembly [11]. In certain cases, product redesign made it less profitable to automate the assembly process.

Only recently has work begun in developing computer-based environments that would enable designers to follow the DFA guidelines. The work reported by Lai [12] and Ulrich [13] are steps in this direction. A Function Description Language (FDL) is presented in [12] to enable designers to describe the functions of components and the relations between components in an assembly. The functional description of an assembly is parsed using DFA principles thereby identifying and eliminating any redundant functions and structures in the assembly. The end result of this process is the generation of a more efficient design of the assembly.

Ulrich's [13] work dealt with the automated construction of schematic and physical descriptions of single-input, single-output (SISO) dynamic systems. Given a desired input/output response of such a system, an algorithm first constructs a schematic diagram using bond graphs. A physical description is then created with one-to-one correspondence

between schematic and physical elements. Function sharing[2.1] is then used to simplify the physical description. This involves making structural changes that allow adjacent physical elements to be replaced by a single element that provides the function of both elements.

Ishii et al. [14] describe the use of Design Compatibility Analysis (DCA) [15] as a means of developing computer-based tools to support Simultaneous Engineering. DCA focuses on quantifying the degree of compatibility between the design requirements (specifications) and a proposed design. It is a general means of suggesting improvements in the proposed design to increase the degree of compatibility. Ishii et al. [14] suggest that this concept be extended to accommodate other life-cycle concerns of a product such as assemblability. The use of knowledge-based systems is proposed to capture compatibility knowledge associated with different product life-cycle concerns. This methodology has been tested in two domains: system design of power generation plants and design of mechanical products for assemblability.

Rehg et al. [16] describe the development of a Computer-Aided Design (CAD) system that can support mechanical design at the assembly level and also serve as a means of integrating design and assembly concerns. A prototype system has been developed for automatically synthesizing automotive window regulators. The key characteristics of this system that distinguish it from traditional CAD systems are the use of: (1) multiple design agents or critics that embody different kinds of expertise required in the design of a particular assembly, and (2) the use of multiple design representations tailored to the needs of the design agents.

Finger et al. [17] and Lu et al. [18,19] provide general descriptions of computer-based systems that can assist in creating mechanical designs that simultaneously meet product life-cycle requirements. They propose the use of: (1) blackboard systems to support and

[2.1] A DFA principle.

integrate multiple perspectives, (2) feature-based design entities to represent design abstractions at different granularities, and (3) constraints to guide the design process by maintaining consistency through constraint propagation. An implementation of such computer-based systems in specific domains has not been described, but the framework[2.2] presented is sufficiently general to provide a domain-independent structure for performing cooperative product and process design. Several aspects of the generic design environment are noteworthy and this research builds upon the general guidelines provided.

One of the drawbacks of the DFA concept is that the established guidelines concentrate mainly on assembly-level concerns and do not consider the manufacturability-related concerns of the individual components within an assembly. Dewhurst [20] proposed an extension to the DFA concept where the task is viewed as occurring at two levels of abstraction. An assembly is designed using DFA principles to develop a design that has the fewest parts and the maximum ease of assemblability. Components of the final design are then individually designed to satisfy the manufacturability-related concerns of each component.

2.2.2 Design for Manufacturability

Design for Manufacturability (DFM) addresses the manufacturing-related concerns of individual piece parts. Dewhurst [20] describes the development of an approximate method for estimating the cost of a machined or formed component early in the design phase. The assumption that the part is processed under ideal processing conditions is made in arriving at a method to estimate the cost. The method described in this work is a compromise between an oversimplified approach and the more traditional detailed cost estimating methods. Since it does not take into consideration all the factors that contribute to the total cost of a part, its validity holds only when one can ensure that the excluded factors contribute

[2.2] A term synonymous with "architecture" and "structure" in this thesis.

negligibly to the final cost of the component. No guidelines have been provided to establish the validity of these assumptions in a particular domain. The burden of ensuring that the assumptions are true is left to the user.

Boothroyd [21] describes the importance of the need to consider machining concerns while developing the design of a part and illustrates several guidelines developed from a machining standpoint. No computer-based framework enabling designers to ensure that these guidelines are met is presented.

Dixon and others [22-26] proposed a different approach to address the Design for Manufacturability concept. The problem considered deals with the issue of developing the design of components manufactured by metal forming processes (like casting, forging, injection molding) to satisfy all manufacturability-related concerns. Since manufacturability-related knowledge in these areas is very heuristic and empirical in nature, a knowledge-based systems approach has been proposed as a solution. In these approaches, the design of a component is described using a feature-based part representation scheme. This description of the part is used as a basis to check for manufacturability-related concerns in the design of a part. Only the manufacturability-related concerns pertaining to individual features of a part are considered. Concerns that involve interactions between two or more features have not been addressed. However, the methodology proposed is promising and some of the work reported in this research expands upon this approach to the DFM concept.

Cutkosky and Tenenbaum [27] present a slightly different methodology for achieving the DFM concept that involves the designer working in "manufacturing" modes. The designer specifies the design he is creating by outlining a sequence of processing steps. For example, a machined part would be defined as a blank which is shaped with operations such as holes, pockets or sweeps. Knowledge-based systems and solid-modeling systems are then used to generate processing requirements and check for violation of manufacturing constraints. The

system has been partly implemented and tested on simple component designs.

2.2.3 Taguichi Approach

A different approach to the Simultaneous Engineering concept at the component level is described by DeVor et al. [28,29]. A framework is presented based on Taguichi's model for the design process. The design process is described as taking place in three stages:- System Design, Parameter Design, and Tolerance Design. In the System Design stage, current experience and technological capabilities are applied to arrive at the most promising design alternative. In the Parameter Design stage, a parametric study and analysis of important factors of the design alternative is conducted to determine their optimal values. At the Tolerance Design stage, the loss function concept from Quality Control literature is used to select allowable tolerances for the important design parameters.

The work by DeVor et al. [28] concentrates on the Parameter Design stage and implements the Simultaneous Engineering concept by identifying design as well as manufacturing-related parameters that should be considered at this stage. Design of Experiments techniques are used to specify the nominal values for the important design and manufacturing-related parameters. The main selection criterion used is to "identify those nominal values which minimize the transmission of functional variation to the output performance as a result of the presence of noise factors operating in the environment in which the product and/or process is functioning" [29]. The methodology has been demonstrated using the face milling process as an example. Part stiffness was identified as the important product parameter. Feedrate, depth of cut, number of inserts, and cutter offset were identified as important process parameters. Mechanistic models [29] were used to perform computer-based simulations of the manufacturing process to determine the process parameter values. Mechanistic models also form an important part of the proposed approach to implement

the Simultaneous Engineering concept. An alternate approach to using these models to determine the effect of design specifications on process performance parameters is described in this thesis.

2.3 Computer-Aided Design (CAD)

The design process, viewed as a dialectic between the designer and what is possible [30], has been a subject of intensive study recently. This section presents a review of the research in developing computer-based environments to assist or automate the product design process. Since this research deals mainly with mechanical components, the review of work in Computer-Aided Design is restricted to the area of mechanical design.

The main intention of this review is to broadly characterize the research in CAD to determine if any relevant aspects of the Simultaneous Engineering concept have been addressed. The next three sections review three different approaches proposed in the literature for modeling the design task: design-evaluation-redesign, refinement and constraint propagation, and multiple agents design. The final section summarizes the main conclusion of this literature review.

2.3.1 Design-Evaluation-Redesign

Howe, Cohen, and others [31-34] present a model of design where the task is viewed as consisting of several cycles of evaluation and redesign. For the system developed by Howe [31], the input to the system is a set of problem parameters describing physical constraints of the design, a set of performance goals, a set of design variables, and an initial design. The initial design is first evaluated to determine if it meets all the goals of the design. If any of the goals are not met, then a particular design variable is chosen (based on knowledge about the design domain) and a change in the value of the variable is proposed. If the overall effect of

the change in the design variable is assessed to be positive, then the change is implemented. The new design is evaluated and the cycle continues until all performance goals have been satisfied. This process can be characterized as a general hill-climbing procedure and works when there is minimum interaction between design variables.

Dixon et al. [33] applied this model of the design process to the design of standard V-belt drives. Principles from utility theory are used to evaluate a design and classify it as acceptable or unacceptable. Kulkarni et al. [34] used this model of the design process for the design of heat fins. The redesign phase of the design process is the most critical task; in the system developed, various sub-modules for performing the redesign phase have been provided. Based on the results of the evaluation phase, one of the modules of the redesign phase will be chosen to improve the design. An extra stage has also been incorporated in the design process to check and compensate for overdesign.

2.3.2 Refinement and Constraint Propagation Design

Tong [30] presents another model of design where design is viewed as consisting of two interleaved activities: *refinement* and *constraint propagation*. Product design is assumed to progress essentially top down, with the design of a complex system being broken down into the design of smaller sub-systems. Since the task of designing the smaller sub-systems cannot be performed independently, it is assumed that the interface between the sub-tasks will be defined. It is often infeasible to completely and precisely define the interface between the tasks before the sub-tasks have been partially performed. Therefore, it is assumed that these interfaces are defined while the individual sub-tasks are performed.

Refinement is the term used to describe the top-down design activity. Refinement for complex design tasks is a two-step recursive process. The first step is a planning step where initial specifications serve as top level goals for plan generation. The second step is execution

of the plan. It results in construction of product design descriptions. Refinement is applied recursively if plan steps serve as sub-goals for more planning. *Constraints* is the mechanism used to record sub-task interconnections, and *Constraint Propagation* is the term used for the activity which applies these constraints and ensures consistency between the refinement activities for individual sub-tasks. The refinement and constraint propagation model of design is a general and powerful model of the design activity and is applicable to both electrical and mechanical design.

McDonald [35] and Langrana [36] used this model to address a task that first requires the determination of the kind of system (e.g. belt system or gears) that will be used, and then requires refinement of the chosen design to meet a user's requirements. McDonald [35] views such a design task as the instantiation of archetypes (or classes) and refinement of the instance(s) created using hierarchical decomposition and recomposition techniques. The various modules (or components) of the final design form the basis for the hierarchical decomposition task. Recomposition is achieved through the establishment of constraints between the instances created for the various modules. A redesign process is initiated if failure occurs either during instantiation of modules, at the decomposition stage, or at the recomposition stage. Design of a drive system has been chosen to demonstrate the efficacy of the proposed framework.

Langrana [36] approaches the design process by making assumptions about the categories of knowledge used in design, the representation used for the designed artifact, and the process by which knowledge is used to achieve a successful design. Three categories of knowledge are identified: implementation, control, and causal knowledge. Frame-based representation is used to model the designed artifact. Design proceeds with a least-commitment, top-down strategy in which constraints arising from implementing one part of the design are communicated to other parts of the design. The system has been demonstrated by implementing

it for the design of motion transmission assemblies.

Brown [37,38] used the refinement and constraint model of design for the Routine Design task. Routine Design is classified as a task that a designer has performed many times with different requirements such that knowledge required to perform the task is available in a highly compiled form. Brown [37] studied the routine design of Air Cylinder systems and proposed a methodology to automate this task. The design task is viewed as "A hierarchy of conceptual specialists solving the problem in a distributed, top-down manner by choosing at each stage of the design from a set of plans, thus refining the design" [37]. A special purpose language for expressing design knowledge called Design Specialists and Plans Language (DSPL) has been developed. DSPL defines: 'Specialists' as being responsible for solving design sub-problems; 'Tasks' as being responsible for a particular task within a sub-problem; and 'Steps' as being responsible for a single decision-making step. DSPL also expresses design plans, redesign knowledge, and constraints between parts of the design being developed.

Mittal and Araya's [39,40] work in modeling the design task is similar in many respects to Brown's work [37] in the area of routine design. Design is viewed by them as consisting of specification, generation, and evaluation stages. Their work focuses on the organization of design knowledge into "design plans" for use in the design generation stage. Plans specified for "goals" are responsible for a subset of the design parameters that define the dimensions of the design space. Goals have alternate methods to make decisions about parameters: sub-plans, generators, calculations, procedures, constraint generators, rule groups, and conjunctive methods. Goals also have constraints attached to their parameters and constraints have "advice" which is activated when constraints on parameters are violated. Design is viewed as knowledge-directed search, the search direction being established by the activation and execution of plans.

2.3.3 Multiple Agents Design

Mayer et al. [41] and Rychener et al. [42] modeled the design process as the integration of multiple, possibly overlapping, design knowledge sources. Mayer et al. [41] model knowledge sources as forward-chaining rules selected by the user, who is also another knowledge source. The most interesting aspect of the system is that potential conflicts between knowledge sources are settled by the sources themselves during run time rather than by the knowledge engineer who creates the system. The design of a speed reducer has been chosen as a suitable demonstration domain.

The ALADIN system developed by Rychener et al. [42] has been used to design alloys at the micro-structural, bulk material property, and production processing levels. Alloy design problem is treated as a planning problem because the final alloy design is a sequence of steps to be taken to produce the alloy. The system uses multiple levels of abstraction to create designs. Meta planning and least commitment strategies using constraints and hypotheses expressed as ranges on values are employed to control the reasoning process.

2.3.4 Link to Simultaneous Engineering

Computer-based systems that have been developed to assist or automate mechanical design are based on knowledge obtained from "design experts." The majority of these systems, therefore, generate designs that satisfy functional and structural considerations. Manufacturability-related concerns have either not been considered or have been ignored because the knowledge to incorporate such concerns in these systems is not readily available. However, it is imperative that future extensions to the design environment frameworks described in this section consider manufacturability requirements. The role of manufacturability evaluation within the refinement and constraint propagation model of design was studied in detail in this research.

2.4 Computer-Aided Process Planning (CAPP)

The focus of this research is on addressing machining-related concerns during product design. Traditionally, a majority of these concerns arise when process plans are developed for parts. Currently, the process planning activity is carried out with a number of computer-based decision aids that assist or automate the process planning task. This section presents a classification of such CAPP systems with the intention of studying if these systems address issues pertinent to the Simultaneous Engineering concept. The first three sections describe three different kinds of CAPP systems: variant, semi-generative, and generative. The last section summarizes the main findings of the literature review.

2.4.1 Variant Systems

Initial work in applying computers to aid the process planning task has been in the area of Variant Process Planning Systems. In this type of CAPP system, parts are grouped into part families, a unique code is generated for each part family, and a standard process plan is developed for each family. Most systems use a well-developed Group Technology (GT) based coding system to develop the unique codes for the various part families. The standard plans are stored in a computer and conveniently keyed under the unique code generated for each family. This type of process planning system is used by first determining the part family of a new part and then retrieving and filling up the standard process plan to reflect the characteristics of this part. CAPP [43], MIPLAN [44] and ACUDATA/UNIVATION [45] are well known examples of such CAPP systems.

2.4.2 Semi-Generative Systems

The semi-generative systems are advanced variant systems and incorporate quasi-generative features. After the part family has been identified (as in a basic variant system), these

systems offer the user several options. One option is to make suitable changes to the standard process plan for each part family. The second option is to begin with an incomplete process plan and complete it for a specific part. A third option is to start from the beginning and create a complete new plan by using various standard process descriptions stored in the computer. Preliminary versions of GENPLAN [46], XPS-1 [47] and CORE-CAPP [48] are examples of such systems.

2.4.3 Generative Systems

Generative process planning systems are designed to automatically synthesize information to develop the process plan for a part. They also combine a manufacturing logic module with a suitable part description scheme to "generate" the process plan for a particular part. Early versions of generative process planning systems used "decision tables" and "decision trees" to capture manufacturing logic and "GT code" or "special purpose languages" to describe the part. Such systems are more complex than their variant counterpart and also more restrictive in the breadth of their application. DCLASS [49], APPAS [50] and CPPP [51] are some of the well known systems of this kind. The advent of CAD databases and wire-frame modeling led to the development of generative process planning systems based on this type of a part description scheme. TIPPS [52] and RPO [53] were some of the early generative systems to explore the link between CAD databases and generative systems.

The next major development in the area of generative process planning systems was the use of AI techniques to model the activities pertaining to the manufacturing logic component of these systems. Concepts from the Knowledge-Based Systems and Planning areas in AI were found to be particularly suitable for applying the captured manufacturing logic to a suitable description of a part, thereby generating the process plan. GARI [54], TOM [55], XCUT [56] and SIPP [57] are some examples of systems that achieved a certain

degree of success in using AI techniques to develop generative process planning systems. Today generative process planning systems are predominantly based on AI techniques and are being applied to different aspects of the process planning problem. Ham and Lu [58] provide a more detailed discussion of current and future research directions in CAPP.

2.4.4 Link To Simultaneous Engineering

Although a great deal of work has been done in developing various CAPP systems, none of the systems address the issue of machining-related concerns raised by the Simultaneous Engineering concept. This is because these systems have been based on the traditional, sequential product development process where process planning begins after completion of the product design. For example, the successful development of a process plan for a part using a CAPP system does not necessarily imply that all machining-related concerns have been resolved. There are situations where machining-related concerns are still present, in spite of the existence of a process plan. In such situations, it is sometimes possible to further reduce manufacturing costs by making suitable design changes that eliminate these concerns without affecting structural and functional considerations. Another example is the situation where there are several different process plans for a particular design of a part, but some or all of the plans become invalid when the part is restricted to be manufactured in a particular manufacturing facility. Here again the mere existence of a process plan does not imply that there are no machining-related concerns.

In spite of these drawbacks, these systems have been instrumental in laying a strong foundation for a Computer Integrated Manufacturing System (CIMS). It would, therefore, be very prudent to extend these systems to tackle some of the issues raised by the Simultaneous Engineering concept. Although variant types of CAPP systems are more prevalent today, the generative CAPP systems are more appropriate for implementing several aspects

of the Simultaneous Engineering concept. This research presents several ways of extending generative process planning systems to address issues related to the Simultaneous Engineering concept.

2.5 Part Representation

Computer-based representation of mechanical parts is a major focus of the research described in this thesis. Various part description schemes have been reported in the literature. A brief review of the more important schemes is presented in the following sections.

2.5.1 Solid Modeling

Solid modeling,[2.3] which grew out of early applications of computers to design and manufacturing-related tasks, provides several significant and distinct methods for representing parts. Solid modeling is concerned with the representation and manipulation of subsets of three-dimensional Euclidean space. It also enables the construction of computer systems to support these tasks. Solid modeling systems have three components: a representation, a means of performing operations on that representation, and a means of querying that representation for information. A brief description of some of the more important solid modeling schemes follows. A more detailed discussion of these representational schemes is provided by Requicha and others [59-61].

The Boundary Representation (B-Rep) scheme models objects by representing their boundaries. A three-dimensional boundary model has data elements that correspond to faces, edges, and vertices. Boundary models are also called "evaluated models" because they store information in a form which is easy to compute. However, the model descriptions are quite verbose and extensive error checking routines are required to ensure their syntactic

[2.3] or Geometric Modeling.

validity.

Constructive Solid Geometry (CSG) modelers are based on the premise that complex shapes are actually combinations of primitive graphic shapes, e.g., cone, torus, sphere, cube, wedge, and cylinder, which can be added or subtracted by means of regularized boolean set operators. These models are complete and valid representations of solids, they have syntactically guaranteed well-formed conditions and are compact over a wide range of solids. Extracting information from these models, however, requires a complex form of evaluation called boundary evaluation which converts the CSG structure to a boundary model.

The Sweeping Modeling scheme is based on the fact that some solid shapes can be created by moving a line or plane on a defined trajectory. Designers often prefer this modeling scheme because these methods are easy to use. However, an important limitation of this scheme is that it can be used only for shapes which have rotational or translational symmetry.

Spatial Enumeration, Cell Decomposition or Tree Decomposition are all octree representation schemes (an octree is the three-dimensional generalization of a binary tree). They represent geometry in an approximate form by modeling an object as a union of non-overlapping predetermined cells. They are very useful for modeling highly irregular solids, although their memory space requirement is rather high. These schemes have been used primarily in the areas of Computer Vision and Medical Imaging.

A solid model representation of the part is an important component of the part representation scheme developed in this research. CSG and B-REP are the schemes primarily used for the design of mechanical parts. These representation schemes were also extensively used in this research.

2.5.2 Feature-Based Modeling

The application of Knowledge-Based Systems concept to design and manufacturing tasks has led to the recent development of an entirely different kind of part representation scheme [62-66]. This is the Feature-Based Part Representation Scheme. In this scheme, parts are represented at a higher level of abstraction than the solid modeling schemes. A feature is any geometric form or entity whose presence or dimensions is relevant to one or more design or manufacturing tasks or whose availability to designers as a primitive facilitates the design process [62]. In such a part representation scheme, typical part features such as holes, bosses, cutouts, flats, slots, bores, threads, chamfers, fillets, grooves, and pockets are explicitly defined unlike their implicit representation in most solid modeling schemes. Feature-based part representation schemes have been proposed fairly recently and work is still under progress to establish their efficacy in design and manufacturing domains. This research made extensive use of the feature-based part representation scheme and made several important contributions to developing the fundamental characteristics of this representation scheme.

Although Solid Modeling and Feature-Based Representation Schemes are very different part modeling schemes, there is a strong need to integrate these two types of part representation schemes. Woodbury [67] conducted some preliminary work in this area and describes an architecture for performing such an integration. Classes of spatial sets, features, abstractions, and constraints are four concepts that are used to define the complete architecture. A prototype implementation of the architecture using object-oriented programming techniques [68] and a boundary based solid modeller [69] is described. Part representation is an important aspect of the computer-based design environment proposed in the research described in this thesis. The need for integrating feature-based part representation schemes

with geometric modeling schemes, and the relevance of the work reported by Woodbury [67], is established and ascertained in this research.

2.6 Artificial Intelligence Tools and Techniques

Artificial Intelligence (AI) played an important role in this research in developing the methodology for addressing the Simultaneous Engineering concept. Section 2.6 provides a brief review of the AI-based concepts and techniques used.

2.6.1 Model-Based Reasoning Systems

Model-Based Reasoning [70-72] is the knowledge-based systems approach to problem solving that involves building, analyzing, and reasoning from an explicit computational model of the structure, principle, function, and behaviour of an underlying system. Separation of the structural/functional model of a problem domain from the problem-solving knowledge is the basis of any model-based reasoning system.

Model-based reasoning is suitable for complex domains where more than one problem has to be solved. Instead of developing task-specific tools in isolation from one another, a central model in such systems provides a unifying foundation for the integration of such tools. The model is usually specified in terms of structured objects (see section on Frame-Based Systems below), object function and behaviour, and relations between the objects. Models in essence are a simplified and selectively abstracted representation of a natural system that is too complex to be represented in its entirety. Tasks such as design, planning, scheduling and diagnosis that are typically attempted with the aid of knowledge-based systems can greatly benefit from such an approach. For example, in [70] a scheduling and diagnostic knowledge-based system is presented for a new manufacturing facility. This system has been built using the model-based reasoning approach. A detailed model of the factory is

the underlying foundation for the scheduling and the diagnostic system. The model consists of objects representing the various components of the factory (e.g. machines, products, and conveyors), behaviours for the objects that are implemented as rules or procedures (e.g. what type of operations can be performed on a machine), and relations among objects (e.g. upstream object and downstream object).

2.6.2 Blackboard Systems

A Blackboard System [73-76] is a method for constructing AI-based environments to solve complex problems where different kinds of knowledge and expertise is needed. The process by which blackboard systems construct solutions is incremental and is based on the progressive application of a variety of knowledge to solution elements at varying levels of abstraction. A Blackboard System is a fairly informal construct and has evolved over the years since HEARSAY-II [74], the first blackboard system, was constructed. A Blackboard System now is universally accepted to be composed of the following three main components:

1. A globally accessible database called the Blackboard. The Blackboard is structured as a linear hierarchy of abstraction levels and contains the results of applying problem-solving knowledge. Items placed on the blackboard at various abstraction levels are called "entries", and are complex structured objects. Each "entry" is composed of a set of attributes which are assigned values after the "entry" is placed on the blackboard.

2. A set of Knowledge Sources. Knowledge Sources represent the problem-solving knowledge contained in the system. They respond to changes in the state of the blackboard by altering its contents.

3. A control component called the Scheduler. The scheduler controls the problem-solving behaviour of the system by monitoring the state of the blackboard and selecting one

or more Knowledge Sources to apply.

A detailed description of Blackboard Systems is provided in [76]. The blackboard system has been used extensively in this research and details are provided in Chapter 5.

2.6.3 Frame-Based Systems

Frame-Based Systems [68,77,78] are designed to provide an easy means of describing structured objects or classes of objects in a particular domain. A frame-based language provides a stereotyped description of individual object classes which are then used to create default descriptions of a particular object. The structured objects can be described at different levels of abstraction and can be combined at run time via an inheritance mechanism. Relationships between classes are expressed as an inheritance hierarchy organized into a tree structure with classes at the lower levels being more specific sub-classes of the classes above it. Slots are used to describe the attributes of a particular class. For example, frame objects can be used to represent vehicles, refinements of the general class of vehicles such as automobiles and trucks, and still further refinements of automobiles such as sedans, coupes and station wagons. Attributes of vehicles such as color, height, length, location, and owner can be represented as slots of the vehicle frame [78].

Although frame-based systems provide no specific facilities for declaratively describing operations on frames, they do provide various ways of attaching procedural information to frames. This procedural attachment capability enables the building of behavioral models of objects and expertise in an application domain. It also provides a powerful form of object-oriented programming in which objects represented as frames can respond to messages sent to them [68]. Active Values and Methods are two ways of attaching procedural information to frames. Active Values are procedures attached to frame slots (attributes) that are invoked when the slot's values are accessed or stored. They are commonly called "demons" because

they constantly monitor changes and uses of a slot's values. Methods are procedures, attached to frames, that respond to messages sent to the frames. Methods are stored as the values of slots that have been identified as *message responders*. Messages sent to frames specify the target method and include any arguments needed by the method. For example, "update.location" can be an active value attached to the "location" slot of the automobile frame [78]. Whenever the value of the "location" slot is changed, this active value will be invoked to update the location of the automobile in a geographical map being displayed to the user. "Diagnose" can be a method attached to the automobile frame that when activated would diagnose electrical faults and put down the result as the value of the slot "electrical.faults" of the automobile frame [78].

Methods are usually defined and given values at the highest possible level in the inheritance hierarchy. Their values can be changed at the lower levels of the hierarchy, either by adding a new local method or by incrementally specializing the existing methods [68]. Incremental specialization can be done by adding *Before* and *After* methods (at subclasses) that are to be performed before and after (respectively) the main method is executed. Incremental specialization can also be achieved by adding *whoppers* that allow a procedural combination of new and inherited methods at the lower levels of an inheritance hierarchy. A more detailed description of Frame-Based Systems is provided in [68,77].

2.6.4 Constraint-Based Systems

Constraint-Based Systems [79,80] are used to represent relationships between slots of frame instances. For example, if the value of "slot3" of frame instance "frame2" is the sum of values for slots "slot1" and "slot2" of frame instance "frame1", then this relationship can be represented by an "adder" constraint between the three slots. Constraints do not state a direction for the computation; rather, they maintain relationships by computing missing

values for slots taking part in a constraint relationship and signal conflicts if the relationship is overly constrained with inconsistent attribute values. A constraint-based system can be viewed as a network of devices connected with wires. Data values may flow along the wires (slots) and computation is performed by the devices (constraints). A device computes using only locally available information, and places newly derived values on other, locally attached wires. In this way computed values are *propagated* [79] from one wire to another. The advantage of such a system is that a single relationship can be used in more than one direction. The slots taking part in a relationship are not labeled inputs or outputs; a constraint will compute with whatever values are available, and produce as many new values as it can (within limitations of the deduction process).

Constraints can be declarative or procedural in nature. The "adder" constraint described above is an example of a declarative constraint. A procedural constraint is one where a program is invoked to determine the value for a particular slot when values are available for a subset of the remaining slots taking part in a constraint relationship. A conflict is said to occur in a constraint-based system when one of the following conditions exist while setting the value of a frame's slot:

- The new value is not being set by the same entity that set the old value or
- The slot already has a value that is inconsistent with the new entry.

A more detailed description of how Constraint-Based Systems can be implemented and invoked in practice is given in [79].

2.7 Summary

Literature in five different areas was reviewed in detail. In the area of Simultaneous Engineering it was indicated that recent work had been concentrating on developing computer-

based environments for implementing this concept. In the area of Computer-Aided Design, three different models of the design process were reviewed: Design-Evaluation-Redesign, Refinement and Constraint Propagation Design, and Multiple Agents Design. In the area of Computer-Aided Process Planning, three kinds of systems were reviewed: Variant, Semi-Generative, and Generative Systems. It was pointed out that the models of the design process and the process planning systems need to be extended to address manufacturability related issues. In this research the refinement and constraint propagation model of design has been considered along with the generative process planning system to address such issues for components manufactured in small and medium lot-sizes. Part Representation was pointed out to be a central concept in the three areas mentioned above. Two different kinds of part representation schemes were reviewed: Solid Modelling and Feature-Based Schemes. In this research both these schemes have been used and it was shown that an integration of the two schemes is necessary to achieve the goals of the Simultaneous Engineering concept. Finally, Artificial Intelligence played an important role in this research and the AI-based concepts and techniques used in this research were briefly reviewed.

Chapter 3

COMPUTER-AIDED SIMULTANEOUS ENGINEERING

3.1 Introduction

A methodology is proposed to address the Simultaneous Engineering concept for components manufactured in small and medium lot-sizes. Machining-related concerns are identified as important elements of this concept and one way to identify and systematically classify these concerns is described. Finally, a conceptual and schematic framework of a computer-based design environment is presented to implement the Simultaneous Engineering concept.

3.2 Proposed Approach to Simultaneous Engineering

To date, two different approaches have been proposed for the Simultaneous Engineering concept. The first approach is to design parts to be manufacturable in existing manufac-

turing facilities[3.1]. The second approach is to concurrently develop the design of the part[3.2] and the design of the facility and the process plan for the facility.[3.3] These two approaches define the two ends of a "Simultaneous Engineering spectrum" in terms of the degree of simultaneity that is achieved. In the second approach, all three tasks namely product design, facility design and process planning are done simultaneously and "maximum simultaneity" is achieved. In the first approach, facility design is not performed and only product design and process planning is performed concurrently. This approach achieves "minimum simultaneity".

Economic considerations normally dictate the approach to be undertaken. For example, for components manufactured in large lot-sizes, it is usually economically feasible to take the second approach to realize the Simultaneous Engineering concept. However, for components manufactured in small and medium lot-sizes, economic considerations often make it infeasible to completely design a new facility for each type of component. Instead of designing these components for existing manufacturing facilities (the first approach), a more viable alternative proposed in this thesis is to achieve the Simultaneous Engineering concept in two stages.

The first stage, entitled Commodity Sourcing, involves designing special manufacturing cells based on existing part designs such that components manufactured in small and medium lot-sizes can be produced in these cells at the lowest possible production cost. Manufacturing cells are becoming increasingly cost effective for such components and are being preferred over other means for processing these parts (such as Flexible Manufacturing Systems) [82]. The second stage involves ensuring that parts are designed or redesigned such that they are "manufacturable" in these specially designed facilities for the lowest

[3.1] The Design for Manufacturability concept [11,22,81].
[3.2] Product design.
[3.3] Process design.

possible **product development cost.** In terms of the Simultaneous Engineering spectrum presented above, this approach would fall in between the two extreme approaches because it achieves a "moderate amount of simultaneity." Unlike the first approach described above, facility design is performed in this case. However, since facility design does not occur concurrently with the product design and process planning activities as in the second approach described above, maximum amount of simultaneity is not achieved. The advantages of realizing the Simultaneous Engineering concept in two stages are as follows:

1. **Improved Product Reliability.** The increase in reliability is due to the development of common engineering designs for similar parts, fewer shop floor operators handling the piece parts, and increased familiarity of operators with the piece parts.

2. **Reduced Product Design, Process Planning, and Machining Time.** The reduction in design, planning, and machining time can be attributed to utilization of existing part designs wherever possible, development of part designs satisfying manufacturing requirements, development of common and generic NC layouts for machine tools, and development of fixture and cutting tool matrices that minimize setup times.

3. **Improved Quality and Cost of Rough Material.** The improvement in quality and cost of rough material is due to the consolidation and reduction of total number of rough material suppliers, and the timely receipt of raw material leading to a reduction in inventory levels.

4. **Improved Machine, Durable, and Perishable Tooling Utilization.** The improvement in utilization is due to the development of fixture, gauge, and cutting tool designs that are based on the common characteristics of a group of parts. This also

contributes towards reduced inventory, maintenance, and storage requirements for such parts.

Although the second step of the proposed approach is similar to the first approach described above, the main difference from a Simultaneous Engineering standpoint is: whereas the machining-related concerns reflected in the proposed approach carry more weight because they are the concerns that arise if the part were to be manufactured in the lowest possible production cost facility, the same cannot be said about the concerns that arise if the part were to be manufactured in an existing manufacturing facility. Due to this, the designer would be less willing to resolve the concerns arising due to an existing manufacturing facility by making changes at the product design end of the product development process.

One way of satisfactorily performing the first stage activities is to use Group Technology concepts [83] to design the special manufacturing facilities. The traditional approach to performing the second stage activities to "ensure manufacturability" is to allow close personal or computer-assisted interaction between product designers and process designers[3.4]. However, this approach is often unsuitable for several reasons:

- Product designers can interact with only those process designers familiar with the capabilities of the specially designed manufacturing facilities.
- Process designers familiar with the capabilities of the specially designed manufacturing facilities are small in number and consequently in heavy demand.
- Product designers are not always capable of determining when to approach process designers during product design to resolve machining-related concerns.

[3.4]In this case, same as process planners.

- The most suitable process designers to interact with may not be located at the same place as the product designers.

Since the dearth of "knowledgeable" process designers is one of the main drawbacks of the above scheme, another approach to performing this activity is to develop an "intelligent" computer-based environment [84] that cooperatively assists designers during product design. The computer-based environment would encapsulate the knowledge of process designers familiar with the capabilities and machining-related concerns of the specially designed manufacturing facilities. The environment would serve as a computer-based associate for product designers and it would cooperatively assist them in creating, modifying, and refining manufacturable product designs at all stages in the product design process.[3.5] The design environment can be built by extending existing computer-based models of design (reviewed in Chapter 2) to incorporate manufacturability concerns. An important advantage of the design environment is that the task addressed is sufficiently general to encompass the first approach to the Simultaneous Engineering concept described above. Thus, from an implementation standpoint, this design environment can also be used (if necessary) for implementing this approach to the Simultaneous Engineering concept. However, since the focus is on the Simultaneous Engineering concept for components manufactured in small and medium lot-sizes, the main considerations for developing the design environment are derived from its use to "ensure manufacturability" under the approach proposed in this research.

Group Technology is a fairly well-established concept for performing Commodity Sourcing [83]. The research reported in this thesis concentrates mainly on developing the framework of the computer-based environment required for the second stage activities.

[3.5] Not just critique product designs after they have been completely developed.

3.3 Identification and Classification of Machining-Related Concerns

A part is deemed to be manufacturable in the specially designed facility for components manufactured in small and medium lot-sizes if no **machining-related concerns** arise when the part has to be processed through that facility. Machining-related concerns are representative of the interaction effects between the product and process design tasks. The success of any computer-based design environment developed for the second stage activities depends on its ability to identify these concerns. This section defines the concept of machining-related concern and presents a systematic way to identify and classify these concerns.

Machining-related concerns in this research are viewed as "problems" arising at the manufacturing end of the product development spectrum because of certain part design characteristics. The problems may vary from "requiring special-purpose machine tools" to "requiring extra operator attention when parts are being processed." However, one common characteristic of all machining-related concerns is the significant effect on production cost, processing time or quality of parts produced if they are not suitably resolved.

One way of identifying machining-related concerns is by determining the various sources of the "problems." For the specially designed manufacturing cells, the various sources of the "problems" are entities that place restrictions on the processing of parts through these manufacturing cells. The machining-related concerns can, therefore, be identified as arising due to

- the *machines* in the cell,
- the *fixtures* in the cell,

- the *cutting tools* in the cell,
- the *gauging equipment* in the cell, and
- the *material-handling equipment* in the cell.

Based on such a classification scheme, the following demonstrative machining-related concerns were identified in a particular domain:

1. **Machining-related concerns that arise because of the characteristics of the machines in the cell:**
   - Tolerance specifications on turns, bores, through holes and blind holes should be within machine capabilities,
   - Length and diameter of part, turns, faces, bores and holes should be within machine capabilities,
   - Through holes should be cored wherever possible,
   - True position tolerances on through and blind holes should be within machine capabilities,
   - Number of turns and bores to be machined in a sequence should be compatible with the type of machines available in a cell,
   - Monthly lot-sizes should be within machine/cell capabilities, and
   - Balancing requirements should be within machine/cell capabilities.
2. **Machining-related concerns that arise because of the characteristics of the fixtures in the cell:**
   - Projections on clamping surfaces should be appropriately spaced to allow fixtures to be properly located;

- Flats or cut-outs on turns preventing fixtures to be suitably mounted should be avoided;
- Clamping surfaces forming thin walls should be avoided;
- Bore sizes should enable complete contact along the surface of a speed-grip bushing;
- Holes should be positioned at an angle that allows the part to be fixtured for easy access.

3. **Machining-related concerns that arise because of the cutting tools in the cell:**
   - Only straight-sided grooves can be made in the cell. Taper-sided grooves are difficult to machine, and should not be made in the cell.
   - Fillet radius specifications should be within the capabilities of the cutting tools provided in the cell.
   - Bore and internal faces diameters below a certain value are difficult to manufacture in the cell.
   - The lengths of turns, bores and holes should be within the tool overhang limits of the cutting tools in the cell.
   - If interrupted cuts have to be made, then there is a lower limit on the size tolerance that can be achieved.
   - Very small chamfers and corner breaks limit the largest feedrate that can be used to machine the part.
   - Small fillet radius specifications limit the range of "feedrates" and "cutting speeds" that can be used for certain operations.

- A sequence of small fillet radii to be manufactured in one setting will drastically reduce cutting tool life and increase cutting tool cost.
- Certain size tolerance specifications severely restrict the range of "feedrates" and "cutting speeds" that can be used.
- Certain surface texture specifications severely restrict the range of "feedrates" that can be used.
- Certain surface texture specifications require an extra finish pass to achieve the required surface finish.
- The material chosen for the part should be compatible with cell capabilities.

4. **Machining-related concerns that arise because of the gauging equipment in the cell:**
   - Bore, turn, and hole diameter specifications should eliminate the need to make new gauges and should correspond to standard diameters that can be measured in the cell.
5. **Machining-related concerns that arise because of the material handling equipment in the cell:**
   - The weight of the rough stock should be within the capabilities of the material-handling equipment available in the cell.

For most of the concerns listed above increase in production cost or processing time seems to be the important factor, and the issue of decrease in the quality of parts produced is only addressed indirectly through these two factors. For example, machine capabilities to consistently maintain certain turn or bore tolerances is one measure of the quality of

parts that can be produced in a particular facility. However, this concern is only reflected indirectly through a perceived increase in production cost due to the addition of an extra grinding operation. This currently is a limitation of the proposed approach and indicates that a more detailed knowledge acquisition task is necessary in order to determine the machining-related concerns that are indicative of a direct effect on quality.

When the machining-related concerns are viewed in isolation, they vary in complexity (in terms of ease of identification) from the simple to the most complex concerns. When considered within the context of the Simultaneous Engineering concept, even the most simplest machining-related concern becomes quite difficult to detect. For example, concerns dealing with tolerances below the capabilities of a manufacturing facility can be classified as a relatively simple machining-related concern. Yet, during the product design process, designers (overwhelmed by the wealth of information that they deal with) are not able to specify appropriate dimensional tolerances for part features. Thus, all the concerns listed above warrant development of the design environment that can enable product designers to systematically address these concerns.

Although the methodology described above enables the enumeration of various machining-related concerns, it does not show how one should determine whether these concerns have arisen for a particular product design. There are different ways of addressing these concerns and suitably determining whether these concerns have arisen is the main responsibility of the computer-based design environment. One aspect of the basic requirement for the design environment is to appropriately model these concerns and provide feedback to designers when the concerns arise.

3.4 Conceptual Framework of the Design Environment

To establish the framework of the computer-based design environment, it is important to first identify when and how this design environment will be used. Figure 3.1 is a schematic diagram of the traditional sequential product development process. Five important stages can be identified in the product development process, namely: conceptual design, layout design, detailed design, pre-production and production. It has been fairly well established that the chances of suitably resolving machining-related concerns are greater when they are brought up as early as possible in the product development process (e.g. at the conceptual design stage). However, because of the high level of abstraction of the part design in the early stages of the design process, the task of determining whether certain machining-related concerns have arisen becomes more difficult as one moves into the early stages of the product design process. This is especially true for those machining-related concerns for which at least some knowledge of how a part will be processed through a facility has to be known before it can determined that these concerns need to be resolved. Due to this, the approach taken in this research is to start by addressing the issue of developing manufacturable product designs at the lower stages of the design process and progressively move into the higher stages as one obtains a better understanding of the task. Extensive component design (the focus of this research) normally takes place at the detailed design stage and this is one of the stages where the product designer needs assistance in developing manufacturable product designs. The scope of the design environment developed in this research has been restricted to providing assistance to the product designer at the detailed design stage. Although the issue of providing feedback to designers about machining-related concerns at the conceptual or layout design stage is equally important, this is not addressed at present. In terms of the stage-wise interactions, the main intention of the design environment developed in

this research is to minimize the number of iterations that usually takes place between the detailed design, pre-production, and production stages under the traditional, sequential product development process.

For developing the framework of this design environment, it is assumed that the activity of designing a part proceeds through a sequence of successive refinement steps, starting from an abstract representation and ending in a concrete physical realization of the part (the refinement and constraint propagation model of design [30]). Each stage of the refinement process is modeled by the conceptual framework as consisting of three different feature spaces, namely functional, structural, and manufacturing spaces as shown in Fig. 3.2. Conceptually, a point in the functional space represents the result of decisions made about the functional aspects of the design of a part. The structural space expresses the results of a designer's efforts to obtain a physical realization of design intentions expressed in the functional space. The attributes of the manufacturing space represent the structural and functional aspects of manufacturing facilities. Manufacturing engineers use the representation of a facility in the manufacturing space to derive manufacturability-related concerns and express the effect of these concerns in the functional and structural spaces.

With the three spaces explicitly defined, the task encompassed by the Simultaneous Engineering concept at any stage in the product refinement process is iterative and, under ideal conditions, can be viewed as proceeding as follows:

1. **Reason about functionality**

   Given a particular design goal, designers reason about the functional requirements that need to be satisfied and establish values for the various attributes that define the functional space. These attributes collectively define the desired end effect of this activity. In general, the end result of the activity can either be a point or a region in

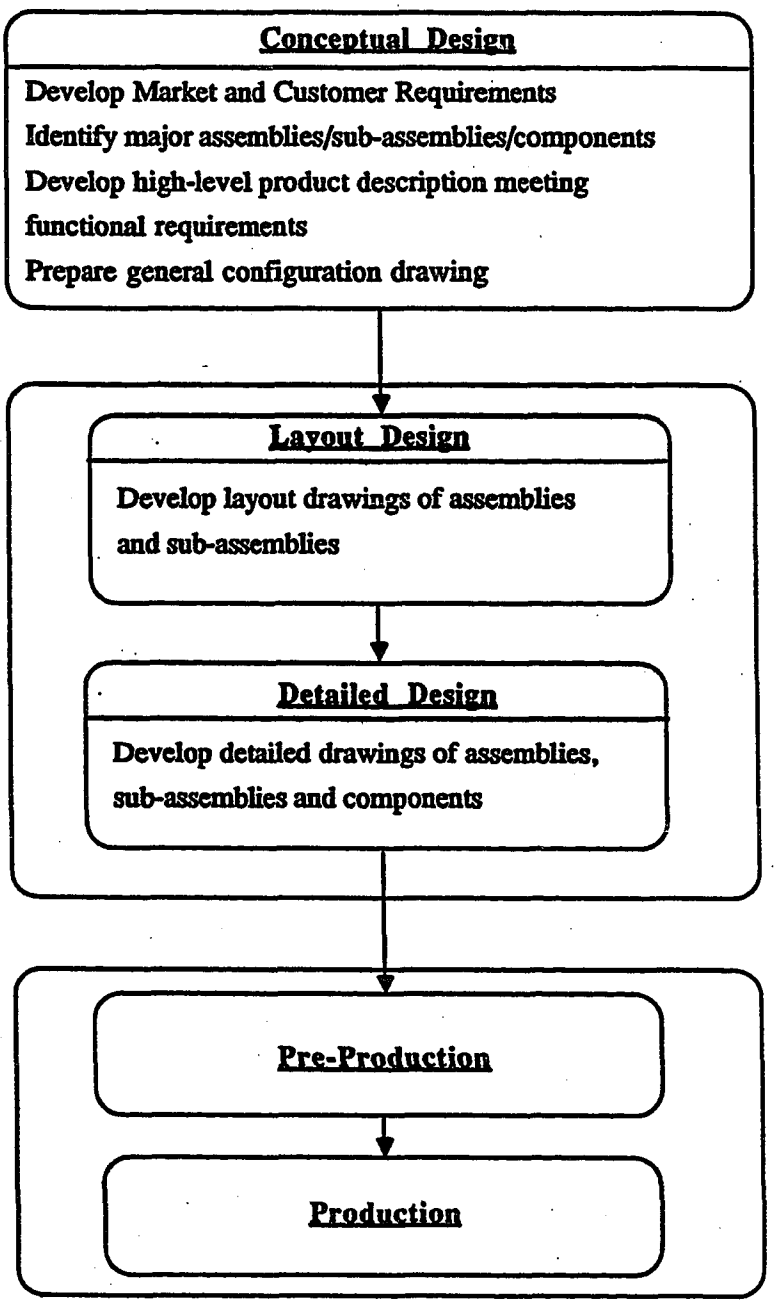

Figure 3.1: Schematic Diagram of Product Development Process

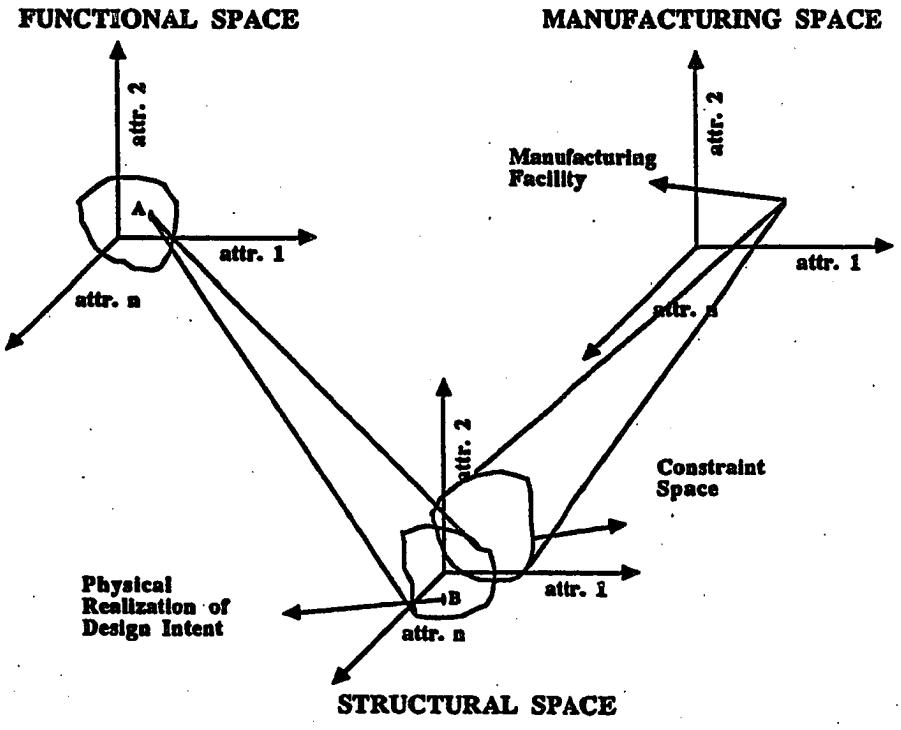

Figure 3.2: Feature Spaces at a Single Abstraction Level

the functional space as shown in Fig. 3.2.

2. **Select a structural representation**

   If a region has been obtained in the functional space, designers select a particular point (say, point A in Fig. 3.2) within this region based on a design criteria established by the designer. The point is then mapped by the designer into the structural space to obtain a physically realizable design satisfying both functional and structural requirements. The end result of the second mapping activity can again be a point or region in the structural space.

3. **Express design intent**

   If a region has been obtained in the structural space, a particular point (say, point B in Fig. 3.2) within the region is selected based on a design criteria established by the designer. The two points A and B in the functional and structural spaces together define the end result of the design task satisfying both functional and structural requirements.

4. **Ensure manufacturability**

   Under the Simultaneous Engineering concept, the designer's intent expressed in the functional and structural spaces (the pair of points) must also satisfy manufacturability requirements. Conceptually, all manufacturability-related concerns derived from the manufacturing space can be viewed as implicitly defining a constraint region in the functional and/or structural space (Fig. 3.2). For machining-related concerns, this region would be primarily described in the structural space.

   If the design in structural space (point B) lies outside the constraint region, then some machining-related concerns have arisen and suitable changes (if possible) must

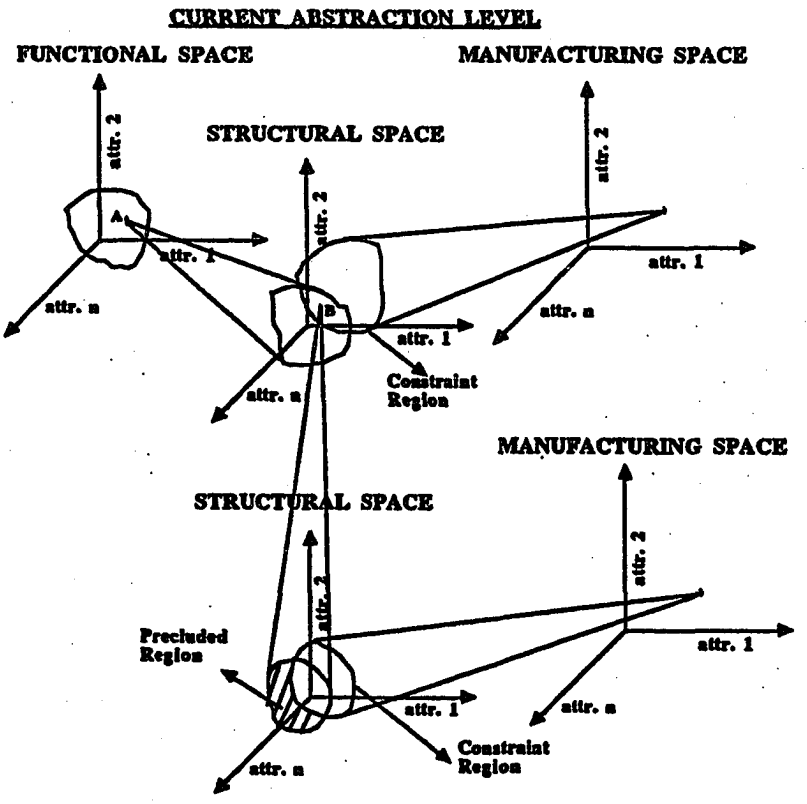

Figure 3.3: Feature Spaces at Multiple Abstraction Levels

be made in the structural and/or functional space to satisfactorily resolve these concerns. It is also possible that the end result of product design decisions lie within the constraint region (for the current abstraction level), but portions of the design refinement at lower levels of abstraction lie outside the constraint region corresponding to those levels (see Fig. 3.3). To prevent machining-related concerns from arising as the product design is refined, problematic design refinements (wherever possible) should be precluded at the current abstraction level.

5. **Provide feedback to designer**

The designer gets feedback about the concerns that have arisen, along with possible changes in the structural/functional space to resolve these concerns. The changes requested are evaluated by the designer to study their implications in the structural and functional spaces. This completes the first cycle of the iterative process and the process continues until either

(a) a satisfactory point is found that lies within the constraint region described in the functional and/or structural space or

(b) no further changes can be accommodated by the designer in the functional/structural space (in which case machining-related concerns continue to exist, and the product development cost increases).

Within the context of this conceptual framework, the focus of the computer-based design environment is to determine if the physical realization of a designer's intent at a particular refinement stage is within the constraint region defined for that stage. It is assumed that the design activity has proceeded up to a certain refinement stage (not necessarily finished) and the result in the functional and structural space has been tentatively identified. The input to the design environment is a description of the end result of the design activity in the structural space and a description of a manufacturing facility in the manufacturing space. The design environment primarily acts as a computer-based associate to the designer, indicating the concerns that have arisen from a machining standpoint (for a particular facility) and suggesting possible changes in the structural space that can resolve these concerns. The process of developing the design or evaluating the ramifications of suggested product design changes in the functional and/or structural space is not addressed and is the responsibility of the designer.

3.5 Schematic Framework of the Design Environment

The constraint region in the structural space is derived from machining-related concerns and is only defined implicitly. The fundamental requirement of the design environment is the explicit determination of whether a point in the structural space lies within such a constraint region at any refinement level. This process involves the development of:

- explicit product and manufacturing facility models to capture the structural and functional specifications of the product design and a particular manufacturing facility and
- a reasoning system to determine if the proposed product design at any stage in the design process raises any machining-related concerns due to the structure and function of the given manufacturing facility.

A Model-Based Reasoning system [70,71] meets this fundamental requirement and is used to develop the framework of the proposed computer-based design environment. Figure 3.4 is a schematic diagram of the design environment. The product model captures the end result of a designer's intent by identifying the values chosen for the structural space design variables (shape, geometry, material, etc.) at any stage of the design cycle. The manufacturing facility model captures the structural and functional aspects of a particular facility as represented in the manufacturing space and contains extensive knowledge about the machines, cutting tools, fixtures, and other equipment present in the facility. The product and manufacturing facility models are the main inputs to the Manufacturability Advisor system (the reasoning system) which processes the part for the manufacturing facility and provides feedback to designers about any machining-related concerns that arise.

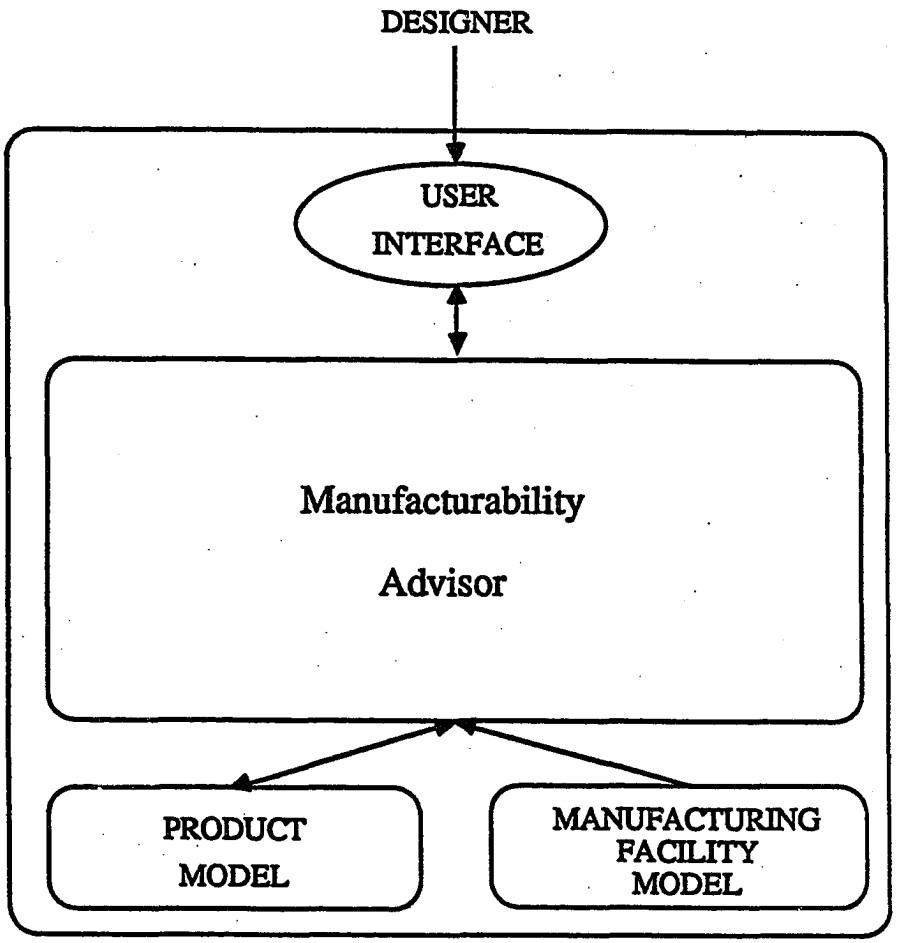

Figure 3.4: Schematic Diagram of Computer-Based Environment

3.6 Summary

A two-stage approach to achieving the Simultaneous Engineering concept for components manufactured in small and medium lot-sizes has been presented. The need for a computer-based design environment to perform the second stage activities of "ensuring manufacturability" was established. A systematic approach to identifying and classifying *machining-related concerns* was presented, and it was shown that all these concerns are equally complex from a Simultaneous Engineering standpoint. A "conceptual framework" was first described to establish the basic requirements of the design environment. It was then shown that model-based reasoning systems can satisfy these requirements, and a schematic framework of the design environment based on this approach was presented.

Chapter 4

PRODUCT AND MANUFACTURING FACILITY MODELS

4.1 Introduction

The models underlying the model-based approach to developing the computer-based design environment are described in detail in this chapter. The chapter begins with a presentation of the basic requirements of the product and manufacturing facility models and briefly describes the modeling methodology undertaken. Various facets of the product and manufacturing facility models that have been developed are then described in detail.

4.2 Modeling Methodology

The product and manufacturing facility models should satisfy the following two basic requirements respectively:

- explicitly capture designer's annotations of a part drawing (the structural space) and

- explicitly capture structural and functional specifications of machines, fixtures, cutting tools, gauges and material-handling equipment available in a manufacturing facility (the manufacturing space).

For example, the product model should enable the designer to specify product design characteristics generated by him at any stage in the design process. These characteristics have been traditionally represented in the form of a part drawing blueprint. Figure 4.1 is a representative example of a typical blueprint. The type of product design characteristics developed are:

- commodity and material specifications,
- rough and finished part geometry,
- dimensions, tolerances, and geometric tolerances,
- datum surfaces and datum targets,
- surface finish and machining requirements,
- special processing specifications, and
- casting dimension and tolerance specifications for rough stock.

A combination of feature-based, geometric, and process performance models meet the basic requirements outlined above. A feature-based model is important for two reasons:

1. Product and process designers constantly reason in terms of features, and this modeling approach greatly facilitates product and manufacturing facility description. For example, product designers reason about placing "geometric tolerances" on a "face" to ensure proper mating conditions or reason about seating O-ring seals using "grooves."

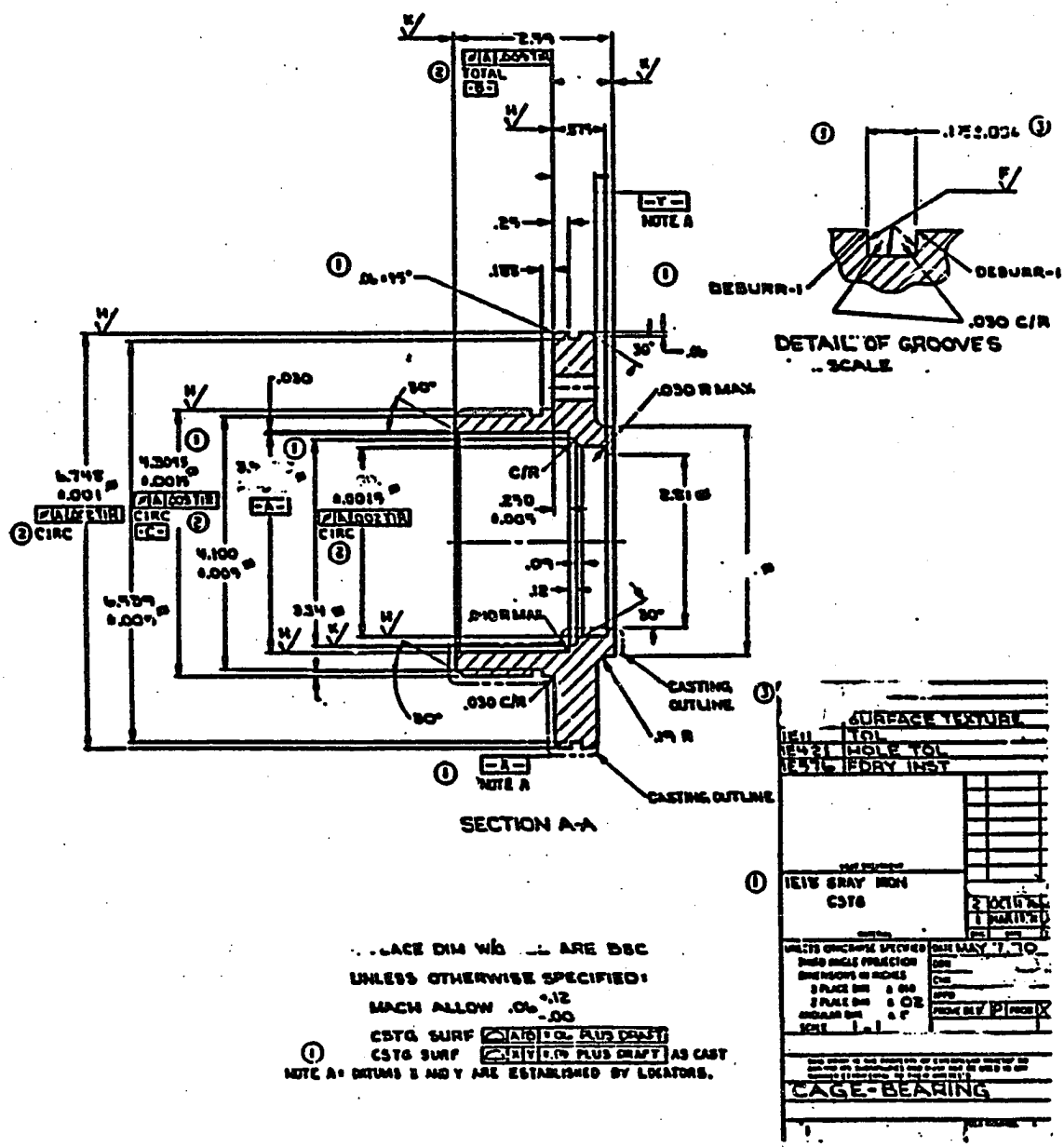

Figure 4.1: Product Design Blueprint

Process designers try to determine "boring tools" for machining "internal diameters" and look for "outer diameters" that can be used to hold a part in a "fixture."

2. Most of the machining-related concerns pertain to individual product and facility features or to the relationship between these features. Therefore, the explicit representation of features enables these concerns to be determined easily.

Geometric models are necessary in order to determine the macroscopic effects of the structural and functional characteristics of the components of a particular manufacturing facility. They are useful for conducting (if necessary) macroscopic simulations of certain cutting tool motions or to check for interference when a part is held in a particular fixture [85]. These activities cannot be conducted using the feature-based model because it is an "incomplete" model. The "incompleteness" arises because not all kinds of geometry-related questions can be answered by using such models.

Process performance models are useful because they enable one to predict the outcome of a particular operation before the actual cut is made. They complement the geometric models by helping to determine the microscopic effects (surface error, surface texture) of the structural/functional specifications of a manufacturing facility.

4.3 Feature-Based Model

Features are parametric descriptions of entities that closely match the way product and process designers view portions of products and manufacturing facilities during the reasoning process. They are higher order abstract forms[4.1] that are used to reason about the geometry, topology, and other characteristics of designed artifacts and originate from the heuristics used while performing these activities [62]. Features are process-activity entities and in the

[4.1] As compared to lines and edges.

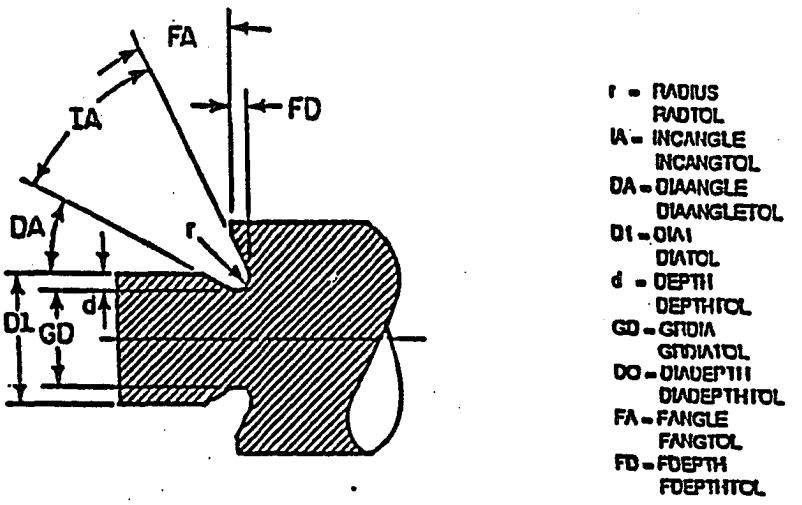

Figure 4.2: Parametric Description of Relief-Groove

domain under consideration: the process is machining and the activity is primarily product and process design. Some of the part characteristics outlined in the previous section are examples of product features. Structural and functional characteristics of fixtures and machines relevant to the product and process design activity are examples of manufacturing facility features. Figure 4.2 is a parametric description of one type of product form feature, namely relief-groove.

Frame- and Constraint-Based Systems have been used to develop the feature-based model. Due to the one-to-one corespondence between the concept of objects or classes in frame-based systems and the concept of features described above, frame structures have been found to be suitable in developing the feature-based model. Features correspond to frames and feature attributes are represented as the slots of the frame.[4.2] Product designs and the manufacturing facility are represented as a union of feature instances, and the product development process is modelled as the creation, deletion, modification, or

[4.2]To enable easy identification, "frame" and "frame attributes" in the text appear in italics.

refinement of product feature instances.

Constraints are used to represent relationships between attributes (slots) of feature instances. In the development of the feature-based model, constraint-based systems have been used primarily to maintain consistency while generating feature instances and to propagate feature attribute values when "adjacency relationships" [86] are established between feature instances.

The remainder of this section describes the following aspects of the feature-based model in greater detail: important product and manufacturing facility feature classes, categories of various feature attributes, and several types of operations that can be performed on features. Appendix A provides a parametric description of representative examples of feature classes described in the next two sections.

4.3.1 Product Feature Inheritance Hierarchy

The inheritance hierarchy underlying the frame-based system used to model product features is shown in Fig. 4.3. The root class is *product.features* and the various product features can be categorized as: form features, precision features, or material features.

Form Features

Form features are used to represent the rough and finished part geometry in the product model. A union of primitive and compound feature instances (see Fig. 4.3) represent the rough and finished part geometry. *Primitive.features* represent the smallest individual entity that can be used to represent the form of a product. *Compound.features* represent a union of two or more primitive features.

Since the scope of this research encompassed only rotational components, this section primarily describes rotational primitive and compound features. Primitive features are

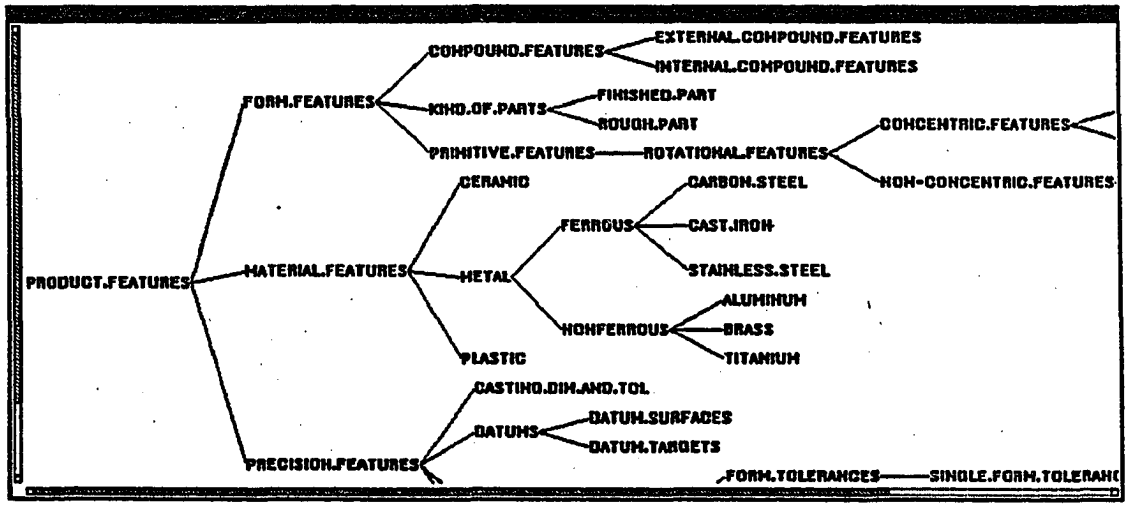

Figure 4.3: Product Features Inheritance Hierarchy

broadly divided into two sub-classes: Concentric and Non-Concentric Features (see Fig. 4.3). *Concentric.Features* are rotational features whose axis of rotation coincides with the primary axis of rotation of the part. *Non-Concentric.Features* are rotational features whose primary axes of rotation are different from, and non-coincidental with, the primary axis of rotation of the part.

Concentric features represent the class of features used to develop the basic shape of the part. One can distinguish between external and internal concentric features of the part. The portion of the inheritance hierarchy of the frame-based system rooted under *Concentric.Features* is shown in Fig. 4.4. If a coordinate reference frame is formed such that the y-axis coincides with the primary axis of rotation of a part, then certain concentric feature categories can be further classified into sub-classes depending on the direction of their surface normal.[4.3] This categorization is made primarily to facilitate the establishment

[4.3]Or, the y-component of the surface normal.

of adjacency relationships between various form features of a part.

Non-concentric rotational features can be broadly classified into *depressions* and *protrusions*. One can distinguish between internal and external depressions and through and not-through depressions. For example, a circular hole is an internal-through depression, but an external keyway is an external-not-through depression. Similarly, one can distinguish between internal and external protrusions and symmetric and not-symmetric protrusions. Figure 4.5 presents the portion of the inheritance hierarchy of the frame-based system rooted under *Non-Concentric.Features*. Figure 4.6 shows a particular part design with only its primitive form features identified. Each feature instance identified in Fig. 4.6 is created by making a unique instance of the corresponding form feature from the form feature inheritance hierarchy.

Compound features are made up of either a combination of two or more concentric primitive features or a combination of concentric primitive features with protrusion and/or depression features. Non-concentric features use concentric features as reference features and are normally specified as part of a compound feature. For example, the two reference features for the through hole shown in Fig. 4.6 are the two faces at either end of the hole. Presently, only two types of compound features are distinguished: *External.Compound.Features* and *Internal.Compound.Features* (see Fig. 4.3). Future extensions to the form feature hierarchy could refine these *Compound.Features* to distinguish between the type of primitive features that make up a compound feature. The main advantage of this specialization is that one can define attributes of interest for each specialization of the compound feature. For example, if one particular specialization of *Compound.Features* were to be made up of through holes, faces and outer diameters then a "thin wall" formed between the hole and the outer diameter could be an attribute of the *Compound.Feature*.

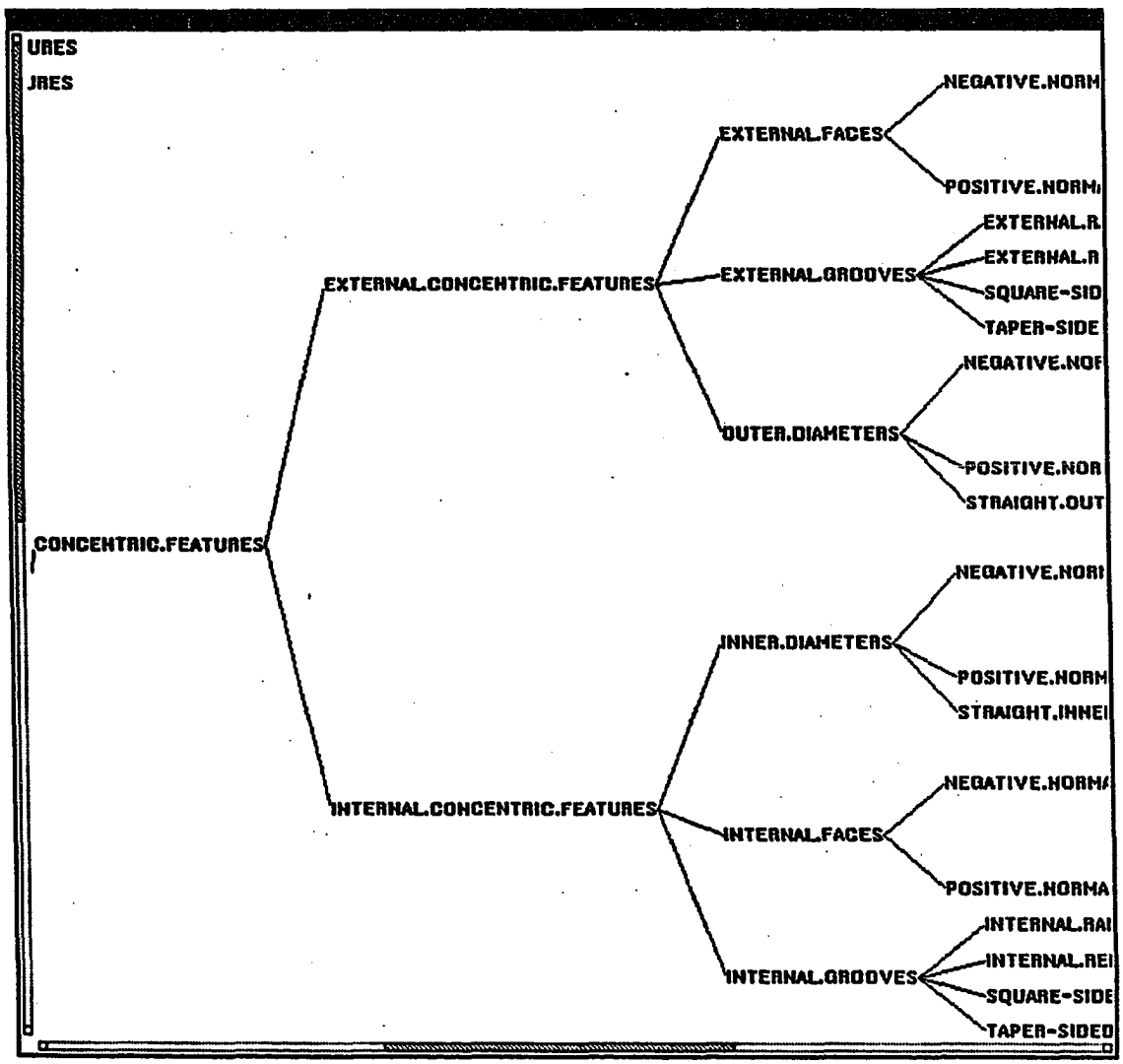

Figure 4.4: Concentric Features Inheritance Hierarchy

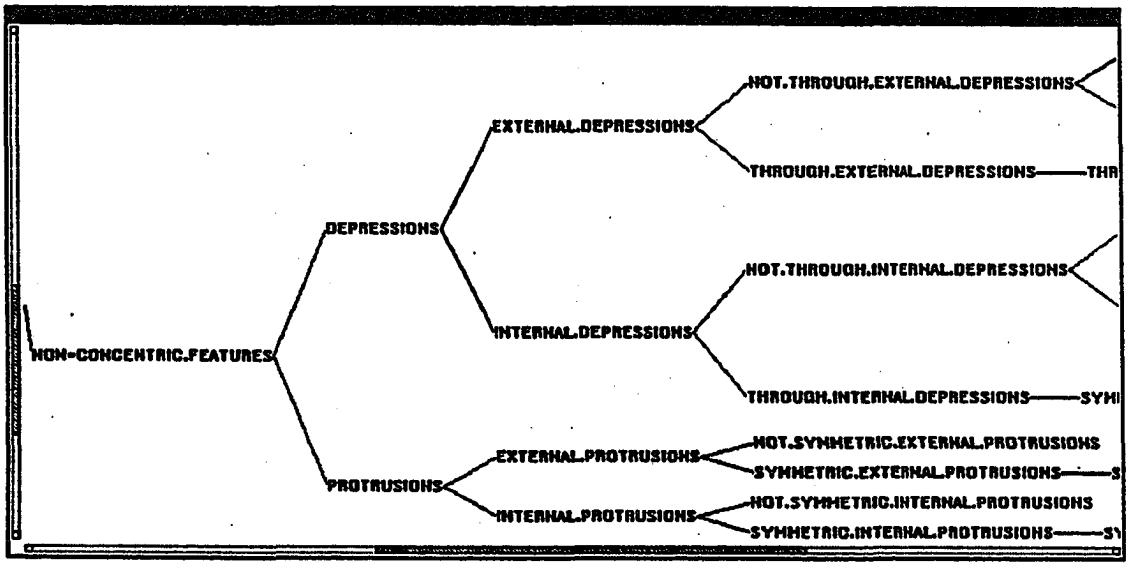

Figure 4.5: Non-Concentric Features Inheritance Hierarchy

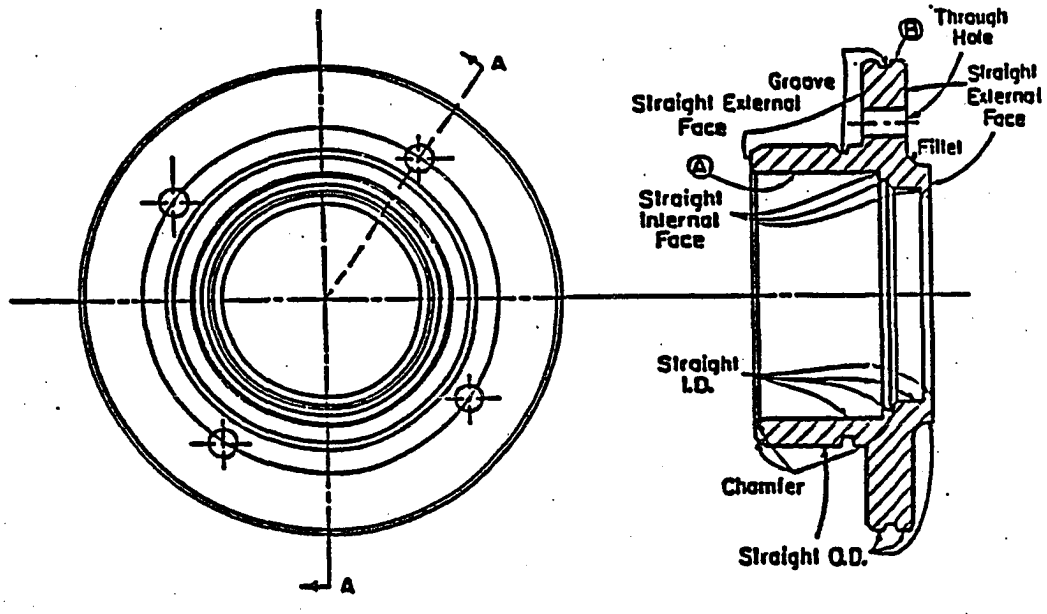

Figure 4.6: Primitive Features of a Part

Such a specialization of *Compound.Features* with an explicit representation of "thin walls" is especially useful when thin walls are of constant interest to the reasoning subsystem.

Material and Precision Features

The material composition, grade, and properties of a part are specified by the Material Features hierarchy. The portion of the inheritance hierarchy rooted under *Material.Features* is shown in Fig. 4.3. The material characteristics of a part are specified by indicating the appropriate material from this class of features.

Precision features are the class of features used to indicate how much a part can vary from its true form and still be acceptable. The portion of the inheritance hierarchy rooted under *Precision.Features* is shown in Fig. 4.7.

Datums are special form features of a part used as reference or starting points for the purpose of controlling other features on the same part. When form features are used as a reference or starting point for measurments, they are termed *Datum.Surfaces*. The form feature of the finished part most important to the function of the part is the Primary Datum Surface. The primary datum surface is labeled datum 'A' (see Fig. 4.1). Often more than one datum surface is needed to make measurements. Secondary and tertiary datum surfaces are formed by using other part form features and are labelled 'B' and 'C', respectively. *Datum.Targets* are established to indicate how to fixture the part on cast or forged surfaces. The main purpose of datum targets is to ensure that the primary datum surfaces for all parts are generated (by machining) in a consistent manner. Datum targets are usually labelled 'X', 'Y' and 'Z' (see Fig. 4.1).

Geometric.Tolerances are the primary means for designers to unambiguously communicate allowable deviations on the form or geometry of a part. Product designers indi-

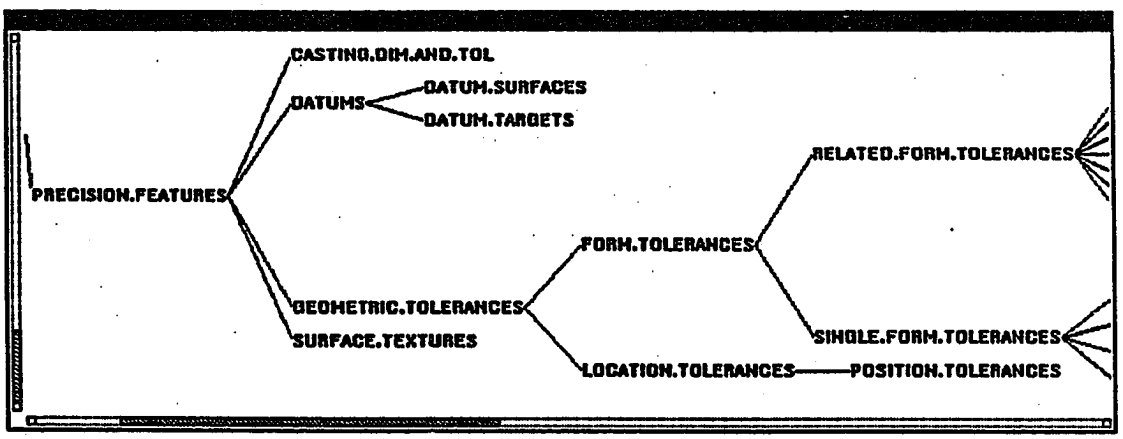

Figure 4.7: Precision Features Inheritance Hierarchy

cate the geometric tolerance on a part design blueprint by using "feature control symbols" (rectangular boxes in Fig. 4.1). Geometric tolerances can be broadly classified into two types: *Location.Tolerances* and *Form.Tolerances*. *Location.Tolerances* are used to indicate allowable tolerances on the location of a particular form feature of a part. For example, *Position.Tolerance* instances indicate allowable tolerances on the location of the through holes in Fig. 4.6.

Form.Tolerances are used to indicate allowable tolerances on the form of various primitive features of a part. *Single.Form.Tolerances* represent the class of tolerances for which no datum surface is needed for their measurement. These tolerances are normally used to refine the tolerances on the size attributes of any form feature and must be smaller in value than the size tolerance on these attributes.

Related.Form.Tolerances represent the class of tolerances for which a datum surface must be specified in order to measure the tolerances. These tolerances are sometimes measured against more than one datum surface. In such cases, datum surface precedence is indicated

by the order in which the datum surfaces are mentioned. The ordering of datum surfaces is important to ensure that the tolerance is legal and not redundant or overspecified when considered along with the other specified tolerances.

Surface.Texture is used to indicate the roughness (smoothness) of form feature surfaces produced by machining processes (indicated by triangular-shaped symbols in Fig. 4.1). Letters or numbers are used to indicate allowable surface textures. Letters 'F', 'H' or 'K' indicate different grades of surface textures on a Surface Texture Comparator. When a surface texture letter is specified, the machined surface produced must be equal to or smoother than the specified surface on the Surface Texture Comparator. Numbers indicated for surface texture specify the maximum "roughness average" (in micrometers) acceptable.

The *Casting.Dim.and.Tol* feature class is used to indicate default characteristics of cast and forged surfaces. For example, this feature class specifies the additional material provided on cast surfaces for machining (*machining.allowance*) and the default profile tolerances for cast surfaces (*datum.surfaces.profile*) on a finished part.

4.3.2 Manufacturing Facility Feature Inheritance Hierarchy

The inheritance hierarchy underlying the frame-based system used to model manufacturing facility features is shown in Fig. 4.8. The root class is *facility.features* and the various facility features can be categorized as: *cell.features*, *cutting.tool.features*, *fixture.features*, *gauge.features*, *machine.features* and *material.handling.features*.

Cell.Features are used to characterize the various manufacturing cell facilities developed for commodities groups such as bearing cages and pulleys. All the cells designed for a particular commodity class are instances of the corresponding facility class. Feature parameters of the various cells describe the components, layout, and other general cell level characteristics of interest to the product and process design activity.

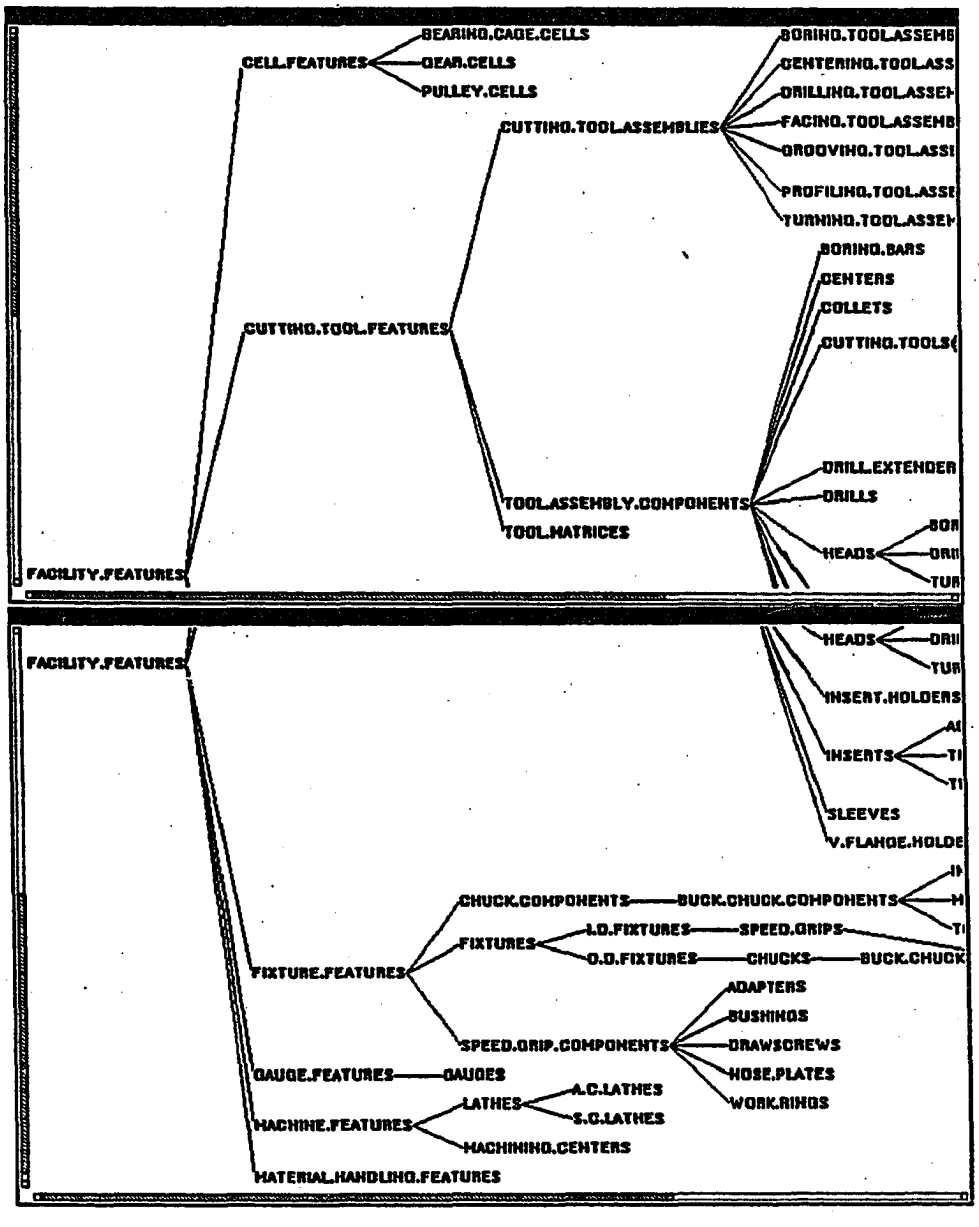

Figure 4.8: Manufacturing Facility Features Inheritance Hierarchy

Cutting.Tool.Features are used to describe the structural and functional characteristics of various cutting tools, cutting tool components, and cutting tool matrices available in a facility. Cutting tool matrices are made up of a number of stations, and some or all of these stations may contain instances of cutting tool assemblies. Cutting tool assemblies are made up of a suitable combination of cutting tool components: *heads*, *collets*, *cutting tools*, *v-flange holders*, etc. Cutting tools in turn are a suitable combination of *inserts*, *insert holders*, *boring bars*, *drills*, etc. Feature parameters such as *insert radius*, *tool overhang*, *minimum bore diameter*, and *drill extender length* capture various structural characteristics of cutting tools. Currently, functional characteristics of cutting tools are not explicitly mentioned but are implicit in the naming convention used for various cutting tool features. For example, *facing.tool.assemblies* are cutting tools to be used in performing the facing operation, whereas *o.d.profiling.tool.assemblies* can be used to perform both facing and turning operations.

Fixture.Features are used to describe the structural and functional characteristics of various fixtures and fixture components used in the manufacturing cells. Fixtures can be split up into *I.D.Fixtures* and *O.D.Fixtures*. *Chucks* are examples of *O.D.Fixtures* and *Speed.Grips* are examples of *I.D.Fixtures*. Figure 4.8 shows the various components of *Buck.Chucks* and *Speed.Grips*. The three main components of Buck Chucks are the master, intermediate, and top jaws. For a given facility, the master and intermediate jaws are permanently fixed to the chuck while the top jaws are quick-change jaws to accomodate different sizes of components. The main components of a Speed-Grip are *adapters*, *nose-plates*, *draw-screws*, *work-rings*, and *bushings*. Again, for a given facility, the adapters and nose plates are permanently fixed to the machine. The draw screws, work rings, and bushings are the components suitably changed to accommodate different types of components.

Machine.Features, *Gauge.Features* and *Material.Handling.Features* are used to characterize the various machines, gauges, and material handling equipment available in a facility. The attributes of *Machine.Features* describe the various fixtures and cutting tools available in a particular machine and capture other machine-level characteristics such as *stroke.length* and *tool.post.home.position*.

Since facilities are not explicitly designed while ensuring manufacturability, the feature-based model for a manufacturing facility is used primarily as a database to retrieve important structural and functional characteristics. Frame-based systems have been used to develop these models, mainly because their expressive representation facilities[4.4] enables one to capture much of the semantic meaning lost in databases. However, as described in [87], one can still obtain the "performance efficiency" of databases by providing a suitable link between the facilty model and a database.

4.3.3 Characterization of Feature Attributes

The various parametric attributes of all product and manufacturing facility features can be classified as belonging to one of the following categories: feature parameters, feature relation parameters, feature operation parameters, and geometric modeling parameters. The slot facet *Slot.Characteristic* identifies the category of a parameter in the parametric descriptions presented in Appendix A.

Feature parameters are used to characterize a particular feature class and play an important role in the reasoning process. Feature relation parameters are of importance in establishing the adjacency relationships between the feature instances of a particular model. Feature operation and geometric modeling parameters are described in Sections 4.3.4 and 4.4 respectively.

[4.4] Interacting classes, composed of multi-valued slots and facets, and organized into hierarchies.

Length, *Diameter*, *Inner.Diameter*, *X-coordinate* and *Y-coordinate* are some of the feature parameters of the *Outer.Diameter* and *Circular.Hole* features described in Appendix A. Dimensional feature parameters constitute an important subset of the class of feature parameter attributes. These feature parameters represent the dimensions and the corresponding size tolerances of the part and components of a manufacturing facility. Dimensions are established by using the standard zero-plane dimensioning scheme. Dimensional feature attributes are also used to set up constraint relationships between the feature parameters of one or more feature instances.

A special kind of Object Oriented Value (OOV) [80], namely a Dimensional OOV is used to represent the value of the dimensional feature attributes. OOV provides a generic way of representing value classes such as dimensions, numbers, symbols, and intervals within the constraint-based system. OOV have "methods" that indicate how various mathematical operations can be performed on various value classes. Each OOV has several operations for the same type of message since the operation performed depends on the value class of the argument(s) passed to it. With the use of OOV, constraints send messages to the values to perform operations instead of applying operations to the values. OOV enables the constraint-based system underlying the feature-based model to be generic; implementation dependent behaviour is attached to the values that are propagated and is not imbedded in the system. Appendix B provides the definition of the dimensional object oriented value and indicates some of the operations that can be performed on it.

Feature relation parameters are used to indicate the adjacency relationships that would exist between instances of two kinds of feature classes. These parameters are used by the constraint-based system to represent constraint relationships between features when their adjacency becomes known in a particular product model. For example, once it is known that an instance of an *External.Face* feature is adjacent to an instance of an *Outer.Diameter*

feature, then equality relationships enable the sharing of relevant feature attribute values (*distance.from.the.zero-plane* and *inner.diameter* for the *External.Face* feature, and *distance.from.the.zero-plane* and *diameter* for the *Outer.Diameter* feature).

4.3.4 Operations on Features

Feature manipulation is performed by specifying various types of operations for features. Methods and Active Values are the primary means of specifying operations on features in the Feature-Based Model. Methods are usually defined and assigned "values" at the highest possible level in the inheritance hierarchy. The methods are changed at the lower levels of the hierarchy either by adding a new local method or by incrementally specializing the methods.

The methods specifying the kinds of operations that can be performed on features can be classified as belonging to one of the following categories: creation methods, modification methods, replacement methods, deletion methods, access methods, constraint methods, user-interface methods, and geometric-modeling methods.

Creation, modification, replacement, and deletion methods, as their names suggest, are the primary means of manipulating features within the feature-based model. The parametric description of features in Appendix A provides examples of such methods for various features. For example, for the feature *Outer.Diameters*, the creation methods are: *Create.New.Instance*, *Generate.Instance.Name*, and *Fill.Parameters.Value*. The *Create.New.Instance* method is called when a new instance of this feature must be created. It illustrates the use of the incremental specialization facility. At the *Product.Features* level, this method involves generation of the name of the new instance and creation of the instance. At the *Primitive.Features* level, an "after method" is added mainly to establish various intra-feature constraint relationships. At the *Concentric.Features* level, another

"after method" is added primarily to establish adjacency relationships between the new and previous feature instances, obtain values (if available) for various feature parameters, and validate the feature.

Access methods are used to obtain various kinds of information about feature instances and relationship between feature instances. For example, consider one particular relationship between the *Through Hole* feature and the largest *Straight O.D.* feature in Fig. 4.6, namely formation of a thin wall between them. An access method for *Straight O.D.* features can be defined to determine and return those *Through Hole* features that form a thin wall[4.5] with the *Straight O.D.* feature in a feature-based model of a part. Thus, if a machining-related concern is "location of holes may lead to the formation of thin walls that are difficult to machine," then this method would be activated before providing feedback to the designer about this concern. Constraint methods are used to establish intra- and inter-feature relationships when feature instances are created. Intra-feature relationships are used to ensure correctness of feature instances created. Some of these methods are indirectly invoked through the use of Active Values attached to feature attributes. User-interface methods are used to update the user-interface to indicate to the product designer the current state of the feature instance being created. Geometric-modeling methods are used to generate geometric models and are described in Section 4.4.

4.4 Geometric Model

The main intent of the research described in this section is to demonstrate how the prerequisite for using a computer-based system that uses geometric models for interference detection and operation simulation can be satisfied within the design environment. The development of the computer-based system for interference detection and operation simula-

[4.5] Suitably defined based on the thickness of the wall

tion has not been performed as part of this research, and is described in [85]. The use of this system requires knowledge about how a part will be fixtured and the operations that will be performed on a part. The reasoning subsystem of the design environment described in Chapter 3 possesses this kind of knowledge. The prerequisite for using the above-mentioned "inteference detection" system within the design environment is to establish a link between the feature-based model and the geometric model of the part and the facility. Although the "interference detection" system has not been used in this research, this section describes one way of generating the geometric model from the feature-based model to satisfy the prerequisite for using such a system. The geometric model developed in this research is based on the Constructive Solid Geometry (CSG) and Boundary Representation (B-REP) schemes [59]. The frame-based solid modeling system VANTAGE [88] has been used to develop the geometric model. Frames are consistently used throughout the VANTAGE system to represent various CSG and B-REP aspects of objects and their inter-relationships. A detailed description of the standard frame structures used to represent the CSG and B-REP models in VANTAGE is given in [88].

Since facility design would have been completed in the first stage of the proposed approach to the Simultaneous Engineering concept, geometric models of the components of the facility would exist at the time of part design. The geometric model of the part is incrementally built while the feature-based model is being created. As indicated earlier, feature instantiation occurs in the following fashion: parameter instantiation takes place, intra- and inter-feature relationships are established, and feature instances are verified and validated. In addition, once the feature instances have been completely created, the portion of the CSG tree (of the part) corresponding to that feature instance is generated. For compound form feature instances, the portion of the CSG tree corresponding to the compound feature is generated from the CSG nodes of the individual primitive features (that make up the

compound feature) through the use of suitable CSG operations. The resultant CSG node in all cases is stored as the value of the *csg.node* attribute of the feature instances.

The CSG nodes of the individual features of a part or components of a facility collectively define the corresponding complete CSG model. Figure 4.9 is the CSG representation generated by the VANTAGE system from the feature-based representation of the part design shown in Fig. 4.10. *Straight.outer.diameter-2-primitive* is an example of the CSG node corresponding to a *Straight.Outer.Diameter* primitive feature, *External.Compound.Features-151408* is an example of the CSG node corresponding to a *Compound.Feature*, and *Finished.Part-151450* is the CSG node corresponding to the entire part design. Applying the boundary evaluation [88] procedure to the CSG model provides the B-REP model of the part. Boundary evaluation involves the generation of the boundary representation for all leaf-nodes and intermediate nodes of the CSG model. The boundary representation of one of the nodes of the CSG model is stored as the value of the "boundary-rep" attribute of the VANTAGE frame structure corresponding to that node [88].

Several form feature attributes are characterized as "geometric-model parameters" or "geometric-model methods" (see parametric feature descriptions in Appendix A): these attributes are used to generate the geometric model of the part. The CSG primitive solids corresponding to feature instances get their parameter values from the feature's parameters. For example, for *Straight.Outer.Diameters*, the primitive solid is a cylinder. Values for the cylinder's radius and height are based on the values for the feature parameters *diameter* and *length*, respectively. In addition, there are six other parameters, namely *rigid.motion.x*, *rigid.motion.y*, *rigid.motion.z*, *rigid.motion.roll*, *rigid.motion.pitch*, and *rigid.motion.yaw* that define a transformation (motion matrix) to be applied to the primitive solid corresponding to a particular feature. The values for these parameters are also dependent on

```
Command: (user::tree)
-------------------------------------------------
NODE      CLASS  LEFT-NODE  RIGHT-NODE  PARENT(S)
-------------------------------------------------
FINISHED.PART-151450          DIF      FINISHED.PART-151449          STRAIGHT.INNER.DIAMETERS-4-PRIMITIVE    NIL
FINISHED.PART-151449          DIF      FINISHED.PART-151448          STRAIGHT.INNER.DIAMETERS-3-PRIMITIVE
(FINISHED.PART-151450)
FINISHED.PART-151448          DIF      FINISHED.PART-151447          STRAIGHT.INNER.DIAMETERS-2-PRIMITIVE
(FINISHED.PART-151449)
FINISHED.PART-151447          DIF      FINISHED.PART-151446          STRAIGHT.INNER.DIAMETERS-1-PRIMITIVE
(FINISHED.PART-151448)
FINISHED.PART-151446          UN       FINISHED.PART-151445          STRAIGHT.OUTER.DIAMETERS-3-PRIMITIVE
(FINISHED.PART-151447)
FINISHED.PART-151445          UN       EXTERNAL.COMPOUND.FEATURES-151403      STRAIGHT.OUTER.DIAMETERS-2-PRIMITIVE
        (FINISHED.PART-151446)
EXTERNAL.COMPOUND.FEATURES-151403      UN      EXTERNAL.COMPOUND.FEATURES-151402      STRAIGHT.OUTER.DIAMETERS-5
-primitive          (FINISHED.PART-151445)
EXTERNAL.COMPOUND.FEATURES-151402      UN      STRAIGHT.OUTER.DIAMETERS-4-primitive   EXTERNAL.GROOVES-1-primit
ive        (EXTERNAL.COMPOUND.FEATURES-151403)
STRAIGHT.OUTER.DIAMETERS-5-primitive   CYL      NIL       NIL       (EXTERNAL.COMPOUND.FEATURES-151403)
EXTERNAL.GROOVES-1-primitive   CYL      NIL       NIL       (EXTERNAL.COMPOUND.FEATURES-151402)
STRAIGHT.OUTER.DIAMETERS-4-primitive   CYL      NIL       NIL       (EXTERNAL.COMPOUND.FEATURES-151402)
STRAIGHT.INNER.DIAMETERS-4-PRIMITIVE   CYL      NIL       NIL       (FINISHED.PART-151450)
STRAIGHT.INNER.DIAMETERS-3-PRIMITIVE   CYL      NIL       NIL       (FINISHED.PART-151449)
STRAIGHT.INNER.DIAMETERS-2-PRIMITIVE   CYL      NIL       NIL       (FINISHED.PART-151448)
STRAIGHT.INNER.DIAMETERS-1-PRIMITIVE   CYL      NIL       NIL       (FINISHED.PART-151447)
STRAIGHT.OUTER.DIAMETERS-3-PRIMITIVE   CYL      NIL       NIL       (FINISHED.PART-151446)
STRAIGHT.OUTER.DIAMETERS-2-PRIMITIVE   CYL      NIL       NIL       (FINISHED.PART-151445)

Command:
```

Figure 4.9: CSG Representation of Part

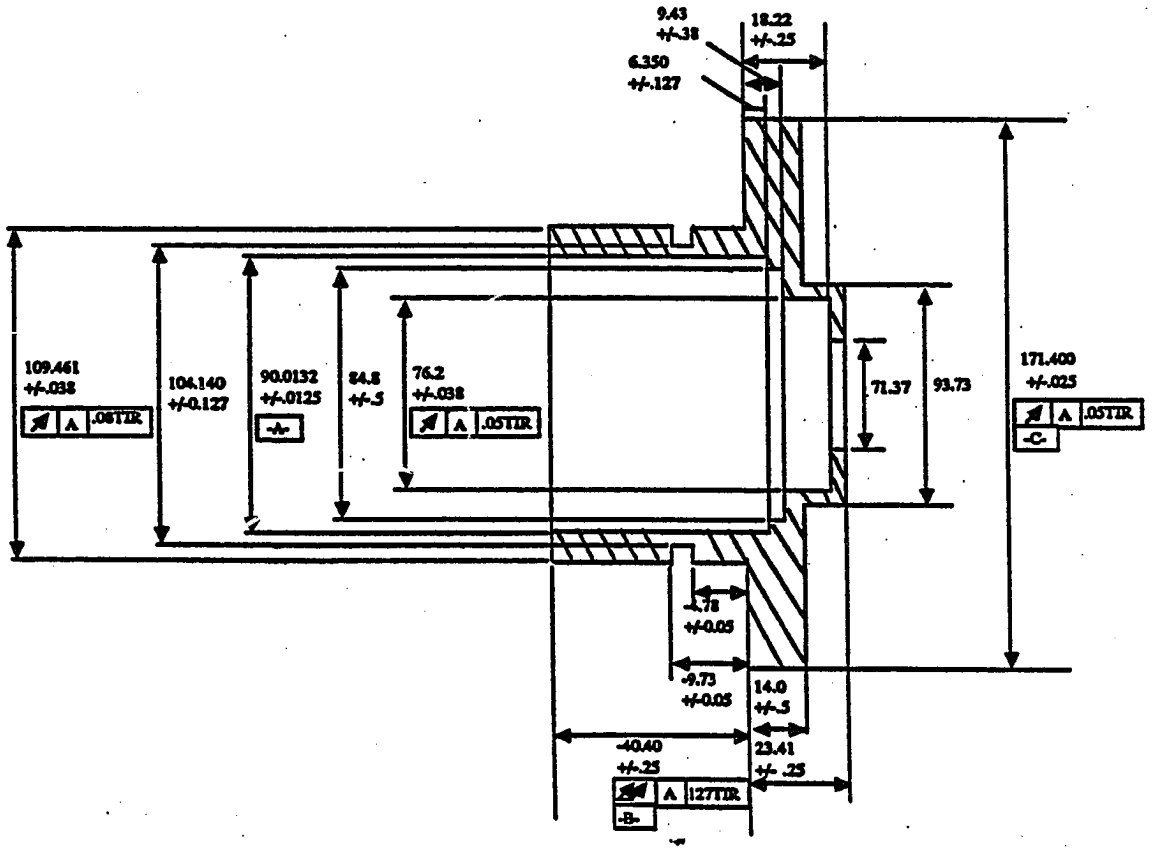

Figure 4.10: Part Design

the values of the feature's parameters and are derived by establishing suitable constraint relationships between the "geometric-model parameters" and "feature parameters." These relationships are established when the intra-feature adjacency relationships are created for a new feature instance. The rigid motion and the primitive solid corresponding to a feature generate the CSG node for that feature. *Generate.csg.node* is the method defined for all features that generates the CSG node.

4.5 Process Performance Model

While describing the procedure to list and classify machining-related concerns in section 3.3 it was indicated that this procedure does not indicate how to identify whether these concerns have arisen for a particular product design. It was mentioned that the task of determining whether these concerns have arisen is the main responsibility of the design environment. The main intent of the research described in this section is to show that in certain cases it is more appropriate to use the process performance model to provide feedback to designers about machining-related concerns. This is especially true when no other kind of knowledge is available or the knowledge available is sparse and highly qualitative in nature. The process performance model enables one to provide the designer a more quantitative description for some of the machining-related concerns.

A process performance model consists of computer-based empirical and mechanistic models that predict the outcome of a particular manufacturing operation [90,91]. Mechanistic models simulate a particular cutting condition on a computer and predict cutting forces, displacement, and vibrations as a function of cutting conditions, cut geometry, and cutting mechanism. The main purpose of the empirical and mechanistic models is to enable one to predict the outcome of a particular operation before the actual cut is made.

The outcome of an operation is usually characterized in terms of performance parameters such as surface error, surface roughness,[4.6] and tool-life. In practice, however, these models are rather difficult to use. One of the primary reasons for this difficulty is the evaluative nature of these models. These models are capable of simulating a particular operation under a specific set of input conditions but are incapable of indicating the changes in the input conditions that will improve the operation. These models can be made more generative in nature by using them in combination with optimization systems. In addition to being able to use the generative systems to determine the cutting conditions for a particular operation, an important advantage of such systems from a Simultaneous Engineering standpoint is that these systems are also useful for providing feedback to designers about certain types of machining-related concerns. For example, the following machining-related concerns from the list of concerns presented in Chapter 3 can be dealt with more suitably by using appropriately developed generative process performance models:

- If interrupted cuts have to be made, then there is a lower limit on the size tolerance that can be achieved.
- Certain size tolerance specifications severely restrict the range of "feedrates" and "cutting speeds" that can be used.
- Certain surface texture specifications severely restrict the range of "feedrates" that can be used.
- Certain surface texture specifications require an extra finish pass to achieve the required surface finish.

In this research generative process performance models have been developed in a partic-

[4.6] Surface Texture.

ular domain and the approach undetaken is described in detail in the remainder of this section[92]. Chapter 6 describes an implementation of this approach that shows how the product designer can be provided feedback about the second type of machining-related concern listed above.

The main purpose of the proposed approach is to establish the values of the cutting parameters for each pass of a particular operation and determine the effect on manufacturing related performance parameters of product design specifications.[4.7] The static mechanistic models used in this research have been developed separately and are described in [93]. Two kinds of inputs are required to use the proposed approach: primary and secondary inputs. The primary inputs are the initial and final workpiece geometry and material, the cutting tool geometry and material, and the aspiration levels for various performance parameters. The secondary inputs are required to use the mechanistic models. Specific cutting pressure (K_T), friction pressure (K_R), influence coefficients, and dominant frequencies and corresponding stiffnesses are representative examples [28,93].

The initial and final workpiece geometry can be obtained from the form "feature parameters" of the feature-based model of a part. The cutting tool geometry and material can be obtained from the "feature parameters" of cutting tool instances that are part of the feature-based model of a facility. Aspiration levels for some of the performance parameters are specified by the precision feature attributes of a part while aspiration levels for other parameters are obtained while developing the process design. The secondary inputs depend on the workpiece geometry and material and cutting tool geometry and material [29,93]. The values for the secondary inputs can be determined after the values for primary inputs are available.

The empirical and mechanistic models developed for predicting surface roughness, sur-

[4.7] Thereby providing feedback to designers about any machining-related concerns that arise.

face error, tool-life, and material removal rate are used to make suitable trade-offs between the aspiration levels set for these parameters. Multiple Criteria Decision-Making (MCDM) techniques [94] provide a number of suitable ways to make such trade-offs and can be used to model the activities involved. The Sequential Multiple Objective Problem Solving technique (SEMOPS) [94] has been used in this research.

Let $\mathbf{x}$, $\mathbf{f(x)}$ and $\mathbf{AL}$ be defined as following:

- $\mathbf{x} = (x_1, x_2, x_3)$ such that its components represent the three cutting parameters: cutting speed, feedrate and depth of cut
- $\mathbf{f(x)} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_T(\mathbf{x}))$ such that its components represent the T multiple objective functions (performance parameters)
- $\mathbf{AL} = (AL_1, AL_2, \ldots, AL_T)$ such that its components represent the aspiration levels corresponding to the T multiple objective functions

For example, if the multiple objectives are surface roughness, surface error, and material removal rate, then the aspiration levels would be the largest values that can be tolerated for surface roughness and surface error and the smallest value that can be tolerated for material removal rate.

Let $\mathbf{d(x)} = (d_1(\mathbf{x}), d_2(\mathbf{x}), \ldots, d_T(\mathbf{x}))$ be a dimensionless indicator function corresponding to $\mathbf{f(x)}$. The relationship between $d_i(\mathbf{x})$ and $f_i(\mathbf{x})$ is determined based on the relationship (ie. $\leq$ *or* $\geq$) between $f_i(\mathbf{x})$ and AL_i. Values of $d_i(\mathbf{x})$ between 0 and 1 always indicate that the aspiration level for that particular objective function has been met. For example, if f_1 represents the objective surface error and AL_1 is the corresponding aspiration level for this objective, then the relationship between them is $f_1 \leq AL_1$ and the value of d_1 corresponding to f_1 can be established as $d_1 = f_1/AL_1$.

The SEMOPS procedure is iterative and each iteration involves solving a principal problem and, at most, T auxiliary problems. For the first iteration, the principal problem and the T auxiliary problems will be as follows:

PRINCIPAL PROBLEM:

Minimize $S_1 = \sum_{t=1}^{T} d_t$

Subject To: $\mathbf{x} \varepsilon X$

AUXILIARY PROBLEMS:

Minimise $S_{1l} = \sum_{t=1}^{T} d_t \quad t \neq l$

Subject To:

$\mathbf{x} \varepsilon X$

$f_l(\mathbf{x}) \geq AL_l$ (or, $f_l(\mathbf{x}) \leq AL_l$)

For l = 1,2,3, . . . ,T and where X defines the feasible region for x.

Each principal and auxiliary problem for a particular pass is a non-linear optimization problem. At the end of each iteration, the resulting vector x and objectives f(x) from the principal problem and the set of auxiliary problems are analyzed. If the results of the analysis are satisfactory (i.e. all aspiration levels have been met), then the iterative process is terminated. If not, the iterations are continued after making suitable changes such as altering the aspiration levels for some objectives, removing certain objectives from further consideration and including them as constraints, or changing the limits on some of the constraints to the problem. The exact nature of the change depends on the results obtained for a particular iteration. The main advantage of using the SEMOPS procedure is that it can be used to solve nonlinear problems and it allows the use of "background information" (about the product and process design task) to suitably adjust or change the

desired aspiration levels for all the objectives at the end of each iteration. Chapter 6 presents a detailed example showing the use of this approach to provide feedback to designers about a certain type of machining-related concern.

4.6 Summary

The Product and Manufacturing Facility Models of the design environment have been described in detail. It was shown that their basic requirements can be satisfied by a combination of Feature-based, Geometric, and Process Performance models. *Features* play an important role in developing the underlying problem-domain models, and it was shown that Frame- and Constraint-Based Systems can be used to develop the feature-based model. Various product and facility feature classes, feature parameter categories, and feature operations were described in detail. The geometric model was based on Constructive Solid Geometry and Boundary Representation schemes, and one way to generate this model from the feature-based model was described. The process performance model consisted of empirical and mechanistic models for predicting the outcome of an operation ahead of the cut. It was indicated that a "generative" rather than "evaluative" process performance model would be useful for providing feedback to designers about machining-related concerns. One way of combining this model with optimization systems to make it "generative" was presented.

Chapter 5

MANUFACTURABILITY ADVISOR

5.1 Introduction

The Manufacturability Advisor system is the reasoning subsystem of the computer-based design environment. The main task of the Manufacturability Advisor system is to enable concurrent reasoning about the product and process design tasks and provide feedback to designers about any machining-related concerns. Due to the two stage approach to the Simultaneous Engineering concept, only the process planning aspect of the process design task has to be considered for developing the Manufacturability Advisor system.

The basic requirements that need to be satisfied by the Manufacturability Advisor system is first presented. Then a brief description of the proposed structure for this system is made, and various facets of the Manufacturability Advisor system are described in detail.

5.2 Multiple Cooperative Knowledge Sources

The basic requirements to be satisfied by the Manufacturability Advisor system depend mainly on the method by which concurrency is achieved between the product design and

process planning tasks. Product design and process planning normally involve two kinds of activities.[5.1] The first type of activity can be classified as a *planning* activity. The primary result of this activity is a plan indicating how the actual product design or process plan description should be generated. The second type of activity can be classified as a *plan execution* activity. This activity is responsible for constructing the product design and process plan descriptions, and its execution depends upon the plan developed by the first type of activity.

To illustrate the distinction between the two kinds of activities described above, consider the process planning task shown schematically in Fig. 5.1. This schematic diagram was established after extensively studying the process planning task carried out by several process planners [92,95].

One type of process planning activity pertains to the determination of:

- the raw material shape and size,
- the machines to process the part,
- the surfaces and fixtures to hold the part on the machines,
- the operations performed on the part,
- the cutting tools to perform the operations,
- the number of passes and cutting parameters for each operation performed on the part, and
- the sequence in which the operations are performed on the part.

[5.1] The description in this section assumes that a task is made up of several activities.

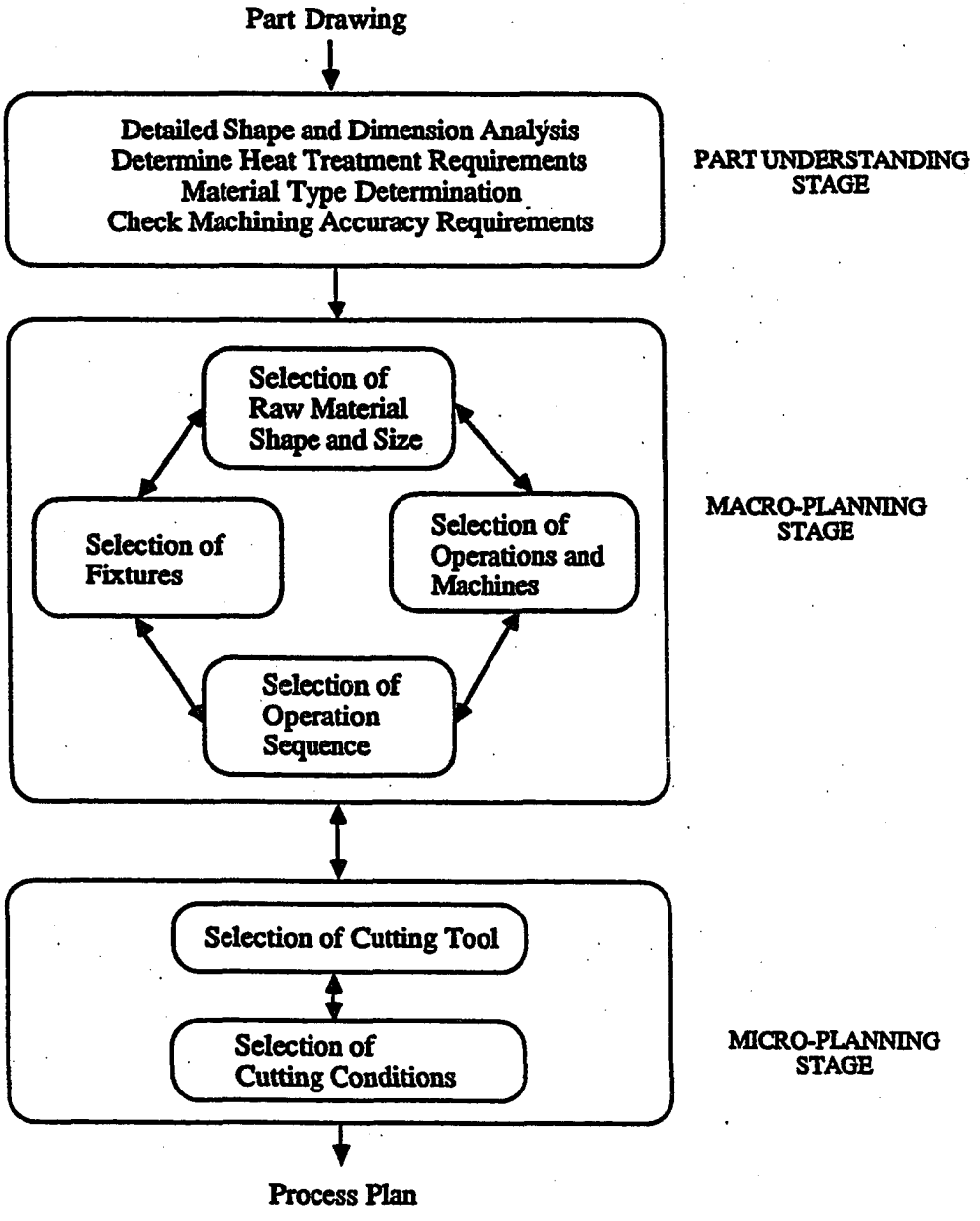

Figure 5.1: Schematic Diagram of Process Planning Task

These activities result in the development of the process plan and are representative examples of *plan execution* activities.

Another type of activity deals with the iterative nature of the process planning task. This is shown in Fig. 5.1 by the bi-directional arrows between the various *plan execution* activities identified above. Decisions involving the determination of *plan execution* activities that will be performed and the time and sequence in which the *plan execution* activities will be executed varies with the complexity of the part and the characteristics of the manufacturing facility. Process planners use "strategic knowledge" gained through years of experience to make such decisions. The end result of such an activity is a plan indicating how the process plan should be generated. These activities are representative examples of *planning* activities.

The complex nature of the product design and process planning domains dictates the formation of two distinct categories of activities in these domains. The *planning* activities enable the *plan execution* activities to proceed in an orderly and guided fashion, thereby preventing a random search of possible product designs and/or process plans. The distinction between the two categories of activities described above is also critical for the fulfillment of the Simultaneous Engineering concept. Concurrency can be achieved under the Simultaneous Engineering concept only by "planning" for concurrency between the product design and process plan execution activities. In order to realize this concept and decrease the number of iterations between product design and process planning, it is important to explicitly reason about the interaction between the *plan execution* activities in these two domains and plan on how and when these activities will take place. Planning for concurrency should be dynamic rather than static and should depend on the partial results generated by the product design and process plan execution activities.

If the product design and process plan execution activities are viewed as *domain-level* activities, then reasoning and planning for the interactions between these domain-level

activities can be viewed as a *control-level* activity. The main requirement for developing the Manufacturability Advisor system is the separation of control- and domain-level knowledge and explicit representation of control knowledge.

The Multiple Cooperative Knowledge Sources (MCKS) paradigm has been used to meet the above mentioned requirements of the Manufacturability Advisor system. The MCKS paradigm has been derived from the blackboard architecture [74-76] for solving complex problems. Under this paradigm, the task of simultaneous product and process design is modeled as a cooperative effort between a hybrid collection of knowledge sources. The proposed MCKS system consists of three main components:

1. A globally accessible database called the Blackboard,

2. A set of Domain Knowledge Sources,

3. A central control component called the Manager.

Figure 5.2 is a schematic diagram of the design environment showing the proposed MCKS structure for the Manufacturability Advisor system. The user[5.2] is the domain knowledge source responsible for the product refinement activity. At various stages in the product design process, the user interacts with several computer-based process refinement domain knowledge sources. Each important task of the process refinement activity is associated with a computer-based knowledge source, e.g., facility selection, fixture selection, machine selection, operation selection, operation sequence selection, cutting tool selection, and cutting parameters selection. The manager is responsible for controlling the concurrency between the product and process refinement knowledge sources. The manager

[5.2] A product designer.

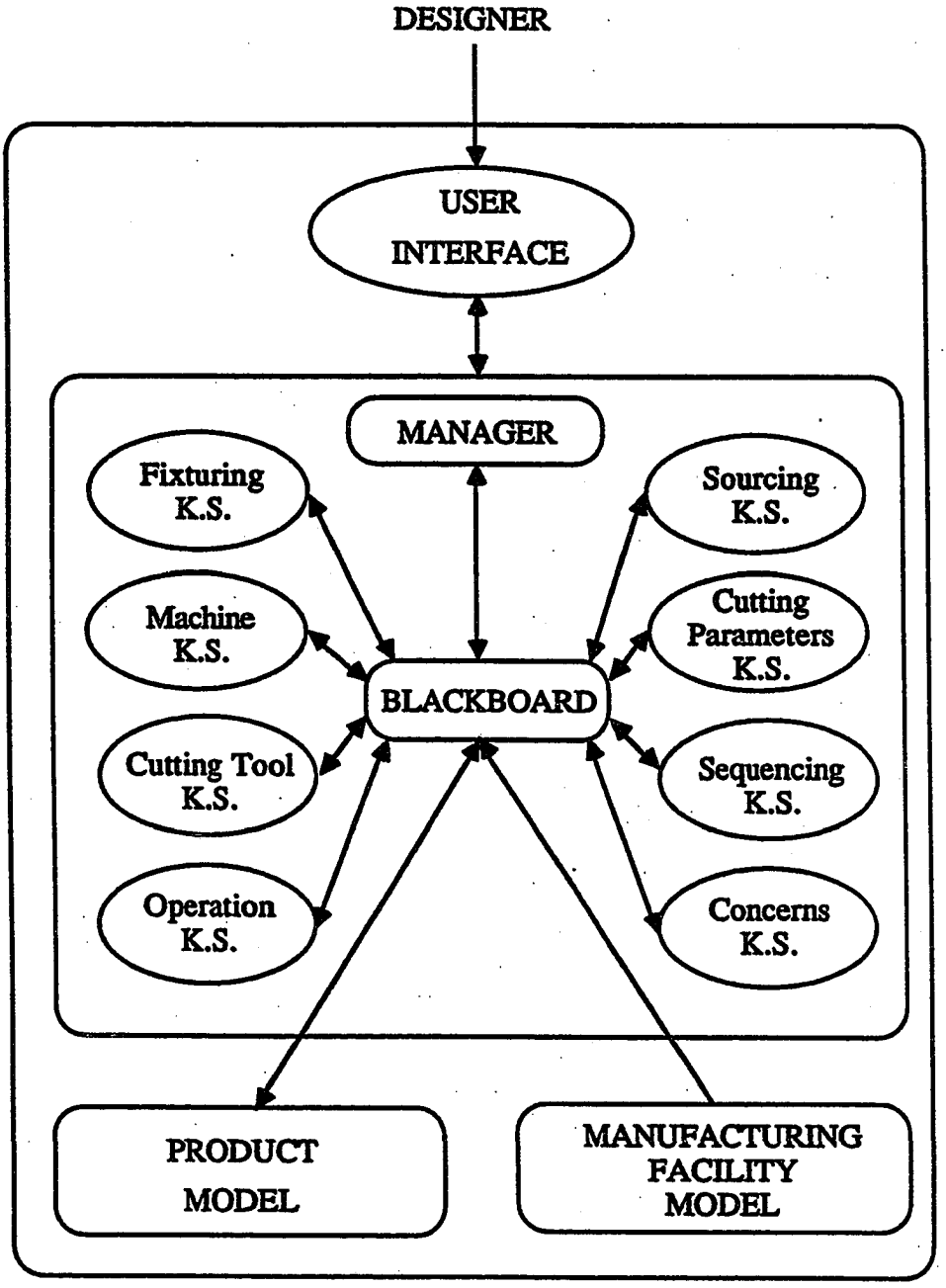

Figure 5.2: Schematic Diagram of Design Environment

explicitly reasons about the kind of concurrency that can take place between the product design and process plan execution activities.

5.3 Blackboard

The blackboard is used to represent the intermediate and final results of concurrent development of the product design and the process plan. The blackboard is organized into linear abstraction levels, and items placed on the blackboard at each level are called *entries*. Entries represent elements of the solution being developed for the product design and process planning tasks and are usually complex structured objects. An entry may be linked to other entries on the same, higher, or lower abstraction levels. Such linked entries represent potential solutions to portions of the problem being solved. The blackboard is the only means of communication between the product design and process planning knowledge sources. These knowledge sources communicate by adding entries to the blackboard or by modifying entries already on the blackboard.

Frame-based systems have been used to implement the blackboard. The inheritance hierarchy underlying the frame-based system used in developing the blackboard and the blackboard entries is shown in Fig. 5.3. There are basically two different kinds of blackboards: the *Control.Blackboard* and the *Domain.Blackboard*. The *Control.Blackboard* is used by the manager to explicitly develop the control plan for concurrency. The *Domain.Blackboard* is used to represent the product design and the manufacturing facility. This blackboard is also used by the computer-based knowledge sources to develop (if necessary) partial process plans for a particular part and facility and to identify any machining-related concerns.

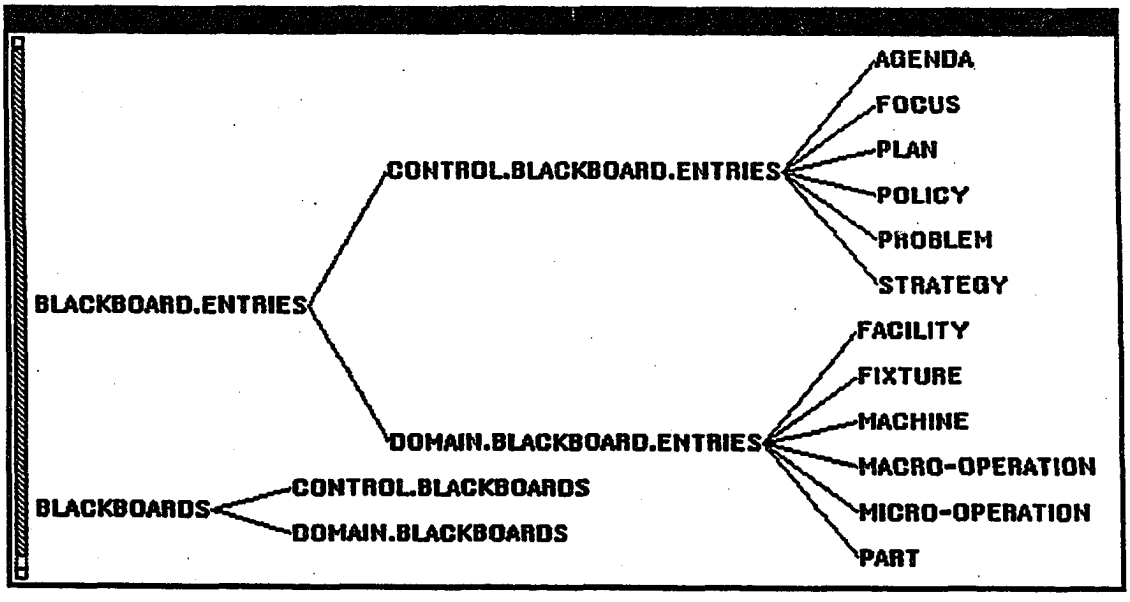

Figure 5.3: Blackboard and Blackboard Entries Inheritance Hierarchy

5.3.1 Control Blackboard

As shown in Fig. 5.4, the *Control.Blackboard* consists of five abstraction levels representing different categories of control decisions. Entries at the Problem, Strategy, Focus, and Policy levels describe desirable actions, and entries at the Agenda level represent feasible actions. Together, these entries specify the control heuristics that are operative during a particular problem-solving interval. In general, the heuristics describing desirable actions are used in evaluating pending Triggered Knowledge Source Instances[5.3] (TKSI) at the Agenda level to schedule the next TKSI for execution. *Control.Blackboard.Entries* (see Fig. 5.3) is the general class of structured objects that represents the various entries made on the control blackboard. The following sections describe in detail relevant attributes for each type of control decision entry.

[5.3] See Section 5.4.

CONTROL BLACKBOARD		DOMAIN BLACKBOARD	
PROBLEM:	The problem to be solved by the system	PART:	Product Model
STRATEGY:	General sequential plan for solving the problem	FACILITY:	Manufacturing Facility Model
FOCUS:	Temporary problem-solving goal	MACHINE:	Machine-level Process Plan
POLICY:	Scheduling criteria	FIXTURE:	Fixture-level Process Plan
AGENDA:	Set of pending triggered knowledge source instances	MACRO-OPERATION:	Macro-operation level Process Plan
		MICRO-OPERATION:	Micro-operation level Process Plan

Figure 5.4: Blackboard Abstraction Levels

Problem

A single decision entry at this level represents the problem that guides an entire problem-solving process. Creating a Problem decision entry on the blackboard initiates problem-solving by triggering and passing control to the manager. The *description* attribute of a Problem (see Fig. 5.5) identifies the type of problem to be solved, for example "modify part." A Problem's *termination.criteria* characterizes an acceptable solution, for example "part has been successfully modified." Changing the Problem's *status* to 'inoperative' terminates problem-solving. A Problem is *implemented.by* adopting one or more Strategies.

```
Entry:  PROBLEM

  Superclasses:        CONTROL.BLACKBOARD.ENTRIES
  Member of:           CLASSES

  Member slot:         ACCOMPLISHED.BY
  Valueclass:          CONTROL.BLACKBOARD.ENTRIES
  Values:              Unknown

  Member slot:         CREATE.NEW.INSTANCE
  Valueclass:          METHOD
  Values:              CREATE.NEW.INSTANCE

  Member slot:         DESCRIPTION
  Valueclass:          (UNION WORD LIST)
  Values:              Unknown

  Member slot:         IMPLEMENTED.BY
  Valueclass:          (LIST.OF CONTROL.BLACKBOARD.ENTRIES)
  Values:              Unknown

  Member slot:         IMPLEMENTS
  Valueclass:          CONTROL.BLACKBOARD.ENTRIES
  Values:              Unknown

  Member slot:         STATUS
  Valueclass:          (ONE.OF OPERATIVE INOPERATIVE)
  Values:              Unknown

  Member slot:         TERMINATION.CRITERIA
  Valueclass:          (LIST.OF LIST)
  Values:              Unknown
```

Figure 5.5: Parametric Description of Problem Level Entry

Strategy

Strategy decision entries establish general sequential plans to solve portions of a Problem. Strategy entries are normally created fairly early so that they can guide the remainder of the problem-solving process. However, there is no restriction on creation of entries later in the problem-solving process in order to refine or alter strategy decisions made earlier. A Strategy entry's *description* attribute (see Fig. 5.6) identifies the type of strategy, for example "new part creation strategy." A Strategy's *termination.criteria* characterizes the desired end result of applying the strategy, for example "new part has been successfully created." A Strategy is *implemented.by* other Strategies or Focuses and, in turn, *implements* Problems or other Strategies. A Strategy entry affects scheduling decisions only indirectly through the Focus entries that implement it. Other relevant attributes of Strategy objects are described in Section 5.4.

Focus

Focus decision entries establish short range problem-solving objectives that are explicitly temporary and operate during restricted problem-solving intervals. Often, a sequence of Focus entries implements a previously created Strategy entry. Focus decision entries can also be created independent of one another and of prior Strategy entries. The *description* attribute of a Focus entry (see Fig. 5.7) identifies the type of focus: for example "create external features." A Focus entry's *termination.criteria* specify the conditions that makes the Focus no longer useful. At any time it is possible for several complementary or competing focus entries to be operative simultaneously. A Focus entry is *implemented.by* one or more Policies and, in turn, *implements* a particular Strategy or Problem. Focus entries affect scheduling decisions through the Policies that implement it.

```
Entry:  STRATEGY

  Superclasses:        CONTROL.BLACKBOARD.ENTRIES
  Member of:           CLASSES

  Member slot:         ACTIVE.PRESCRIPTION
  Valueclass:          (UNION WORD LIST)
  Values:              Unknown

  Member slot:         CURRENT.PRESCRIPTION
  Valueclass:          (UNION WORD LIST)
  Values:              Unknown

  Member slot:         DESCRIPTION
  Valueclass:          (UNION WORD LIST)
  Values:              Unknown

  Member slot:         IMPLEMENTED.BY
  Valueclass:          (LIST.OF CONTROL.BLACKBOARD.ENTRIES)
  Values:              Unknown

  Member slot:         IMPLEMENTS
  Valueclass:          CONTROL.BLACKBOARD.ENTRIES
  Values:              Unknown

   Member slot:         STATUS
  Valueclass:          (ONE.OF OPERATIVE INOPERATIVE)
  Values:              Unknown

  Member slot:         STRATEGY.GENERATOR
  Valueclass:          (LIST.OF (UNION WORD LIST))
  Values:              Unknown

  Member slot:         TERMINATION.CRITERIA
  Valueclass:          (LIST.OF LIST)
  Values:              Unknown
```

Figure 5.6: Parametric Description of Strategy Level Entry

```
Entry:  FOCUS

  Superclasses:        CONTROL.BLACKBOARD.ENTRIES
  Member of:           CLASSES

  Member slot:         ACCOMPLISHED.BY
  Valueclass:          CONTROL.BLACKBOARD.ENTRIES
  Values:              Unknown

  Member slot:         DESCRIPTION
  Valueclass:          (UNION WORD LIST)
  Values:              Unknown

  Member slot:         IMPLEMENTED.BY
  Valueclass:          (LIST.OF CONTROL.BLACKBOARD.ENTRIES)
  Values:              Unknown

  Member slot:         IMPLEMENTS
  Valueclass:          CONTROL.BLACKBOARD.ENTRIES
  Values:              Unknown

  Member slot:         RATING.CRITERIA
  Values:              Unknown

  Member slot:         RATING.FUNCTION
  Valueclass:          METHOD
  Values:              FOCUS-POLICY.RATING.FUNCTION

  Member slot:         STATUS
  Valueclass:          (ONE.OF OPERATIVE INOPERATIVE)
  Values:              Unknown

  Member slot:         TERMINATION.CRITERIA
  Valueclass:          (LIST.OF LIST)
  Values:              Unknown
```

Figure 5.7: Parametric Description of Focus Level Entry

Policy

Policy decision entries contain local and global scheduling criteria that will be used to favour TKSI's on the Agenda with particular attributes and values. Depending on how they are created, some Policy decisions can remain operative from the time they are created until the end of the problem-solving process. Ordinarily, multiple Policy decisions operate simultaneously. A Policy entry's *rating.criteria* attribute (see Fig. 5.8) indicate a predicate or function on one or more TKSI attribute-value pairs. For example, the criterion 'Action Blackboard = Control Blackboard' prescribes execution of TKSI's whose actions occur on the control blackboard. The *weight* of a Policy is the expected value[5.4] of its prescribed actions and is a value between 0 and 1. For example, values of 0.8, 0.5, and 0.2 for the weight of a particular policy indicate a high, medium, and low priority for the actions prescribed by the Policy. The *rating.function* of a Policy is a method to determine the actual rating of a particular TKSI against the policy. The following two conditions need to be satisfied before a Policy on the blackboard can affect scheduling decisions:

1. There are TKSI's on the Agenda with the attributes and values prescribed by its *rating.criteria* and

2. The current integration and scheduling rules[5.5] incorporate the Policy.

A Policy *implements* a Problem or one or more Focuses and Strategies. Policy decision entries do not explicitly state any termination criteria and remain operative as long as the entries that they *implement* at the higher abstraction levels remain operative.

[5.4] Importance or value.
[5.5] See Section 5.4.

```
Entry:  POLICY

  Superclasses:        CONTROL.BLACKBOARD.ENTRIES
  Member of:           CLASSES

  Member slot:         DESCRIPTION
  Valueclass:          (UNION WORD LIST)
  Values:              Unknown

  Member slot:         IMPLEMENTS
  Valueclass:          CONTROL.BLACKBOARD.ENTRIES
  Values:              Unknown

  Member slot:         RATING.CRITERIA
  Valueclass:          (LIST.OF (UNION WORD LIST))
  Values:              Unknown

  Member slot:         RATING.FUNCTION
  Valueclass:          METHOD
  Values:              FOCUS-POLICY.RATING.FUNCTION

  Member slot:         STATUS
  Valueclass:          (ONE.OF OPERATIVE INOPERATIVE)
  Values:              Unknown

  Member slot:         WEIGHT
  Valueclass:          NUMBER
  Values:              Unknown
```

Figure 5.8: Parametric Description of Policy Level Entry

Agenda

The Agenda identifies all pending TKSI's on each problem-solving cycle. The highest rated TKSI based on the currently active Focuses and Policies is the one chosen for execution during the next problem-solving cycle.

5.3.2 Domain Blackboard

The domain blackboard represents the product model, manufacturing facility model, and the process plan for a particular part design and facility. The linear abstraction hierarchy of the domain blackboad consists of six levels, namely Part, Facility, Machine, Fixture, Macro-Operation, and Micro-Operation (see Fig. 5.4). The general class of structured objects that all entries on the domain blackboard belong to is named *Domain.Blackboard.Entries* (see Fig. 5.3). Entries at the Part and Facility level (see Fig. 5.9 and Fig. 5.10) represent the product and manufacturing facility models, respectively. Entries on the remaining levels represent a different abstract view of the process plan such as decisions about what machines to use, where to fixture the part, what operations to perform, and what cutting tools to use. Appendix C lists all the attributes of domain blackboard entries. The following sections describe in detail only relevant attributes of such entries.

Machine

Decision entries at this level represent a machine-level abstraction of the process plan. A subset of entries at this level defines the machines to be used in manufacturing a part and the sequence in which the part will be processed on these machines. At any time in the domain problem-solving process, several complementary or competing Machine entries may exist simultaneously. The *description* attribute (see Fig. 5.11) of a Machine indicates the

```
Entry:  PART

  Superclasses:        DOMAIN.BLACKBOARD.ENTRIES
  Member of:           CLASSES

  Member slot:         COMMODITY.TYPE
  Values:              Unknown

   Member slot:         CURRENT.FEATURE.SEQUENCE
  Valueclass:          (ONE.OF EXTERNAL.FEATURES INTERNAL.FEATURES)
  Values:              Unknown

  Member slot:         FACILITY
  Valueclass:          FACILITY
  Values:              Unknown

  Member slot:         FINISHED.PART
  Valueclass:          FINISHED.PART
  Values:              Unknown

  Member slot:         LARGEST.DIAMETER
  Values:              Unknown

  Member slot:         LOT.SIZE
  Values:              Unknown

  Member slot:         MATERIAL
  Valueclass:          MATERIAL.FEATURES
  Values:              Unknown

  Member slot:         OVERALL.LENGTH
  Values:              Unknown

  Member slot:         PART.NUMBER
  Values:              Unknown
```

Figure 5.9: Parametric Description of Part Level Entry

```
Entry:  FACILITY

  Superclasses:        DOMAIN.BLACKBOARD.ENTRIES
  Member of:           CLASSES

  Member slot:         DESCRIPTION
  Valueclass:          CELL.FEATURES
  Values:              Unknown

  Member slot:         FACILITY.CLASS
  Valueclass:          CELL.FEATURES
  Values:              Unknown

  Member slot:         MACHINE
  Valueclass:          MACHINE
  Values:              Unknown

  Member slot:         PART
  Valueclass:          PART
  Values:              Unknown
```

Figure 5.10: Parametric Description of Facility Level Entry

```
Entry:  MACHINE

  Superclasses:        DOMAIN.BLACKBOARD.ENTRIES
  Member of:           CLASSES

  Member slot:         DESCRIPTION
  Valueclass:          MACHINE.FEATURES
  Values:              Unknown

  Member slot:         FACILITY
  Valueclass:          FACILITY
  Values:              Unknown

  Member slot:         FIXTURE
  Valueclass:          FIXTURE
  Values:              Unknown

  Member slot:         NEXT.MACHINE
  Valueclass:          MACHINE
  Values:              Unknown

  Member slot:         PREVIOUS.MACHINE
  Valueclass:          MACHINE
  Values:              Unknown
```

Figure 5.11: Parametric Description of Machine Level Entry

type of machine that will be used to process the part, for example a "2-SC Lathe." A Machine's *previous.machine* and *next.machine* indicate the Machines preceding and following the current Machine in the abstract representation of the process plan at this level. A Machine entry's *fixture* indicates the type of fixtures that will be used to hold the part on this machine.

Fixture

Entries at this level primarily indicate the fixtures that will be used to hold the part during the machining process. The *description* attribute of a Fixture entry indicates the type of fixture that will be used to hold the part, for example "speed-grip.1." The *machine* attribute (see Fig. 5.12) indicates the machine on which the fixture is mounted, and the *macro-operation* attribute indicates the macro-operations that will be performed while the part is held in this fixture. The *primary.fixturing.surface* and *secondary.fixturing.surface* of a Fixture entry indicates the primitive features that will be used to hold the part. A Fixture's *previous.fixture* and *next.fixture* attributes are self explanatory. The sequential ordering of the Fixture level entries need not necessarily be identical[5.6] to the ordering of the Machine level entries in the final process plan.

Macro-Operation

Decision entries at this level represent a macro-operation (e.g., turn, drill) level abstraction of the process plan. A subset of entries at this level defines the macro-operations, part features, and machines on which these operations will be performed. Once again it is possible for several complementary or competing macro-operation level decisions to exist simultaneously at any time during the problem-solving process. The *description* of a Macro-Operation entry indicates the type of operation to be performed: for example "turn." A Macro-Operation entry's *feature* attribute (see Fig. 5.13) indicates the product feature that will be generated as a result of this operation, for example a *chamfer*. A Macro-Operation's *previous.macro-operation* and *next.macro-operation* indicate the operations preceding and following the current Macro-Operation. A Macro-Operation entry's *fixture* is the Fixture

[5.6]On a particular machine, a part may be manufactured by being held in different orientations and/or fixtures.

```
Entry:  FIXTURE

  Superclasses:        DOMAIN.BLACKBOARD.ENTRIES
  Member of:           CLASSES

  Member slot:         DESCRIPTION
  Values:              Unknown

  Member slot:         MACHINE
  Valueclass:          MACHINE
  Values:              Unknown

  Member slot:         MACRO-OPERATION
  Valueclass:          MACRO-OPERATION
  Values:              Unknown

  Member slot:         NEXT.FIXTURE
  Values:              Unknown

  Member slot:         PREVIOUS.FIXTURE
  Values:              Unknown

  Member slot:         PRIMARY.FIXTURING.SURFACE
  Values:              Unknown

  Member slot:         SECONDARY.FIXTURING.SURFACE
  Values:              Unknown
```

Figure 5.12: Parametric Description of Fixture Level Entry

that will be used to hold the part while performing this operation, and its *micro-operation* are the micro-operation steps in which the operation will be performed.

Micro-Operation

A subset of entries at this level define the micro-operations to be performed (e.g., rough turn, start drill), the cutting tools to be used, and the cutting parameters for each micro-operation. The *description* of a Micro-Operation entry indicates the type of operation to be performed, for example, "rough turn." *Previous.micro-operation* and *Next.micro-operation* attributes (see Fig. 5.14) are self explanatory. The ordering of Micro-Operation entries need not necessarily be the same as the ordering of the Macro-Operation entries in the final process plan. A Micro-Operation entry's *cutting.tool* is the cutting tool assembly that will be used to perform this operation. A Micro-Operation's *cutting.speed*, *feed*, and *depth.of.cut* indicates the cutting parameters that will be used for this operation. The *macro-operation* attribute indicates the macro-operation that is accomplished by a particular Micro-Operation.

5.4 Knowledge Sources

Knowledge sources mainly describe problem-solving or action knowledge and are represented as a collection of Knowledge Source Instances (KSI). The problem-solving knowledge in KSI's is represented as condition-action pairs:[5.7] a description of when they are applicable (conditions) and the relevant actions to be performed. The condition part monitors the blackboard for additions or modifications while the action part makes further additions or modifications to the blackboard.

[5.7] Synonymous with rules.

```
Entry:  MACRO-OPERATION

  Superclasses:        DOMAIN.BLACKBOARD.ENTRIES
  Member of:           CLASSES

  Member slot:         DESCRIPTION
  Values:              Unknown

  Member slot:         FEATURE
  Valueclass:          PRIMITIVE.FEATURES
  Values:              Unknown

  Member slot:         FIXTURE
  Valueclass:          MACHINE
  Values:              Unknown

  Member slot:         MICRO-OPERATION
  Valueclass:          MICRO-OPERATION
  Values:              Unknown

  Member slot:         NEXT.MACRO-OPERATION
  Values:              Unknown

  Member slot:         PREVIOUS.MACRO-OPERATION
  Values:              Unknown
```

Figure 5.13: Parametric Description of Macro-Operation Level Entry

```
Entry:  MICRO-OPERATION

  Superclasses:        DOMAIN.BLACKBOARD.ENTRIES
  Member of:           CLASSES

  Member slot:         CUTTING.SPEED
  Valueclass:          NUMBER
  Values:              Unknown

  Member slot:         CUTTING.TOOL.ASSEMBLY
  Valueclass:          CUTTING.TOOL.ASSEMBLIES
  Values:              Unknown

  Member slot:         DEPTH.OF.CUT
  Valueclass:          NUMBER
  Values:              Unknown

  Member slot:         DESCRIPTION
  Values:              Unknown

  Member slot:         FEED-RATE
  Valueclass:          NUMBER
  Values:              Unknown

  Member slot:         MACRO-OPERATION
  Valueclass:          MACRO-OPERATION
  Values:              Unknown

  Member slot:         NEXT.MICRO-OPERATION
  Values:              Unknown

  Member slot:         PREVIOUS.MICRO-OPERATION
  Values:              Unknown
```

Figure 5.14: Parametric Description of Micro-Operation Level Entry

Frame-based systems have been used to develop the knowledge sources, and the inheritance hierarchy underlying them is shown in Fig. 5.15. Knowledge sources can be broadly classified into two types: *Manager* and *Domain.Knowledge.Sources.* A listing of the attributes of *Knowledge.Sources* is provided in Appendix C. The attributes *premise* and *conclusion* are used to indicate the conditions and actions of a particular knowledge source instance. The attributes *from.blackboard, to.blackboard, from.level,* and *to.level* indicate the blackboards and levels addressed by the condition and action portions of the knowledge source. The *rating.conditions* of a knowledge source are variable-value pairs, such as "Action.Level = Machine." They are used while making scheduling decisions on the blackboard (as described below). The remaining attributes of *Knowledge.Sources* shown in Appendix C are described in detail in [97].

There are three different kinds of knowledge source instances. In describing them, it is useful to define the term *context* or *world.* A *context* or *world* is the term used to describe an assumption set and is a label for a set of facts. In describing the blackboard structure, it was mentioned that several complementary or competing decisions entries can simultaneously exist on the blackboard at any time during the problem-solving process. *Contexts* are used to distinguish and identify such complementary or competing decisions. *Same World* knowledge source instances are those instances whose condition and action parts are applicable in a single context. All the conditions of such an instance have to be true in a single context and the actions add, change, or delete facts in the same context. *New World* knowledge source instances are those instances that are applicable when their conditions are true in one or more contexts. A new context, obtained by merging all the contexts in which the conditions of the instance are true, is the result of applying the knowledge possessed by this instance. Actions that are specified by the instance are made in the new context (and do not affect the contexts used for merging). *Deduction* knowledge

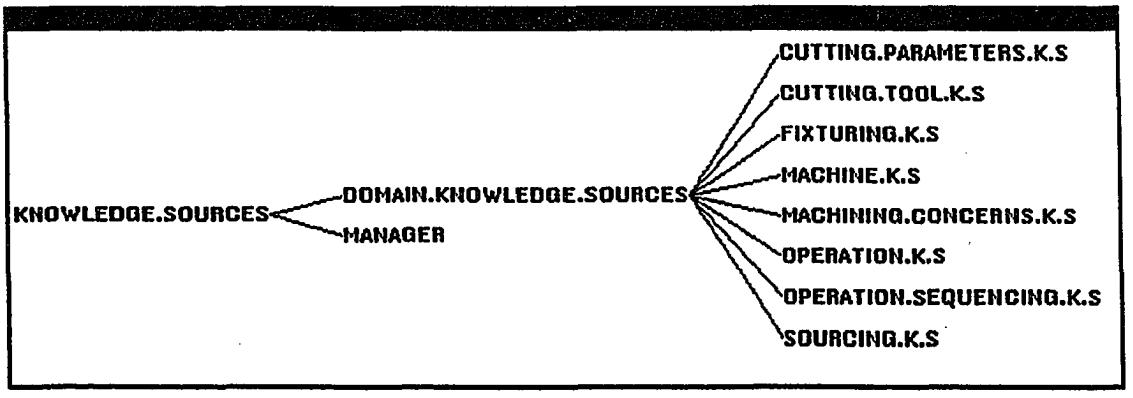

Figure 5.15: Knowledge Sources Inheritance Hierarchy

source instances are used to represent generalized dependencies between facts in one or more worlds. The application of these instances results in the creation of TMS [96] justifications. *Deduction* KSI's do not create new worlds, as compared to *New World* KSI's; they are applicable only in existing worlds.

When a knowledge source instance has its conditions satisfied, it generates a Triggered Knowledge Sources Instance (TKSI). A TKSI is a knowledge source instance with its variables instantiated. At any instance, a particular knowledge source instance can generate more than one TKSI depending on the state of the blackboard.

Problem-solving under the MCKS paradigm is agenda-based and consists of an execution cycle made up of three main parts: Interpretation, Agenda Maintenance, and Scheduling. Interpretation involves taking the TKSI chosen for execution, ensuring that its conditions are still true, and performing its actions. Agenda Maintenance involves determining new TKSI's that are generated and updating the agenda of triggered TKSI's. Scheduling involves choosing the next TKSI for execution from the agenda using explicitly defined scheduling rules. The scheduling rule is normally specified as the value of the attribute

scheduling.function for the policy object *policy-1* on the blackboard. This scheduling function takes into account the explicit scheduling decisions made on the control blackboard while determining the next TKSI for interpretation.

The rest of this chapter briefly describes the nature of the task carried out by the various computer-based knowledge sources. The knowledge source instances used as examples are only representative of the KSI's that define any particular knowledge source. In describing the KSI's only the "rule" portion of the KSI is provided.

5.4.1 Manager

The manager is in charge of controlling the problem-solving behavior of the system. The manager establishes the basic problem-solving strategy that will be adopted by the domain knowledge sources in developing the required solution. The control knowledge possessed by the manager causes certain actions to be preferred when scheduling decisions are made. As mentioned before, under the Simultaneous Engineering concept, the manager is primarily responsible for explicitly developing the plan to ensure concurrency between the product design and process planning tasks. The manager performs this task by developing this plan on the control blackboard. The knowledge possessed by the manager can be further classified into three categories: Domain-Specific Control Knowledge, Generic Control Knowledge, and Domain-Independent Control Knowledge.

One way to set up control knowledge is to explicitly define the knowledge required to generate the entries at various abstraction levels on the *Control.Blackboard*. This category of knowledge is defined as Domain-Specific Control Knowledge. For example, the condition part of *Manager-1* ascertains whether a strategy level entry is to "create a new part," and the facility to manufacture the part has been determined. The action part indicates that a new Focus entry should be made on the control blackboard to provide feedback

about machining-related concerns. *Manager-2* indicates that if a strategy level entry is to determine how to fixture a part using a speed-grip and the primary and secondary fixturing surfaces have been determined, then a new focus entry should be created to determine how these surfaces will be machined.

```
Rule:  MANAGER-1

(IF (THE STRATEGY OF CONTROL.BLACKBOARD IS ?STRATEGY)
    (THE STATUS OF ?STRATEGY IS OPERATIVE)
    (LISP (EQUAL 'CREATE.NEW.PART.STRATEGY (UNIT.NAME ?STRATEGY)))
    (THE PART OF DOMAIN.BLACKBOARD IS ?PART)
    (THE FACILITY OF ?PART IS ?BLACKBOARD.FACILITY)
    (THE DESCRIPTION OF ?BLACKBOARD.FACILITY IS ?FACILITY)
    THEN
    DO
    (THE STATUS OF FEEDBACK.MACHINING.CONCERNS.FOCUS IS OPERATIVE)
    (THE FOCUS OF CONTROL.BLACKBOARD IS
                FEEDBACK.MACHINING.CONCERNS.FOCUS))

Rule:  MANAGER-2

(IF (THE STRATEGY OF CONTROL.BLACKBOARD IS ?STRATEGY)
    (THE STATUS OF ?STRATEGY IS OPERATIVE)
    (LISP (EQUAL 'DETERMINE.PART.FIXTURE.DETAILS (UNIT.NAME ?STRATEGY)))
```

```
(THE FIXTURE OF DOMAIN.BLACKBOARD IS ?FIXTURE.ENTRY)
(THE DESCRIPTION OF ?FIXTURE.ENTRY IS ?FIXTURE)
(?FIXTURE IS IN SPEED.GRIPS)
(THE PRIMARY.FIXTURING.SURFACE OF ?FIXTURE.ENTRY IS ?A.FEATURE)
(THE SECONDARY.FIXTURING.SURFACE OF ?FIXTURE.ENTRY IS ?ANOTHER.FEATURE)
THEN
DO
(THE STATUS OF SPEED.GRIP.SURFACE.OPERATIONS.FOCUS IS OPERATIVE)
(THE FOCUS OF CONTROL.BLACKBOARD IS
                SPEED.GRIP.SURFACE.OPERATIONS.FOCUS))
```

Instead of explicitly specifying control knowledge as described above, another way to specify it is in terms of a skeletal plan. Such a plan provides 'skeletal' descriptions of the strategy, focus, and policy entries that are actually instantiated by a particular control plan. Instances of *Plan* objects, another type of *Control.Blackboard.Entries* as shown in Fig. 5.3, are used to represent various skeletal plans. The *precondition* attribute specified for the various plans indicate when and where a particular skeletal plan is appropriate and can be activated. These preconditions are used to create control knowledge source instances before starting the product development process. The attributes *accomplished.by*, *active.prescription*, *current.prescription*, and *remaining.strategy.generator* of control blackboard entries are used to determine when these entries should be activated, updated, or deactivated.

Knowledge required to instantiate and update relevant skeletal plans and control blackboard entries is an example of Generic Control Knowledge. *Manager-3*, for example, modifies a newly-posted Strategy entry such that its first substrategies or focuses are made

operative. *Manager-4* updates an active strategy entry such that the next step of the strategy plan is activated.

```
Rule:  MANAGER-3

(IF (?STRATEGY IS IN STRATEGY)
    (THE STRATEGY OF CONTROL.BLACKBOARD IS ?STRATEGY)
    (THE STATUS OF ?STRATEGY IS OPERATIVE)
    (LISP (UNITMSG ?STRATEGY 'GET.CURRENT.PRESCRIPTION ?$WORLD$))
    (EQUAL ?CURRENT.PRESCRIPTION
           (LISP (UNITMSG ?STRATEGY 'GET.CURRENT.PRESCRIPTION ?$WORLD$)))
    THEN
    DO
    (CHANGE.TO
      (THE CURRENT.PRESCRIPTION OF ?STRATEGY IS ?CURRENT.PRESCRIPTION))
    (CHANGE.TO
      (THE ACTIVE.PRESCRIPTION OF ?STRATEGY IS ?CURRENT.PRESCRIPTION))
    (LISP (UNITMSG ?STRATEGY
                   'UPDATE.REMAINING.STRATEGY.GENERATOR ?$WORLD$)))


Rule:  MANAGER-4

(IF (?STRATEGY IS IN STRATEGY)
    (THE STATUS OF ?STRATEGY IS OPERATIVE)
```

```
(THE ACTIVE.PRESCRIPTION OF ?STRATEGY IS ?PRESCRIPTION)
(LISP (STRING-EQUAL ?PRESCRIPTION "NONE"))
(THE REMAINING.STRATEGY.GENERATOR OF ?STRATEGY IS
                                    ?REMAINING.STRATEGY.GENERATOR)
(LISP (UNITMSG ?STRATEGY 'GET.CURRENT.PRESCRIPTION ?$WORLD$))
(EQUAL ?CURRENT.PRESCRIPTION
       (LISP (UNITMSG ?STRATEGY 'GET.CURRENT.PRESCRIPTION ?$WORLD$)))
THEN
DO
(CHANGE.TO (THE CURRENT.PRESCRIPTION OF ?STRATEGY IS
                                        ?CURRENT.PRESCRIPTION))
(CHANGE.TO (THE ACTIVE.PRESCRIPTION OF ?STRATEGY IS
           ?CURRENT.PRESCRIPTION))
(LISP (UNITMSG ?STRATEGY 'UPDATE.REMAINING.STRATEGY.GENERATOR
                         ?$WORLD$)))
```

Finally, Domain-Independent Control Knowledge is generated while solving a particular problem, especially in those cases where control knowledge is specified in a skeletal form. Some of the control knowledge source instances belonging to this category are created by the triggering of generic control knowledge. Termination knowledge that is required to make strategies and focuses of the skeletal plan inoperative at the appropriate time is an example of this kind of knowledge. *Manager-5* is the knowledge source instance that is triggered to create such termination knowledge source instances.

Rule: MANAGER-5

```
(IF (?BLACKBOARD.OBJECT IS IN CONTROL.BLACKBOARD.ENTRIES)
    (OR (THE STRATEGY OF CONTROL.BLACKBOARD IS ?BLACKBOARD.OBJECT)
        (THE FOCUS OF CONTROL.BLACKBOARD IS ?BLACKBOARD.OBJECT))
    (THE STATUS OF ?BLACKBOARD.OBJECT IS OPERATIVE)
    THEN
    DO
    (LISP
     (UNITMSG '(MANAGER-6 KNOWLEDGE-SOURCES)
              'CREATE.TERMINATION.KNOWLEDGE.SOURCES
              ?BLACKBOARD.OBJECT
              ?$WORLD$)))
```

5.4.2 Domain Knowledge Sources

Domain knowledge sources are responsible for constructing the actual product design and process plan and for providing feedback to designers about machining-related concerns. Domain knowledge sources perform their activity by creating and modifying decisions on the domain blackboard. As mentioned earlier, the user is the knowledge source responsible for product design activities. Several computer-based knowledge sources are responsible for the process planning activities and for determining if machining-related concerns have arisen.

Sourcing Knowledge Source

The Sourcing[5.8] knowledge source is responsible for decisions related to determining the kind of facilities that can be used to manufacture a particular part. *Sourcing.K.S-1* indicates

[5.8] In industrial parlance, a term used to indicate where a product is manufactured.

that a particular class of facilities (e.g. *Bearing.Cage.Cells*) can be used to process a part if the commodity and material specifications of the part are within facility level capabilities. *Sourcing.K.S-2* indicates that a particular context is incorrect if the largest diameter of a part is greater than the largest diameter that can be manufactured in the facility assigned to it (in that context).

Rule: SOURCING.K.S-1

```
(IF (THE PART OF DOMAIN.BLACKBOARD IS ?PART)
    (THE COMMODITY.TYPE OF ?PART IS ?COMMODITY.TYPE)
    (THE MATERIAL OF ?PART IS ?MATERIAL)
    (ALL ?FACILITY.CLASS ARE CELL.FEATURES)
    (THE COMMODITY.TYPE.LIMITATION OF ALL ?FACILITY.CLASS IS ?COM.LIMIT)
    (LISP (EQUAL ?COMMODITY.TYPE ?COM.LIMIT))
    (THE MATERIAL.LIMITATION OF ALL ?FACILITY.CLASS IS ?MAT.LIMIT)
    (LISP
     (OR (EQUAL (UNIT.NAME ?MAT.LIMIT) (UNIT.NAME ?MATERIAL))
         (UNIT.DESCENDANT.P ?MATERIAL ?MAT.LIMIT 'MEMBER)
         (UNIT.DESCENDANT.P ?MATERIAL ?MAT.LIMIT 'SUBCLASS T)))
    (EQUAL ?FACILITY
           (K.S.EQUAL (FORMAT NIL "~A-~A-~A" ?PART ?COMMODITY.TYPE
                              ?FACILITY.CLASS)
                      (UNITMSG '(FACILITY BLACKBOARD-LIBRARY)
                               'CREATE.NEW.INSTANCE)))
    THEN
```

```
    IN.NEW.WORLD
    (THE FACILITY OF DOMAIN.BLACKBOARD IS ?FACILITY)
    (THE PART OF ?FACILITY IS ?PART)
    (THE FACILITY.CLASS OF ?FACILITY IS ?FACILITY.CLASS)
    (THE FACILITY OF ?PART IS ?FACILITY)
    (LISP (INFORM.ABOUT.FACILITY.CLASS ?PART ?$WORLD$)))

Rule:  SOURCING.K.S-2

(WHILE (THE PART OF DOMAIN.BLACKBOARD IS ?PART)
       (THE FACILITY OF ?PART IS ?PART.FACILITY)
       (THE DESCRIPTION OF ?PART.FACILITY IS ?FACILITY)
       (THE LARGEST.DIAMETER.LIMITATION OF ?FACILITY IS ?LIMIT)
       (THE LARGEST.DIAMETER OF ?PART IS ?DIAMETER)
       (LISP (> ?DIAMETER ?LIMIT))
      BELIEVE FALSE)
```

Machine Knowledge Source

This knowledge source is primarily responsible for making decisions related to the machine(s) and the ordering of machines that will be used to manufacture the part. The main result of decisions made by this knowledge source is the addition of Machine entries on the domain blackboard. The condition part of *Machine.K.S-1* determines whether the facility to process a part is known and whether a machine in the facility can manufacture some of the primitive features of the part. The action part of the knowledge source instance prescribes the creation of a Machine entry and instantiation of relevant Machine attributes.

The creation of this entry does not necessarily imply that this machine will be used in the final process plan. The Machine entry is only a hypothesis and is, therefore, created in a separate world.

Rule: MACHINE.K.S-1

```
(IF (THE PART OF DOMAIN.BLACKBOARD IS ?PART)
    (THE FINISHED.PART OF ?PART IS ?FINISHED.PART)
    (THE FACILITY OF ?PART IS ?FACILITY)
    (THE DESCRIPTION OF ?FACILITY IS ?CELL.FEATURE)
    (THE MACHINE OF ?CELL.FEATURE IS ?MACHINE)
    (LISP (REQUIRE.MACHINE? ?MACHINE ?FINISHED.PART))
    (EQUAL ?MACHINE.LEVEL.ENTRY
           (K.S.EQUAL (FORMAT NIL "~A-~A" ?MACHINE ?CELL.FEATURE)
                      (UNITMSG '(MACHINE BLACKBOARD-LIBRARY)
                               'CREATE.NEW.INSTANCE)))
    THEN
    IN.NEW.AND.WORLD
    (THE FACILITY OF ?MACHINE.LEVEL.ENTRY IS ?FACILITY)
    (THE MACHINE OF ?FACILITY IS ?MACHINE.LEVEL.ENTRY)
    (THE DESCRIPTION OF ?MACHINE.LEVEL.ENTRY IS ?MACHINE)
    (THE MACHINE OF DOMAIN.BLACKBOARD IS ?MACHINE.LEVEL.ENTRY))
```

Machine.K.S-2 indicates: If *?machine1* is a Machine entry with a fixture in the class *chucks* and *?machine2* is another Machine entry with a fixture in the class *speed.grips*, then one possible partial processing order is to process the part on *?machine1* before processing

it on *?machine2.*

```
Rule:  MACHINE.K.S-2

(IF (THE MACHINE OF DOMAIN.BLACKBOARD IS ?MACHINE1)
    (THE FIXTURE OF ?MACHINE1 IS ?FIXTURE.ENTRY1)
    (THE DESCRIPTION OF ?FIXTURE.ENTRY1 IS ?FIXTURE1)
    (?FIXTURE1 IS IN CLASS CHUCKS)
    (THE MACHINE OF DOMAIN.BLACKBOARD IS ?MACHINE2)
    (THE FIXTURE OF ?MACHINE2 IS ?FIXTURE.ENTRY2)
    (THE DESCRIPTION OF ?FIXTURE.ENTRY2 IS ?FIXTURE2)
    (?FIXTURE2 IS IN CLASS SPEED.GRIPS)
    (LISP (NOT (EQUAL ?MACHINE1 ?MACHINE2)))
    (CANT.FIND
     (OR (?FIXTURE1 IS IN CLASS COMBINATION.FIXTURES)
         (?FIXTURE2 IS IN CLASS COMBINATION.FIXTURES)))
    THEN
    IN.NEW.AND.WORLD
    (THE PREVIOUS.MACHINE OF ?MACHINE2 IS ?MACHINE1)
    (THE NEXT.MACHINE OF ?MACHINE1 IS ?MACHINE2))
```

Fixturing Knowledge Source

This knowledge source is responsible for making decisions about how a part will be fixtured to perform a subset of macro-operations required to manufacture the part. Decisions made by this knowledge source result primarily in the generation and instantiation

of entries on the domain blackboard at the fixture level. *Fixturing.K.S-1* indicates: If a *Straight.Inner.Diameter* feature is a datum surface and the *diameter*, *diameter tolerance*, and *length* of the feature is within the capabilities of a speed-grip, then this feature can be used as the *primary.fixturing.surface* for the speed-grip bushing.

```
Rule:  FIXTURING.K.S-1

(IF (THE FIXTURE OF DOMAIN.BLACKBOARD IS ?FIXTURE.ENTRY)
    (THE DESCRIPTION OF ?FIXTURE.ENTRY IS ?FIXTURE)
    (?FIXTURE IS IN CLASS SPEED.GRIPS)
    (?DATUM.SURFACE IS IN DATUM.SURFACES)
    (OR (THE DATUM.TYPE OF ?DATUM.SURFACE IS A)
        (THE DATUM.TYPE OF ?DATUM.SURFACE IS B))
    (THE PRIMITIVE.FEATURE.INSTANCE OF ?DATUM.SURFACE
                IS ?PRIMITIVE.FEATURE)
    (?PRIMITIVE.FEATURE IS IN CLASS STRAIGHT.INNER.DIAMETERS)
    (THE KIND.OF.PART OF ?PRIMITIVE.FEATURE IS ?FINISHED.PART)
    (?FINISHED.PART IS IN CLASS FINISHED.PART)
    (LISP (SPEED.GRIP.COMPATIBLE? ?FIXTURE ?PART ?PRIMITIVE.FEATURE))
    THEN
    IN.NEW.AND.WORLD
    (THE PRIMARY.FIXTURING.SURFACE OF ?FIXTURE.ENTRY IS ?PRIMITIVE.FEATURE)
    (LISP (INFORM.ABOUT.SPEED.GRIP.PRIMARY.SURFACE ?FIXTURE
                ?PRIMITIVE.FEATURE ?$WORLD$)))
```

The condition part of *Fixturing.K.S-2* ascertains whether a geometrical tolerance has been specified on a *Straight.External.Face* feature with respect to the feature chosen as the *primary.fixturing.surface* for a speed-grip. A check is also made to determine whether such a feature is appropriately located on the part for the workring of the speed-grip. The action part indicates that the external face feature can be used as the *secondary.fixturing.surface* for speed-gripping the part.

```
Rule:  FIXTURING.K.S-2

(IF (THE FIXTURE OF DOMAIN.BLACKBOARD IS ?FIXTURE.ENTRY)
    (THE PRIMARY.FIXTURING.SURFACE OF ?FIXTURE.ENTRY IS ?PRIMARY.FEATURE)
    (THE DESCRIPTION OF ?FIXTURE.ENTRY IS ?FIXTURE)
    (?FIXTURE IS IN CLASS SPEED.GRIPS)
    (THE DATUM OF ?PRIMITIVE.FEATURE IS ?DATUM)
    (THE DATUM.TYPE OF ?DATUM IS ?DATUM.TYPE)
    (EQUAL ?SECONDARY.SURFACE.CLASS
           (LISP (S.G.SECONDARY.SURFACE.CLASS? ?FIXTURE
                        ?PART ?PRIMARY.FEATURE)))
    (?GEOMETRIC.TOLERANCE IS IN CLASS RELATED.FORM.TOLERANCES)
    (NOT (OR (?GEOMETRIC.TOLERANCE IS IN CLASS PROFILE.OF.A.LINE)
             (?GEOMETRIC.TOLERANCE IS IN CLASS PROFILE.OF.A.SURFACE)))
    (LISP (MEMBER ?DATUM.TYPE (GET.VALUES ?GEOMETRIC.TOLERANCE
                                    'DATUM.SURFACES)))
    (THE PRIMITIVE.FEATURE OF ?GEOMETRIC.TOLERANCE IS ?SECONDARY.FEATURE)
    (?SECONDARY.FEATURE IS IN CLASS ?SECONDARY.SURFACE.CLASS)
```

```
(THE KIND.OF.PART OF ?SECONDARY.FEATURE IS ?FINISHED.PART)
(?FINISHED.PART IS IN CLASS FINISHED.PART)
THEN
IN.NEW.AND.WORLD
(THE SECONDARY.FIXTURING.SURFACE OF ?FIXTURE.ENTRY
                    IS ?SECONDARY.FEATURE)
(LISP (INFORM.ABOUT.SPEED.GRIP.SECONDARY.SURFACE
              ?FIXTURE ?SECONDARY.FEATURE ?$WORLD$)))
```

Operation Knowledge Source

This knowledge source is responsible for determining the operations that will be performed to manufacture a particular part. The main result of decisions made by this knowledge source is the creation of Macro-Operation and Micro-Operation entries. For example, the condition part of *Operation.K.S-1* determines whether a speed-grip is being used as a fixture and whether the *primary.fixturing.surface*, *secondary.fixturing.surface*, and the fixture to be used before the part is held in the speed-grip (*?previous.fixture*) are known. The action part prescribes the creation of Macro-Operation entries with their *feature*, *description*, and *fixture* attributes appropriately instantiated. This rule indicates that one way to manufacture the features used as the *primary.fixturing.surface* and *secondary.fixturing.surface* is to machine them while the part is held using the fixture *?previous.fixture*.

```
Rule:  OPERATION.K.S-1

(IF (THE FIXTURE OF DOMAIN.BLACKBOARD IS ?FIXTURE.ENTRY)
    (THE DESCRIPTION OF ?FIXTURE.ENTRY IS ?FIXTURE)
```

```
(?FIXTURE IS IN CLASS SPEED.GRIPS)
(THE PRIMARY.FIXTURING.SURFACE OF ?FIXTURE.ENTRY IS ?PRIMITIVE.FEATURE)
(THE SECONDARY.FIXTURING.SURFACE OF ?FIXTURE.ENTRY
                        IS ?SECONDARY.FEATURE)
(THE PREVIOUS.FIXTURE OF ?FIXTURE.ENTRY IS ?PREVIOUS.FIXTURE)
(EQUAL ?MACRO-OPERATION.ENTRY1
       (K.S.EQUAL
            (FORMAT NIL "~A-~A" ?PRIMITIVE.FEATURE ?PREVIOUS.FIXTURE)
            (UNITMSG '(MACRO-OPERATION.ENTRY BLACKBOARD-LIBRARY)
                     'CREATE.NEW.INSTANCE)))
(EQUAL ?MACRO-OPERATION.ENTRY2
       (K.S.EQUAL
            (FORMAT NIL "~A-~A" ?SECONDARY.FEATURE ?PREVIOUS.FIXTURE)
            (UNITMSG '(MACRO-OPERATION.ENTRY BLACKBOARD-LIBRARY)
                     'CREATE.NEW.INSTANCE)))
THEN
IN.NEW.AND.WORLD
(THE FEATURE OF ?MACRO-OPERATION.ENTRY1 IS ?PRIMITIVE.FEATURE)
(THE DESCRIPTION OF ?MACRO-OPERATION.ENTRY1 IS BORING)
(THE FIXTURE OF ?MACRO-OPERATION.ENTRY1 IS ?PREVIOUS.FIXTURE)
(A MACRO-OPERATION OF ?PREVIOUS.FIXTURE IS ?MACRO-OPERATION.ENTRY1)
(THE MACRO-OPERATION OF DOMAIN.BLACKBOARD IS ?MACRO-OPERATION.ENTRY1)
(THE FEATURE OF ?MACRO-OPERATION.ENTRY2 IS ?SECONDARY.FEATURE)
(THE DESCRIPTION OF ?MACRO-OPERATION.ENTRY2 IS FACING)
(THE FIXTURE OF ?MACRO-OPERATION.ENTRY2 IS ?PREVIOUS.FIXTURE)
```

```
    (A MACRO-OPERATION OF ?PREVIOUS.FIXTURE IS ?MACRO-OPERATION.ENTRY2)
    (THE MACRO-OPERATION OF DOMAIN.BLACKBOARD IS ?MACRO-OPERATION.ENTRY2))
```

Operation.K.S-2 indicates that if a Macro-Operation entry involves drilling a *Straight.Circular.Hole* feature, and a tight position geometric tolerance has been specified for the feature (for example, a tolerance value less than 0.05 mm), then this operation will have to be completed in two micro-operation passes, namely "start drill" and "finish drill."

```
Rule:  OPERATION.K.S-2

(IF (THE MACRO-OPERATION OF DOMAIN.BLACKBOARD IS ?MACRO-OPERATION.ENTRY)
    (THE FEATURE OF ?MACRO-OPERATION.ENTRY IS ?FEATURE)
    (?FEATURE IS IN CLASS STRAIGHT.CIRCULAR.HOLES)
    (THE DESCRIPTION OF ?MACRO-OPERATION.ENTRY IS DRILLING)
    (THE FIXTURE OF ?MACRO-OPERATION.ENTRY IS ?FIXTURE.ENTRY)
    (THE MACHINE OF ?FIXTURE.ENTRY IS ?MACHINE.ENTRY)
    (THE DESCRIPTION OF ?MACHINE.ENTRY IS ?MACHINE)
    (LISP (TIGHT.POSITION.TOLERANCE? ?MACHINE ?FEATURE))
    (EQUAL ?MICRO-OPERATION.ENTRY1
           (K.S.EQUAL (FORMAT NIL "~A-~A-1" ?FEATURE ?MACHINE)
                      (UNITMSG '(MICRO-OPERATION BLACKBOARD-LIBRARY)
                               'CREATE.NEW.INSTANCE)))
    (EQUAL ?MICRO-OPERATION.ENTRY2
           (K.S.EQUAL (FORMAT NIL "~A-~A-2" ?FEATURE ?MACHINE)
                      (UNITMSG '(MICRO-OPERATION BLACKBOARD-LIBRARY)
                               'CREATE.NEW.INSTANCE)))
```

```
THEN
DO
(THE NO.OF.PASSES OF ?MACRO-OPERATION.ENTRY IS 2)
(THE DESCRIPTION OF ?MICRO-OPERATION.ENTRY1 IS START.DRILL)
(THE MACRO-OPERATION OF ?MICRO-OPERATION.ENTRY1
                    IS ?MACRO-OPERATION.ENTRY)
(THE MICRO-OPERATION OF ?MACRO-OPERATION.ENTRY IS
                    ?MICRO-OPERATION.ENTRY1)
(THE MICRO-OPERATION OF DOMAIN.BLACKBOARD IS
                    ?MICRO-OPERATION.ENTRY1)
(THE DESCRIPTION OF ?MICRO-OPERATION.ENTRY2 IS FINISH.DRILL)
(THE MACRO-OPERATION OF ?MICRO-OPERATION.ENTRY2
                    IS ?MACRO-OPERATION.ENTRY)
(THE MICRO-OPERATION OF ?MACRO-OPERATION.ENTRY IS
                    ?MICRO-OPERATION.ENTRY2)
(THE MICRO-OPERATION OF DOMAIN.BLACKBOARD IS
                    ?MICRO-OPERATION.ENTRY2)
```

Operation Sequencing Knowledge Source

This knowledge source is responsible for specifying the partial or complete ordering of macro- and micro-operations. For example, the condition part of *Operation.Sequence.K.S-1* determines whether a Macro-Operation entry involves machining a *Straight.Inner.Diameter* feature and whether the *feature.to.the.left* and *feature.to.the.right* of the inner diameter feature are in the class *negative.normal.internal.faces.* A check is also made to determine if any decision has been made about how the part will be fixtured while these operations are

performed. The action part indicates that the internal face features can be machined at the same time as the straight inner diameter feature.

```
Rule:  OPERATION.SEQUENCING.K.S-1

(IF (THE MACRO-OPERATION OF DOMAIN.BLACKBOARD IS ?MACRO-OPERATION.ENTRY2)
    (THE FEATURE OF ?MACRO-OPERATION.ENTRY2 IS ?PRIMITIVE.FEATURE)
    (OR (?PRIMITIVE.FEATURE IS IN CLASS STRAIGHT.INNER.DIAMETERS)
        (?PRIMITIVE.FEATURE IS IN CLASS NEGATIVE.NORMAL.INNER.DIAMETERS))
    (THE FEATURE.TO.THE.LEFT OF ?PRIMITIVE.FEATURE IS ?FEATURE-A)
    (?FEATURE-A IS IN CLASS NEGATIVE.NORMAL.INTERNAL.FACES)
    (THE FEATURE.TO.THE.RIGHT OF ?PRIMITIVE.FEATURE IS ?FEATURE-B)
    (?FEATURE-B IS IN CLASS NEGATIVE.NORMAL.INTERNAL.FACES)
    (THE MACRO-OPERATION OF DOMAIN.BLACKBOARD IS ?MACRO-OPERATION.ENTRY1)
    (THE FEATURE OF ?MACRO-OPERATION.ENTRY1 IS ?FEATURE-A)
    (THE MACRO-OPERATION OF DOMAIN.BLACKBOARD IS ?MACRO-OPERATION.ENTRY3)
    (THE FEATURE OF ?MACRO-OPERATION.ENTRY3 IS ?FEATURE-B)
    (CANT.FIND (THE FIXTURE OF ?MACRO-OPERATION.ENTRY1 IS ?SOME.FIXTURE))
    (CANT.FIND (THE FIXTURE OF ?MACRO-OPERATION.ENTRY2 IS ?SOME.FIXTURE))
    (CANT.FIND (THE FIXTURE OF ?MACRO-OPERATION.ENTRY3 IS ?SOME.FIXTURE))
    THEN
    DO
    (THE NEXT.MACRO-OPERATION OF ?MACRO-OPERATION.ENTRY1 IS
                ?MACRO-OPERATION.ENTRY2)
    (THE PREVIOUS.MACRO-OPERATION OF ?MACRO-OPERATION.ENTRY2 IS
```

```
                ?MACRO-OPERATION.ENTRY1)
    (THE NEXT.MACRO-OPERATION OF ?MACRO-OPERATION.ENTRY2 IS
                ?MACRO-OPERATION.ENTRY3)
    (THE PREVIOUS.MACRO-OPERATION OF ?MACRO-OPERATION.ENTRY3 IS
                ?MACRO-OPERATION.ENTRY2))
```

The condition part of *Operation.Sequence.K.S-2* ascertains whether two Micro-Operation entries involve a rough facing operation that is performed with the same fixture and *facing.tool.assembly* and whether the primitive features that they machine are adjacent.[5.9] The action part indicates that there is a partial ordering between these two micro-operations.

```
Rule:  OPERATION.SEQUENCING.K.S-2

(IF (THE MICRO.OPERATION OF DOMAIN.BLACKBOARD IS ?MICRO-OPERATION.ENTRY1)
    (THE DESCRIPTION OF ?MICRO-OPERATION.ENTRY1 IS ROUGHING)
    (THE CUTTING.TOOL.ASSEMBLY OF ?MICRO-OPERATION.ENTRY1 IS
                ?CUTTING.TOOL.ASSEMBLY)
    (?CUTTING.TOOL.ASSEMBLY IS IN CLASS FACING.TOOL.ASSEMBLIES)
    (THE MACRO-OPERATION OF ?MICRO-OPERATION.ENTRY1
                IS ?MACRO-OPERATION.ENTRY1)
    (THE FIXTURE OF ?MACRO-OPERATION.ENTRY1 IS ?FIXTURE.ENTRY)
    (LISP
     (RIGHT.ADJACENT.MICRO-OPERATION? ?FIXTURE.ENTRY ?FEATURE
```

[5.9] Not necessarily physically adjacent. Two feature instances belonging to the same class are adjacent if no other feature instance from that class occurs between them in the feature-based model of the part.

```
        'ROUGHING ?CUTTING.TOOL.ASSEMBLY ?$WORLD$))
(EQUAL ?MICRO-OPERATION.ENTRY2
       (RIGHT.ADJACENT.MICRO-OPERATION?
            ?FIXTURE.ENTRY ?FEATURE 'ROUGHING
                 ?CUTTING.TOOL.ASSEMBLY ?$WORLD$))
THEN
DO
(THE NEXT.MICRO-OPERATION OF ?MICRO-OPERATION.ENTRY1 IS
       ?MICRO-OPERATION.ENTRY2)
(THE PREVIOUS.MICRO-OPERATION OF ?MICRO-OPERATION.ENTRY2 IS
       ?MICRO-OPERATION.ENTRY1))
```

Cutting Tool Knowledge Source

This is the knowledge source responsible for determining the cutting tool assemblies and the parameters of cutting tools that will be used to perform micro-operations. *Cutting.Tool.K.S-1*, for example, indicates that if a Micro-Operation entry involves "rough turning" a straight outer diameter feature, then the operation can be performed if an appropriate *turning.tool.assembly* is available in the *Tool.Matrix* of the machine that will be used to perform the operation. The cutting tool parameters of the chosen *turning.tool.assembly* have to be within the range of allowable values for an ideal "rough turn" operation.

```
Rule:  CUTTING.TOOL.K.S-1

(IF (THE MICRO-OPERATION OF DOMAIN.BLACKBOARD IS ?MICRO-OPERATION.ENTRY)
    (THE DESCRIPTION OF ?MICRO-OPERATION.ENTRY IS ROUGHING)
```

```
(THE MACRO-OPERATION OF ?MICRO-OPERATION.ENTRY IS ?MACRO-OPERATION)
(THE DESCRIPTION OF ?MACRO-OPERATION IS TURNING)
(THE FEATURE OF ?MACRO-OPERATION IS ?FEATURE)
(?FEATURE IS IN CLASS STRAIGHT.OUTER.DIAMETERS)
(THE FIXTURE OF ?MACRO-OPERATION IS ?FIXTURE)
(THE MACHINE OF ?FIXTURE IS ?MACHINE.ENTRY)
(THE DESCRIPTION OF ?MACHINE.ENTRY IS ?MACHINE)
(THE TOOL.MATRIX OF ?MACHINE IS ?TOOL.MATRIX)
(LISP (ROUGH.TURN.TOOL.ASSEMBLY? ?TOOL.MATRIX))
(EQUAL ?CUTTING.TOOL.ASSEMBLY (ROUGH.TURN.TOOL.ASSEMBLY? ?TOOL.MATRIX))
THEN
(THE CUTTING.TOOL.ASSEMBLY OF ?MICRO-OPERATION.ENTRY IS
     ?CUTTING.TOOL.ASSEMBLY))
```

The condition part of *Cutting.Tool.K.S-2* ascertains whether the following facts are true: (1) A Micro-Operation entry involves a "roughing" operation for a chamfer, (2) The feature adjacent to the chamfer is a straight external face, (3) The "roughing" operation for the face is performed with a *facing.tool.assembly*, and (4) No other external features of the part are "rough turned" while the part is held in the same fixture to perform the roughing operations. The action part indicates that the *facing.tool.assembly* can also be used to "rough" the chamfer.

```
Rule:  CUTTING.TOOL.K.S-2

(IF (THE MICRO-OPERATION OF DOMAIN.BLACKBOARD IS ?MICRO-OPERATION.ENTRY)
    (THE DESCRIPTION OF ?MICRO-OPERATION.ENTRY IS ROUGHING)
```

```
(THE MACRO-OPERATION OF ?MICRO-OPERATION.ENTRY IS ?MACRO-OPERATION)
(THE FEATURE OF ?MACRO-OPERATION IS ?FEATURE)
(OR (?FEATURE IS IN CLASS POSITIVE.NORMAL.EXTERNAL.CHAMFERS)
    (?FEATURE IS IN CLASS NEGATIVE.NORMAL.EXTERNAL.CHAMFERS))
(THE FIXTURE OF ?MACRO-OPERATION IS ?FIXTURE)
(LISP (ADJACENT.STRAIGHT.EXTERNAL.FACE.FEATURE? ?FEATURE))
(EQUAL ?ADJACENT.FEATURE (ADJACENT.STRAIGHT.EXTERNAL.FACE.FEATURE?
                                ?FEATURE))
(THE MACRO-OPERATION OF ?FIXTURE IS ?MACRO-OPERATION.ENTRY-2)
(THE FEATURE OF ?MACRO-OPERATION.ENTRY-2 IS ?ADJACENT.FEATURE)
(THE MICRO-OPERATION OF ?MACRO-OPERATION.ENTRY-2 IS
     ?MICRO-OPERATION.ENTRY-2)
(THE DESCRIPTION OF ?MICRO-OPERATION.ENTRY-2 IS ROUGHING)
(THE CUTTING.TOOL.ASSEMBLY OF ?MICRO-OPERATION.ENTRY-2 IS
     ?CUTTING.TOOL.ASSEMBLY)
(?CUTTING.TOOL.ASSEMBLY IS IN CLASS FACING.TOOL.ASSEMBLIES)
(LISP (LIMITED.ROUGHING.OPERATION? ?FIXTURE
              ?FEATURE ?ADJACENT.FEATURE ?$WORLD$))
THEN
IN.NEW.AND.WORLD
(THE CUTTING.TOOL.ASSEMBLY OF ?MICRO-OPERATION.ENTRY IS
          ?CUTTING.TOOL.ASSEMBLY))
```

Cutting Parameters Knowledge Source

This knowledge source is responsible for determining the cutting parameters (cutting speed, feedrate, and depth of cut) to be used for a particular micro-operation. This is the knowledge source that would use the generative process performance model described in Chapter 4 to determine values for the various cutting parameters. However, this knowledge source has not been currently implemented because of the unavailability of relevant process performance models in the domain chosen for implementation (see Chapter 6 for more details).

Machining Concerns Knowledge Source

This is the knowledge source responsible for providing feedback to designers about any machining-related concerns that arise. Instances of this knowledge source are always applicable only in a single context. *Machining.Concern.K.S-1*, for example, indicates that if a *Straight.Inner.Diameter* is one of the features of a part that will be made in a manufacturing cell and the tolerance on its *diameter* is below the capabilities of the machines in the facility, then an extra grinding operation would be needed to machine the feature.

```
Rule:  MACHINING.CONCERNS.K.S-1

(IF (THE PART OF DOMAIN.BLACKBOARD IS ?PART)
    (THE CURRENT.FEATURE.INSTANCE OF ?PART IS ?FEATURE)
    (?FEATURE IS IN CLASS STRAIGHT.INNER.DIAMETERS)
    (THE FACILITY OF ?PART IS ?BLACKBOARD-FACILITY)
    (THE DESCRIPTION OF ?BLACKBOARD-FACILITY IS ?FACILITY)
    (LISP (BELOW.I.D.DIAMETER.TOLERANCE.LIMIT ?FEATURE ?FACILITY))
    THEN
```

```
DO
(LISP (INFORM.ABOUT.I.D.TOLERANCE ?FEATURE ?FACILITY ?$WORLD$)))
```

Machining.Concern.K.S-2 indicates: if there are a minimum of three Micro-Operation entries that generate *fillets* of radii less than a critical radius (e.g. 0.762 mm), if all three operations use the same cutting tool assembly on a particular machine, and if the features are machined with the part held in the same fixture, then rapid tool wear will take place unless the radius of some of the fillets is increased.

```
Rule:  MACHINING.CONCERNS.K.S-2

(IF (THE MICRO-OPERATION OF DOMAIN.BLACKBOARD IS ?MICRO-OPERATION)
    (THE DESCRIPTION OF ?MICRO-OPERATION IS FINISHING)
    (THE MACRO-OPERATION OF ?MICRO-OPERATION IS ?MACRO-OPERATION)
    (THE FEATURE OF ?MACRO-OPERATION IS ?FEATURE)
    (OR (?FEATURE IS IN CLASS NEGATIVE.NORMAL.INTERNAL.FILLETS)
        (?FEATURE IS IN CLASS POSITIVE.NORMAL.INTERNAL.FILLETS))
    (LISP (CRITICAL.RADIUS? ?FEATURE))
    (THE FIXTURE OF ?MACRO-OPERATION IS ?FIXTURE)
    (THE CUTTING.TOOL.ASSEMBLY OF ?MICRO-OPERATION
        IS ?CUTTING.TOOL.ASSEMBLY)
    (LISP (SEQUENCE.OF.FILLETS? ?FIXTURE ?FEATURE ?CUTTING.TOOL.ASSEMBLY))
    THEN
    DO
    (LISP (INFORM.ABOUT.RAPID.TOOL.WEAR ?FIXTURE ?FEATURE
                        ?CUTTING.TOOL.ASSEMBLY ?$WORLD$)))
```

5.5 Summary

The reasoning subsystem of the design environment has been described in detail in this chapter. The Multiple Cooperative Knowledge Sources Paradigm was shown to satisfy the basic requirements of the reasoning subsystem. Under this approach, domain knowledge sources were responsible for developing the product design and the process plan and the manager was responsible for controlling the concurrency between these two tasks. The blackboard of such a system consists of two parts: Control and Domain Blackboard. The control blackboard is used by the manager to develop the "plan" for concurrency, and the domain blackboard is used by the domain knowledge sources to describe the product and manufacturing facility models and the process plan for a particular facility. A detailed description of the control and domain blackboard levels and the manager and domain knowledge sources was presented.

Chapter 6

IMPLEMENTATION

6.1 Introduction

A description of the implementation of the Simultaneous Engineering concept for components manufactured in small and medium lot-sizes is presented in this chapter. A description of the domain chosen for implementation is presented in Section 6.2 followed by the salient points of implementing the Commodity Sourcing concept in this domain. The layout of the user-interface of the computer-based design environment is described in Section 6.3. Two different example problems used to present the details of the actual working of the design environment are presented in Sections 6.4 and 6.5.

6.2 Choice of Domain

Concurrent product and process design of bearing cages is the domain chosen for implementing the Simultaneous Engineering concept for components manufactured in small and medium lot-sizes. bearing cages are the components in a sub-assembly or assembly that are primarily used to house and properly locate bearings. They are also used for several auxiliary functions such as localizing and preventing seepage of oil, serving as end caps for sub-assemblies, and providing sufficient lubrication for bearings and other components in

an assembly.

Commodity Sourcing is the first step in implementing the Simultaneous Engineering concept. This task was not performed as part of this research. However, important results obtained after performing this task are summarized here for completeness. For bearing cages, this process involved the study of 1584 existing part designs, elimination of duplicate part designs (total of 967 part designs), and design of special manufacturing cells based on the general characteristics of the remaining 617 part designs. Since 95% of these part designs were made of cast iron with different compositions, the special manufacturing cells were restricted to manufacturing bearing cages made out of the above material. The remaining bearing cages were made of aluminium and steel and were slated to be purchased from outside suppliers.

In order to design the manufacturing cells, the general characteristics of the parts made of cast iron were determined. Some examples of the general characteristics of parts observed were: 89% of the parts have *Outer.Diameter* features whose diameter is within the range of 2SC/2AC Lathes, 95% of the parts had *Inner.Diameter* features whose diameter was greater than 63.5mm, and 70% of the parts had some type of internal or external *Groove*. Based on such an analysis, it was decided to design and develop four different kinds of manufacturing cells that could make these bearing cages at the lowest possible production cost. These cells had several common and distinct capabilities. For example, a capability common to all cells was the availability of a machining center to drill holes. A distinguishing factor was that certain cells were designed to be set up and "retooled" quickly, making them suitable to manufacture bearing cages in very small lot-sizes while other cells were designed to be "retooled" less often and were suitable for manufacturing bearing cages in medium lot-sizes. The design and development of these cells marked the end of the first stage in implementing the Simultaneous Engineering concept.

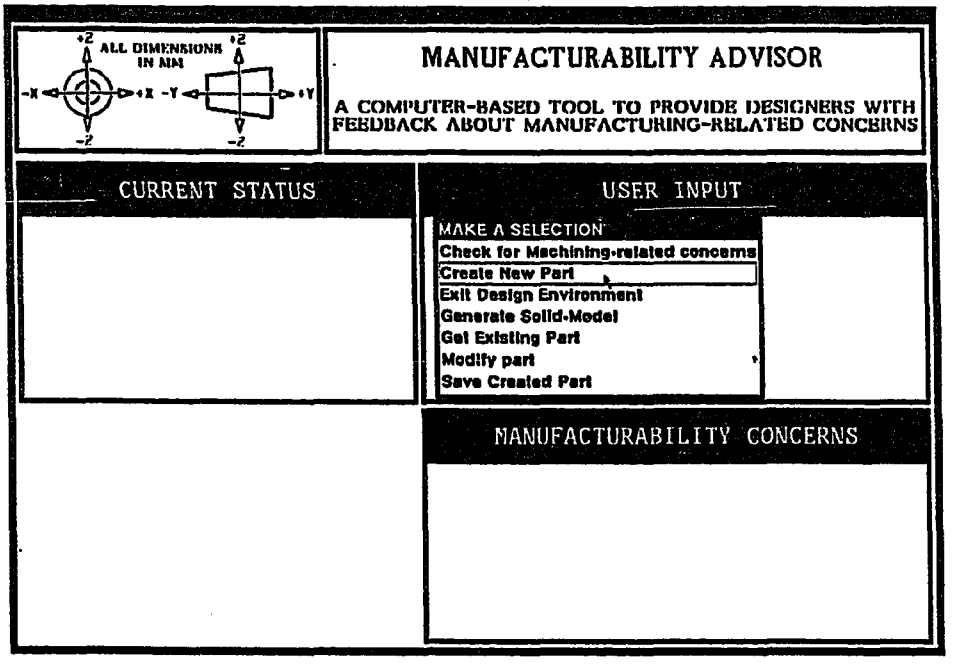

Figure 6.1: User Interface

6.3 Design Environment User-Interface

The computer-based design environment has been built on a Symbolics 3640 workstation through extensions to the general capabilities of a hybrid knowledge-based system development tool: Knowledge Engineering Environment (KEE) [97]. The User-Interface of the design environment is shown in Fig. 6.1. Keepictures [97], an object-oriented graphics system, has been used to develop the user-interface. "Boxes" and "Lines," which are standard picture classes within Keepictures, have been used to develop various portions of the interface. The user interacts with the design environment through keyboard input or by choosing from a menu of options using a mouse.

The computer screen displaying the user-interface is split into three main regions. The

region entitled Current Status indicates the current state of the product design activity. It is used as a help facility for the designer and displays the primitive and compound form features that have been created as well as the form feature currently being created. The region entitled User Input enables the designer to describe relevant information about a particular product design. The region entitled Manufacturability Concerns is used to inform the product designer about any machining-related concerns that have arisen. The designer's primary mode of interaction with the system is by creating and modifying the design of a part in terms of its product features. The designer can also save and retrieve part designs that have been created.

6.4 Demonstration of Design Environment

An example of the method by which the computer-based design environment is used to provide feedback to designers about machining-related concerns is presented in this section [98]. The cooperative product development activities of the product and process refinement domain knowledge sources are described, and representative examples of two kinds of machining-related concerns are presented. The first type of concerns is those for which one need not have prior knowledge of how a part will be processed through a facility in order to determine if these concerns have arisen. The second type of concerns is those for which at least some knowledge of how a part will be processed through a facility has to be known before it can be determined that these concerns need to be resolved.

6.4.1 Creating a Part Design

The bearing cage design that will be initially created as part of this demonstration is shown in Fig. 6.2. The basic shape of this part is similar to the finished part design shown in Fig. 4.6 except for the absence of certain features like fillets, chamfers, grooves, and through holes.

The design in Fig. 6.2 is at an intermediate stage, where the designer has finished developing the basic shape of the bearing cage but has not yet decided how it will be bolted to a sub-assembly. This illustrates that the designer can interact with the design environment at all stages in the design process and not necessarily after the complete product design has been finalized.

The designer begins the product design process by choosing the "Create New Part" option from the initial menu. This choice leads to a Problem entry being created on the *Control.Blackboard* and control of the product development activity is passed to the manager. As mentioned in Chapter 5, in order to activate skeletal plans if their pre-conditions are satisfied, domain-specific control knowledge source instances would have been created before the start of the problem-solving process. The posting of the entry at the problem level satisfies the preconditions for one such knowledge source instance, and a Strategy decision entry is created on the control blackboard. The manager's initial strategy is: develop new part design; determine the facilities that can be used to manufacture the part; choose an appropriate facility; and, process the part for the facility chosen. The initial strategy is primarily sequential in nature.

With the instantiation of the Strategy entry, several generic control knowledge source instances are triggered and placed on the Agenda. One of these instances *Manager-3* is chosen for execution and leads to the instantiation of another Strategy entry for the first sub-strategy, i.e. "develop new part design." This strategy, in turn, consists of three Focus entries as specified by the skeletal plan: develop general part characteristics; design external features; and, design internal features. The activation and deactivation of these entries are controlled by generic and domain-independent control knowledge source instances. The initial focus of the manager is on the development of the general characteristics of the part

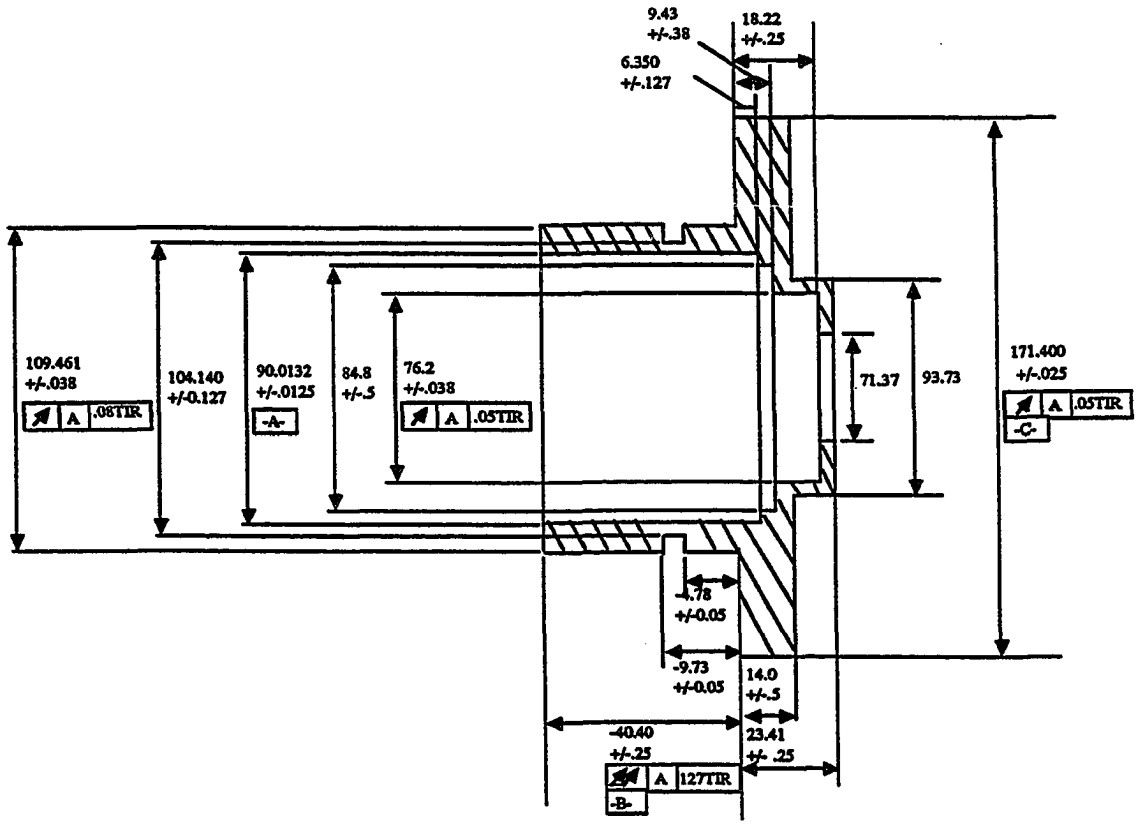

Figure 6.2: Preliminary Bearing Cage Design

being designed so that (if possible) process-related decisions[6.1] can be made, enabling more concurrent and less sequential product and process design.

Domain-level problem-solving[6.2] begins by concentrating on the first Focus entry. Control is passed to the product designer who develops a general high-level description of the part. Commodity type, material specifications, monthly lot-size, overall length, and the largest diameter of the part are some part characteristics that can be used to generate the

[6.1] Decisions regarding the process design or the process plan.
[6.2] That is, creation of decision entries on the domain blackboard.

high-level description of the part. The designer can specify such characteristics by choosing from a menu of options. For the product design under consideration, the type of commodity and material are specified. These product design decisions lead to the creation of a Part entry on the domain blackboard with appropriate attribute values.

Once the designer finishes developing the general part characteristics, additional knowledge source instances are triggered. An instance of the *Sourcing.K.S* is scheduled, and it is determined that the manufacturing cells designed for bearing cages can be used to make this part. This leads to the creation of an entry at the facility level on the domain blackboard. This is one example of a process-related decision being made while the design is in progress. The *Sourcing.K.S.* is able to make this decision because of a policy level decision, established by the manager at the start of the problem-solving activity, giving importance to such decisions. Subsequent to this "sourcing" decision, an additional Focus entry is created by the manager enabling feedback (to the designer) of machining-related concerns common to all bearing cage cells. This control level decision changes the original strategy of the manager and makes the product development activity more concurrent and less sequential.

Eventually, a domain-independent control knowledge source instance is scheduled for interpretation leading to the current focus being made inoperative. After this, other control knowledge source instances possessing higher priority change the focus to designing the external features of the part. The designer individually creates the external features of the part by choosing from a menu of options (see Fig. 6.3). Figure 6.4 shows the designer creating and specifying a subset of the parameters of one of the *Straight.External.Faces* of the part design. A parametric description of the feature is displayed, and the designer can obtain a description of a particular parameter by placing and clicking the mouse on that parameter. Each individual feature instance that is created is placed on the domain

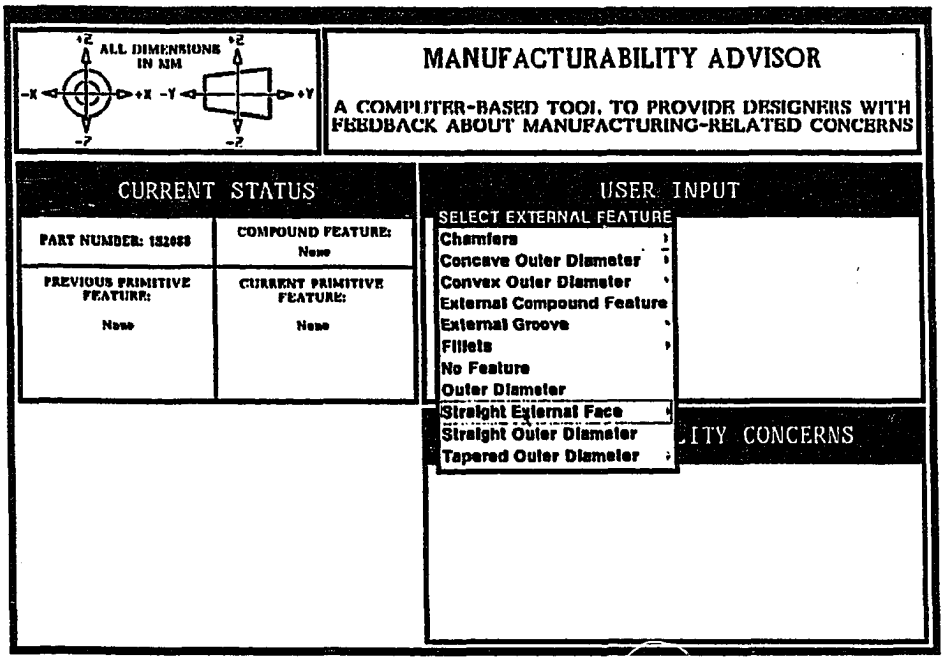

Figure 6.3: External Features Menu

blackboard and is associated with the entry that was made earlier at the part level on this blackboard.

While generating the design of the form features, the designer does not have to indicate all the parameter values because certain values can be inferred based on adjacency relationships. For example, Fig. 6.5 shows the creation of the *Straight.Outer.Diameter* feature adjacent to the *External.Face* feature created earlier. A relationship always needs to be maintained between the three feature parameters: *left.end.distance*, *right.end.distance*, and *length* of the *Outer.Diameter* feature. The *left.end.distance* parameter value can be determined based on adjacency relationships between the face and the outer diameter features. The designer has to specify the value for only one of the two remaining parameters and the design environment will be able to infer the value for the third parameter. Figure 6.5 shows

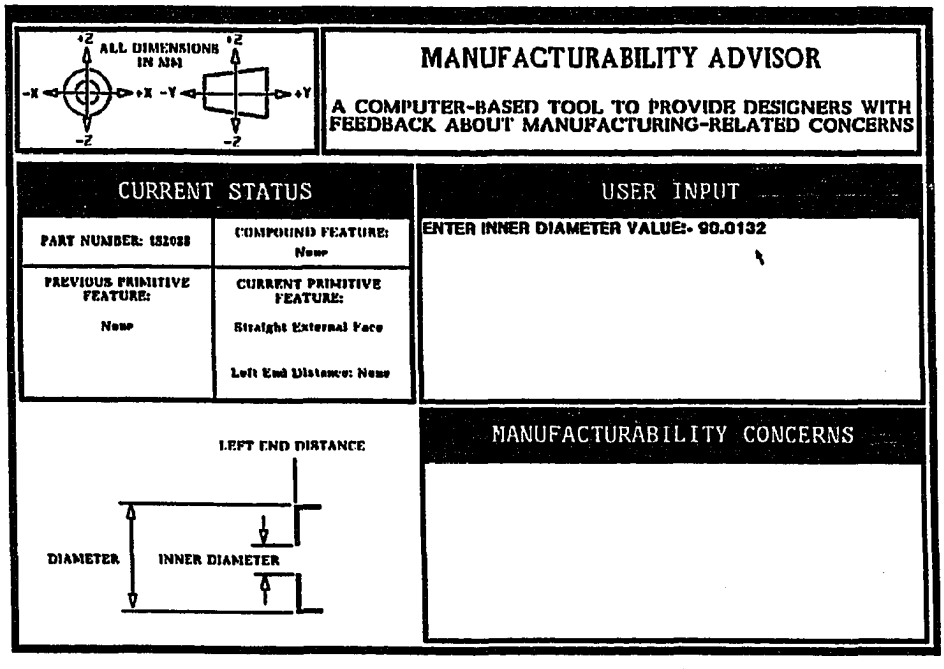

Figure 6.4: Creation of Straight External Face

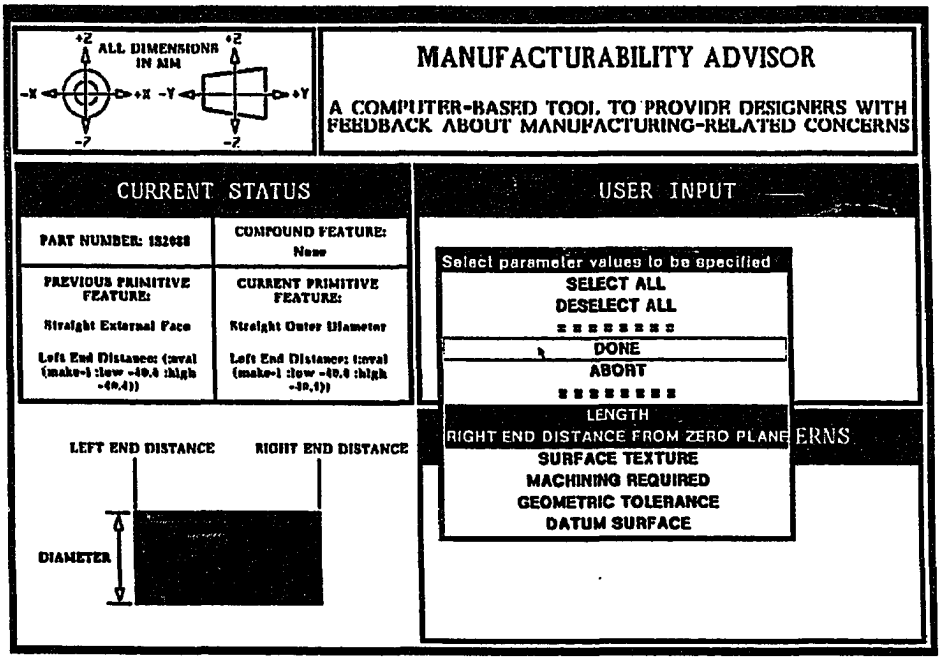

Figure 6.5: Creation of Straight Outer Diameter

the designer choosing from the menu of options to specify both the parameters instead of one. If inconsistent values are specified for these parameters, they would be detected by the constraint-based system. As shown in Fig. 6.6, the designer will be prompted to retract the value of one of the three parameters responsible for the inconsistency.

Once the external features of the part have been created, the focus changes to creating the internal features of the part. Internal features are created by following the same approach used to create the external features of the part. This results in the development of a complete feature-based model of the part. Note that the geometric model of the part would also have been generated concurrently. The creation of these two models leads to the fulfillment of the first sub-strategy, "develop new part design."

Due to changes made earlier in the manager's strategy, the second sub-strategy is triv-

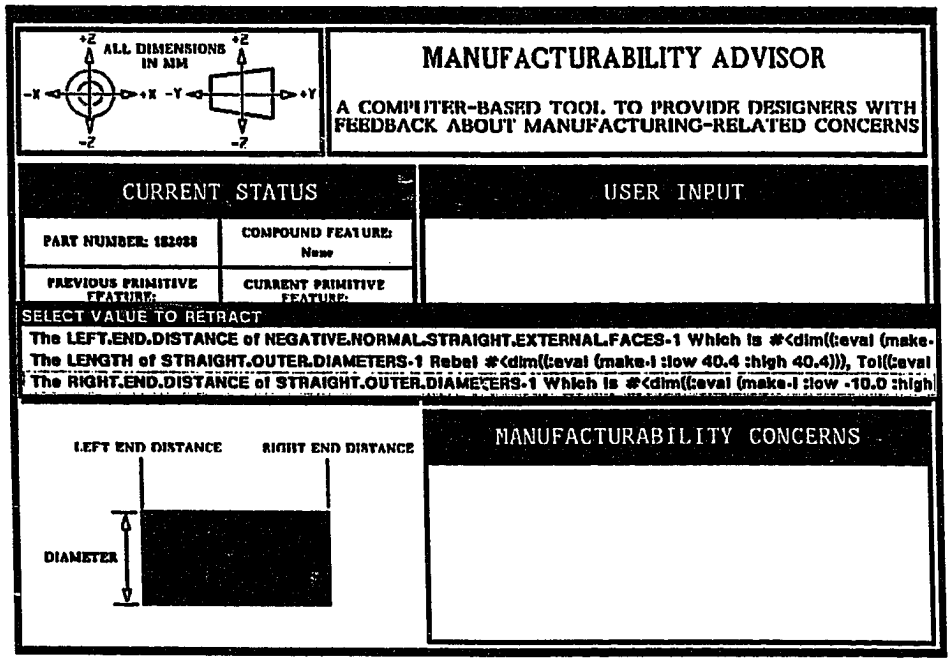

Figure 6.6: Determination of Inconsistent Feature

ially satisfied because the facilities that can be used to manufacture the part have already been determined. The third sub-strategy, "choose a particular facility," becomes active through the creation of a Strategy level entry. Since all the general characteristics of the part have not been determined, the initial strategy of the manager is to create two Focuses: determine additional part characteristics; and determine the facility to manufacture the part. The creation of the first Focus entry passes control to the product designer. The only part characteristics specified by the designer is the monthly lot-size.[6.3] The creation and activation of the second Focus entry triggers instances of *Sourcing.K.S* and one of them is scheduled for interpretation. Since the facility finally chosen to process the part has been restricted to *Bearing.Cage.Cells*, the *Sourcing.K.S* is able to determine that the bearing cage cell, *First.Cell*, is suitable for manufacturing this part. The entry originally made at the facility level is updated to reflect this decision. A schematic diagram of the facility chosen is shown in Fig. 6.7. The facility consists of two 2SC-lathes and a horizontal machining center.

The determination of the facility to manufacture the part triggers several machining-related concerns. Figure 6.8 shows one of the concerns being indicated to the designer in the Manufacturability Concerns box. The tolerance specified by the designer on the diameter of one of the *Straight.Internal.Diameter* features (dia. = 90.0132 mm) is beyond the capability of the chosen facility. Meeting this tolerance specification would require an extra grinding operation, resulting in an increase in production cost. This concerns has arisen due to the type of machines available in the manufacturing cell. In addition, it is determined that the plug gauges available in the cell would be unable to measure this diameter. However, there is a plug gauge available that can measure a diameter very close to the specified diameter. If this gauge is used, then it would prevent proliferation of new gauges. This

[6.3] 20 pieces/month.

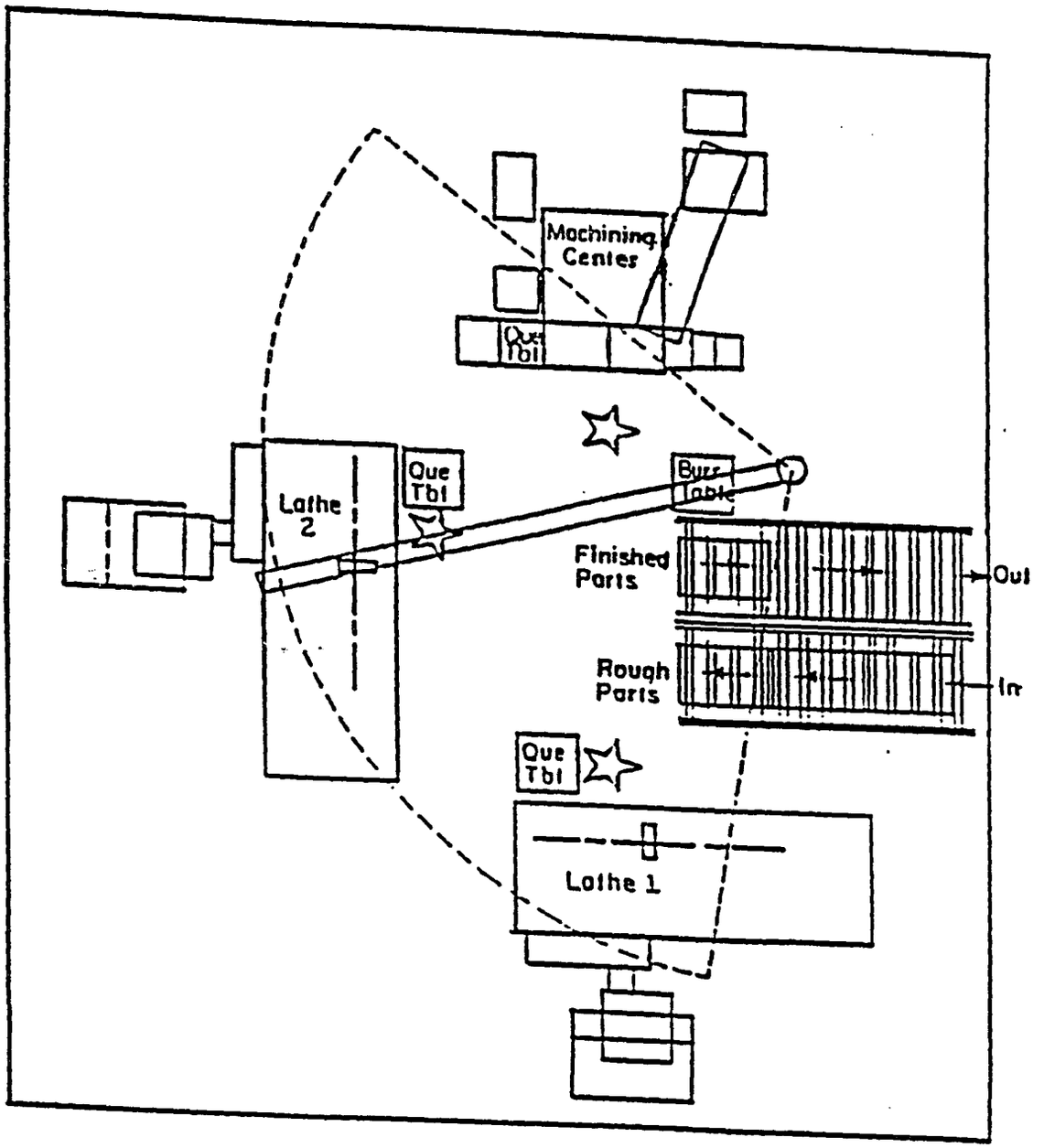

Figure 6.7: Manufacturing Cell

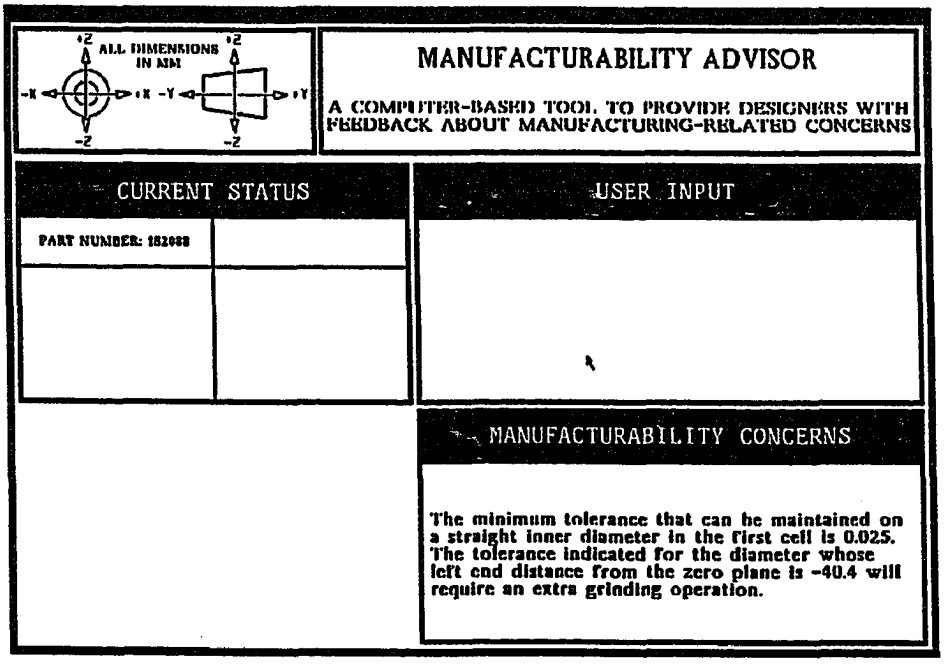

Figure 6.8: Feedback of Machining-Related Concern to Designer

is an example of a concern that has arisen due to the type of gauges available in the cell. This is also an example of a case where the product designer has over-committed himself. The product designer could have avoided this concern by providing a range of allowable values for the diameter instead of a single value. These concerns are examples of the first kind of machining-related concern for which knowledge of how a part is processed through a particular facility is not required.

The determination of the facility to manufacture the part also leads to the third sub-strategy being satisfied. The manager now concentrates on the last sub-strategy, namely "process the part for the facility chosen." The *Machining.K.S* determines that the two Lathes available in the cell can be used to machine some of the form features of the part. Two Machine level entries are created in separate contexts to indicate the choice of machines.

The fixture assigned to Lathe-1 (see Fig. 6.7) is a Chuck and the fixture assigned to Lathe-2 is a Speed-Grip, since they are the only fixtures available for these machines. This results in the creation of Fixture level entries in the individual contexts. In a third context formed by merging the first two contexts, it is determined by the *Machining.K.S* that the part should be processed on Lathe-1 before Lathe-2 because of the type of fixtures asssigned to these two machines.

Based on the part and facility characteristics (redundancy in tooling available on the machines), and the partial process plan developed thus far, a new Strategy entry is created by the manager. The strategy is: determine how to fixture the part on the machines; determine the operations to be performed; and, determine the sequence of operations. The first sub-strategy has two focuses: determine how to fixture the part by using a Speed-Grip on Lathe-2 and determine how to fixture the part by using a Chuck on Lathe-1.

Process-related domain-level problem-solving begins by concentrating on the first focus. Two important decisions are made. The *Straight.Inner.Diameter* feature whose *diameter* is 90.0132 mm is chosen to speed-grip the part on the second lathe because of its diameter, tolerance, and length characteristics. The *Straight.External.Face* feature whose *inner.diameter* is 109.461 mm is chosen to locate the work ring in the second lathe because it has a geometric tolerance that has to be held with respect to the internal feature chosen to speed-grip the part. In addition, these two features are extremely suitable for fixturing the part because they are also datum surfaces. These two decisions are added as attributes of the Fixture level entry made for the Speed-Grip on Lathe-2. The decisions also trigger other control and domain knowledge source instances. In particular, a domain-specific control knowledge source instance is scheduled that creates a Policy level entry. This entry gives greater importance to knowledge sources that make process level decisions regarding the surfaces chosen to speed-grip the part. This change in rating criteria schedules an instance

of *Operation.K.S* for interpretation in the next problem-solving cycle. Macro-Operation entries are created indicating that the features chosen to speed-grip the part on Lathe-2 can be manufactured on Lathe-1.

The determination of the fixturing surfaces for Lathe-2 leads to the current focus being satisfied. The focus changes to determining how to fixture the part on Lathe-1. *Fixturing.K.S* instances are scheduled again, and it is determined that the *Straight.Outer.Diameter* feature whose *diameter* is 171.4 mm diameter and *Straight.External.Face* feature whose *inner.diameter* is 93.73 mm will be used to chuck the part. Once these process design decisions are made, a machining-related concern arises. The sectional thickness of the flange chosen to fixture the part on Lathe-1 is found to be insufficient by the *Machining.Concerns.K.S.* If the part is machined under normal chucking pressures, then the shape of the part could get distorted after the machining is completed. In order to prevent this distortion, chucking pressures, cutting speeds, and feedrates will have to be decreased to machine the part on Lathe-1. However, this leads to a decrease in productivity and requires the operator on the shopfloor to pay greater attention to the operations performed on Lathe-1. This is an example of a machining-related concern for which partial knowledge about process design decisions is required in order to determine if the concern has arisen.

The processing of the part for the chosen facility continues in the manner described. The current focus becomes inoperative, and the strategy changes to determining the operations that will be performed on the part. Other machining-related concerns could arise as more process plan details are developed. The chances of making major design changes to accommodate processing considerations are very high when such concerns are indicated before product design decisions are finalized. Since the fixturing-related concern has arisen at an intermediate stage in the design process, the designer would be willing to increase

the sectional thickness of the flange. For the part design used as an example here,[6.4] the *Straight.External.Face* feature whose *inner.diameter* is 93.73 mm is a cast surface. This surface is not used to mate with any other part in an assembly. The designer, therefore, has some leeway in increasing the sectional thickness of the flange. However, the designer would not be very accommodating after the part design has been completely developed and finalized. This is because any changes made at this stage may have drastic repercussions on the previous design activity and may even nullify some portions of the design. For example, the final part design shown in Fig. 4.6 indicates that bolt clearance holes pass through the flange whose thickness is increased. If the sectional thickness of the flange is increased, then its effect on the length of the bolts would have to be studied (whether it is sufficient or needs to be increased, which will then affect the load carrying capability of the bolts).

All the machining-related concerns identified as a result of creating the new part design can lead to a potential increase in machining cost. The increase in cost must be adequately justified by the designer based on the functional and structural requirements of the part. If the designer can resolve some or all of the concerns by making suitable design changes, he can do so by modifying the previously created design.

6.4.2 Modifying a Part Design

A description of how changes to feature and feature parameters can be incorporated into a part that has already been created is presented. Let us assume that the designer has determined that he can increase the sectional thickness of the flange chosen to fixture the part on Lathe-1. From the menu of options shown in Fig. 6.1, the designer can choose the "Change Parameters" option under the "Modify Part command."[6.5] This choice establishes

[6.4]The part design used here is not hypothetical but is based on a real-world case.
[6.5]By moving the mouse in the direction shown by the arrow.

a new Problem entry on the control blackboard. A domain-specific control knowledge source instance is triggered and scheduled. A new strategy is established by the manager: determine the feature parameter to change and the effect this change has on process design decisions on the domain blackboard.

Due to the first focus, the designer specifies the feature parameter to change through a menu of options available as shown in Fig. 6.9. The designer chooses to change the length of a *Straight.Outer.Diameter* feature (dia= 171.4 mm) from 14 mm to 20 mm. Once this change is made, the current focus is made inoperative. The final step is to determine the effect of this change on process design decisions on the domain blackboard. Process-related decisions are once again generated at various levels on the domain blackboard. These decisions indicate that no changes are necessary in the original process design decisions that were made. Moreover, there are no new machining-related concerns. The goals of the problem entry created are satisfied and the entry is made inoperative on the control blackboard. Any further problem-solving activity is stopped by this action.

If the designer has also determined the need for a groove feature (see Fig. 6.2) to seat an O-ring Seal then the designer can initiate problem solving by choosing the "Replace Feature" option under the "Modify Part" command. The manager's new strategy to solve this problem is: determine the feature to replace; create the replacement feature; and, determine the effect of the replacement on process design decisions made on the domain blackboard.

Domain-level problem-solving begins with the designer specifying the feature to be replaced, in this case a *Straight.Outer.Diameter*. The replacement feature is an instance of *External.Compound.Features* and consists of the following three primitive feature instances: two *Straight.Outer.Diameter* features and a *Groove* feature. The effect of the replacement on

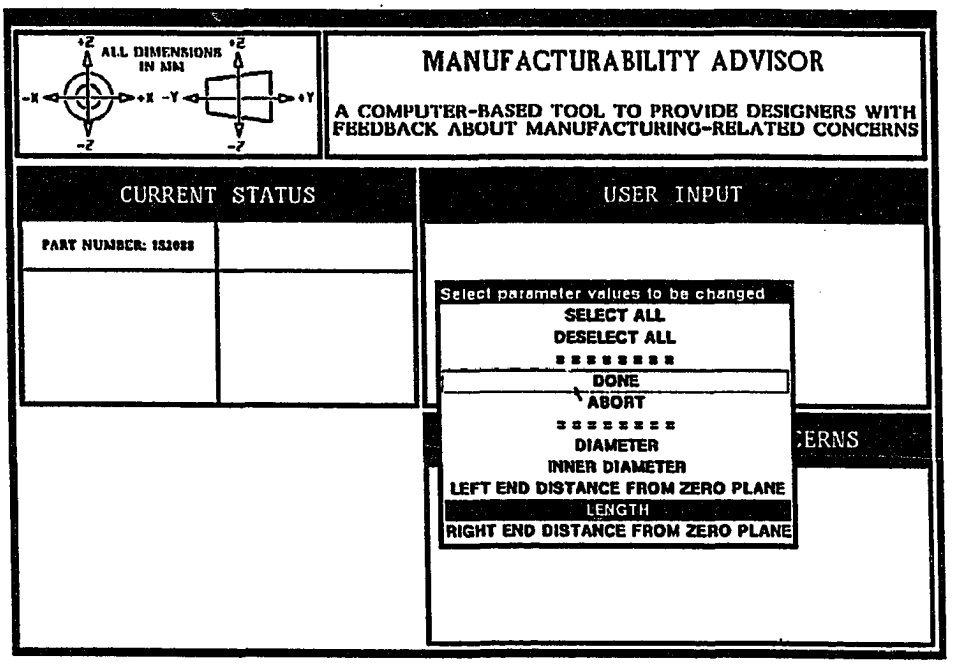

Figure 6.9: Determination of Feature Parameter to Change

process design decisions made on the domain blackboard is then determined. New process-related decision entries are generated once again on the domain blackboard. These decisions indicate that the three new primitive features that have been created can be manufactured on Lathe-1. This leads to the creation of three Macro-Operation entries corresponding to these features. 'Turning' is the operation specified for the outer diameter features and 'Grooving' is the operation specified for the groove feature.

The creation of the Macro-Operation entries triggers an instance of the *Cutting-Tool-.K.S.*. Since the groove is used to seat an O-ring Seal, either a straight-sided or taper-sided groove can be used. However, the *Groove* feature is refined to be a *Straight-Sided.Groove* feature because it can be manufactured more easily than taper-sided grooves. This is another example of the cooperation between the designer and the computer-based knowledge sources. If the designer had over-committed himself and created a taper-sided groove, then this would have led to a machining-related concern. This example shows that by following a least commitment approach, a designer can successfully avoid certain types of concerns. Once the process-related decisions are made, the problem level entry that was created to replace a feature is made inoperative, stopping all problem-solving activity.

6.4.3 Saving and Retrieving a Part Design

Since the designer interacts with the design environment at various stages in the design process, he can save and retrieve parts using these options on the main menu shown in Fig. 6.1. Moreover, to prevent proliferation of parts, a designer need not always create a new part design. He can retrieve a previously created part and make suitable changes (if necessary) to develop the new part design. Saving a particular part design involves saving the current state of the blackboard. Thus, when a designer retrieves a part design at any stage in the design process, he also automatically retrieves all the process-related decisions

that have been made for that particular design.

6.5 Demonstration of Process Performance Model

The main intent of the demonstration described in this section is to show how the generative process performance model can be used to provide feedback to designers about certain types of machining-related concerns. Since process performance models were not available for the bearing cages domain, the following demonstration has been developed independently in a different domain and is not linked to the portion of the design environment described in the previous section.

The process performance model used as an example has been developed for "turning" thin-walled cylinders [93]. The workpiece used as an example in this demonstration is a hollow, open-ended, thin-walled cylinder. It is made of 1018 steel and has an average diameter of 12 in. and an overall length of 32 in. The initial wall thickness is 0.25 in. and the final wall thickness is 0.15 in. The designer's specifications for the two performance parameters, surface error and surface roughness, are that they be less than 0.001 in. and 50 μin., respectively. The following demonstration shows the use of the process performance model to determine the downstream (manufacturing) effect of the designers specifications for these parameters. The manufacturing related performance parameters chosen are material removal rate and tool-life.

Since 0.10 in. of material must be removed and the final surface error has to be less than 0.001 in., at least two passes will be required to perform the turning operation [92]. The main objective of the first pass is to meet the aspiration levels for only three performance parameters, namely material removal rate, tool-life, and surface error. Surface roughness is not critical for this pass and can be specified as a constraint rather than as an objective

Table 6.1: First Pass- First Iteration

AL= (MRR= 6.0, TOOLLF= 50, SE= 0.002)

S		x			f(x)			
	SPEED	FEED $\times 10^{-3}$	DOC $\times 10^{-2}$	MRR	TOOLLF	SE $\times 10^{-3}$	SR ≤ 200	POWER ≤ 8
S_1= 3.55	420.4	13.8	8.0	5.6	40.8	2.5	191	4.6
S_{11}= 2.68	437.7	14.3	8.0	6.0	39.0	2.8	200	4.9
S_{12}= 2.65	334.5	14.3	8.0	4.6	50.0	2.7	200	3.8
S_{13}= 2.88	600.0	7.0	8.0	3.9	37.2	2.0	47	3.9

function. The initial aspiration levels for the various objective functions for this pass are material removal rate of at least 6 $in.^3$, tool-life of at least 50 minutes, and surface error no greater than 0.002 in. Surface roughness and total power consumption are constrained to be no greater than 200 μin. and 8 HP, respectively. The results of the first iteration for the primary and auxiliary problems described in Chapter 4 are shown in Table 6.1.

A number of interesting observations can be made based on the results shown in Table 6.1. The output for the primary problem (row 1) indicates that the aspiration levels on the performance parameters chosen as objective functions for the first pass cannot be met simultaneously. The results for the third auxiliary problem (row 4) indicate that material removal rate and tool-life decrease drastically if an aspiration level of 0.002 in. on surface error must be attained. The results in the columns for material removal rate and tool-life indicate that if material removal rate is increased, then tool-life decreases and vice-versa. Lastly, the two constraints indicate that the constraint on power is easily achieved, but the surface roughness values for the first three problems is set close to its upper limit of 200 μin.

The first observation indicates that a trade-off must be made based on the relative

Table 6.2: First Pass– Second Iteration

AL= (MRR= 6.0, TOOLLF= 50, SE= 0.003)

S		x			f(x)			
	SPEED	FEED $\times 10^{-3}$	DOC $\times 10^{-2}$	MRR	TOOLLF	SE $\times 10^{-3}$	SR ≤ 250	POWER ≤ 8
S_1= 3.15	376.4	16.2	8.0	5.9	42.9	2.9	250	4.6
S_{11}= 2.18	383.5	16.3	8.0	6.0	42.2	3.0	250	4.7
S_{12}= 2.20	319.2	16.3	8.0	5.0	50.0	3.0	250	3.9
S_{13}= 2.18	365.4	16.3	8.0	5.7	44.1	3.0	250	4.5

importance of the three objective functions. Since this is not the final operation, it is more critical to meet the aspiration levels on material removal rate and tool-life than to meet the aspiration level on surface error. Thus, for the next iteration, the aspiration level for surface error is relaxed[6.6] to be no greater than 0.003 in. The upper limit on surface roughness is also relaxed to be 250 μin; since this is not the final pass. These changes are incorporated in the second iteration and the results of this iteration are shown in Table 6.2.

The end result of the second set of iterations is that the aspiration level on surface error can now be met, but the aspiration levels set for material removal rate and tool-life cannot be met simultaneously. A suitable trade-off is required between material removal rate and tool-life. Surface error can now be removed as an objective function and included as a constraint with an upper limit of 0.003 in. Material removal rate is chosen as being more critical than tool-life and the aspiration level on tool-life is lowered for the next iteration to 45 minutes. Two more iterations are needed before the final results for this pass are achieved (see Table 6.3). This table indicates that for the first pass, cutting speed is 383.4 feet per minute, feedrate is .0163 inches per revolution, and depth of cut is 0.08 in. The

[6.6]Currently by the user, but see [92].

Table 6.3: First Pass- Last Iteration

AL= (MRR= 6.0, TOOLLF= 42)

S		x			f(x)			
	SPEED	FEED $\times 10^{-3}$	DOC $\times 10^{-2}$	MRR	TOOLLF	SE $\times 10^{-3}$	SR ≤ 250	POWER ≤ 8
S_1= 1.99	383.4	16.3	8.0	6.0	42.2	3.0	250	4.7
S_{11}= 0.99	383.4	16.3	8.0	6.0	42.2	3.0	250	4.7
S_{12}= 0.99	383.4	16.3	8.0	6.0	42.2	3.0	250	4.7

Table 6.4: Second Pass- First Iteration

AL= (MRR= 1.0, TOOLLF= 100, SE= 0.001, SR= 50)

S		x			f(x)			
	SPEED	FEED $\times 10^{-3}$	DOC $\times 10^{-2}$	MRR	TOOLLF	SE $\times 10^{-3}$	SR	POWER ≤ 8
S_1= 4.70	437.6	4.8	2.0	0.51	101.5	1.3	24	0.7
S_{11}= 7.71	421.1	15.2	1.9	1.46	74.0	1.9	223	1.6
S_{12}= 3.70	445.5	4.8	2.0	0.52	100.0	1.3	24	0.7
S_{13}= 4.90	300.0	3.5	2.0	0.25	161.4	1.0	12	0.4
S_{13}= 4.22	445.5	4.8	2.0	0.52	100.0	1.3	24	0.7

values of the performance parameters for this setting are: material removal rate 6.0 $in.^3$, tool-life 42.2 minutes, surface error 0.003 in., and surface roughness 250 μin.

In the second pass, 0.02 in. of material will be removed, and all the performance parameters, material removal rate, tool-life, surface error, and surface roughness, are chosen as objective functions with suitable aspiration levels. The results of the first iteration for this pass are shown in Table 6.4.

The results indicate that the aspiration levels on tool-life and surface roughness can be easily met. However, there is a large trade-off between material removal rate and surface

Table 6.5: Second Pass– Last Iteration

AL= (MRR= 0.25, SE= 0.001)

S		x			f(x)			
	SPEED	FEED $\times 10^{-3}$	DOC $\times 10^{-2}$	MRR	SE $\times 10^{-3}$	TOOLLF ≥ 100	SR ≤ 50	POWER ≤ 8
S_1= 2.0	300.0	3.5	2.0	0.25	1.0	161.4	12	0.4
S_{11}= 1.0	300.0	3.5	2.0	0.25	1.0	161.4	12	0.4
S_{12}= 1.0	300.0	3.5	2.0	0.25	1.0	161.4	12	0.4

error. The results of the second auxiliary problem indicate that the surface error is close to 0.002 in. if the requirement on material removal rate of 1 $in.^3$ must be attained. Since this is the final operation, the designer's requirement placed on surface error is critical. Therefore, the aspiration level for material removal rate must be relaxed. In successive iterations, tool-life and surface roughness are included as constraints and the aspiration level on material removal rate is relaxed. The results from the final iteration for this pass are shown in Table 6.5.

The results in Table 6.5 indicate that for the second pass, the cutting speed is 300.0 feet per minute, the feedrate is .0035 inches per revolution, and depth of cut is 0.02 in. The values of the performance parameters for this setting are material removal rate 0.25 $in.^3$, tool-life 161.4 minutes, surface error 0.001 in., and surface roughness 12 μin.

The results of this pass demonstrate the consequences of satisfying the designer's requirements on surface error and surface roughness. Although the surface roughness requirement is easily satisfied, the stringent requirement placed on surface error leads to a significant decrease in the material removal rate (from 1 $in.^3$ to 0.25 $in.^3$). The end result of this is a decrease in productivity. However, by conducting such an analysis (for critical operations) and providing feedback to designers about any machining-related concerns that arise, one

can enable them to reconsider certain part specifications that have already been determined.

6.6 Summary

An implementation of the design environment for the concurrent product and process design of bearing cages has been described in detail. It was shown that the product and process design domain knowledge sources can cooperatively assist each other in developing a manufacturable product design. Representative examples of two different kinds of machining-related concerns were presented. The need for providing designer's feedback about machining-related concerns as early as possible in the product development process was demonstrated. The importance of the "generative" process performance model for providing feedback to designers about certain types of machining-related concerns was established with the help of a suitable example.

Chapter 7

CONCLUSIONS AND RECOMMENDATIONS

The Simultaneous Engineering concept for components manufactured in small and medium lot-sizes is the focus of this research. The best approach to this concept is to achieve this concept in two stages: the first stage involves the design of special low cost manufacturing cells based on existing part designs, and the second stage involves the development of a computer-based design environment to "ensure manufacturability" of new part designs in these specially designed facilities. This approach has been validated mainly by implementing it for the concurrent product and process design of Bearing Cages. An estimated 45%[7.1] savings in manufacturing cost for Bearing Cages alone [99] signifies the potential of the proposed methodology.

Several important observations were made while developing the design environment to "ensure manufacturability." These observations are discussed in detail in Section 7.1. The main contributions of this research are summarized in Section 7.2. Section 7.3 provides recommendations for future research work.

[7.1] Approximately $3 million.

7.1 Observations

7.1.1 General Characteristics of the Design Environment

The general characteristics that make the design environment convenient to use, modify, and extend are discussed in detail.

A more convenient and natural user-interface for the designer, where he is in control of the product design process, is one of the characteristics of the design environment developed in this research. This is in contrast to the interface for typical rule-based AI systems in which knowledge about the structure/function of the problem domain is not explicit but "compiled" into the rules of the reasoning system. For example, Fig. 7.1 shows a portion of an interactive session from a rule-based process planning system developed for tube assemblies [100]. Knowledge of tube assembly features such as "Beads", "Plugs," and "Powdered Metal Fitting" has not been separated from process planning knowledge such as "a special pickler cycle without acid bath is required for powdered metal fitting." Due to this, instead of the user specifying the relevant features of a part, the system prompts the user to indicate the existence (or non-existence) of certain structural or functional part features. For the design environment developed in this research, this type of interaction would not "naturally correspond" to the thought process flow of the designer. However, the development of an explicit product model and its separation from the reasoning system in the design environment provide the requisite flexibility for the development of a more appropriate interface. The interface developed gives the designer (instead of the computer-based system) complete control of the design task while he is creating the product model.

Under the current structure of the design environment, the development of the product model is entirely the user's responsibility. An added advantage of having an explicit product model is that the existing interface can be replaced by a "design automation" system that

```
.
.
Will extra tube be needed on the other end for the tube to be processed ?
   1. yes
   2. no
>> 2.
Is there a powdered metal fitting on either end of the tube ?
   1. yes
   2. no
>> why
If there is a powdered metal fitting at either end of the tube then a
special pickler cycle (operation 300) without an acid bath is required.
Is there a powdered metal fitting on either end of the tube ?
   1. yes
   2. no
>> 1.
Does the tube require one or more BEADS at either end ?
   1. yes
   2. no
>> 2.
Is a plug needed for a fitting at either end of the tube ?
   1. yes
   2. no
>> 2.
Is there a note on the drawing which states that the tube must withstand a
pressure test of greater than or equal to 21000 kpa ?
   1. yes
   2. no
>> 1.
Checking tooling availability:
Is a high pressure adapter available for each end of this tube ?
   1. yes
   2. no
>> 1.
.
.
.
```

Figure 7.1: Sample Interactive Session with a Typical Rule-Based System

assists the user in performing the actual design task. In this case, the user interacts with the "design automation" system, and the output from this system is the relevant product model that meets design specifications. Use of the product model as a shared medium of communication between the design and manufacturing-related tasks is an important aspect of this research. This makes information about a part accessible and prevents duplication of effort in determing part characteristics.

Another advantage of the explicit product and manufacturing facility models of the design environment is the extensive "vocabulary" that the models provide for reasoning about the structure/function of the problem-solving domain. The product and manufacturing facility features described in detail in Chapter 4 (for developing the feature-based model) are one example of this characteristic of the design environment. The same advantage holds true for the multi-level structure of the blackboard (described in detail in Chapter 5). The models and the blackboard structure are primarily responsible for the fine-grained representation of knowledge in the reasoning subsystem, allowing subtle distinctions to be made and leading to fewer quantum leaps of inference. This aspect is most evident while developing a partial process plan to determine if any machining-related concerns have arisen.

During the course of developing the design environment, it was realized that important parallels existed between the approach proposed in this thesis and a "parametric programming" approach to developing such types of systems using conventional programming languages. An implementation of the parametric programming concept for the design of "excavator buckets" [101] was studied in detail, and feedback was obtained from the system developer about the advantages and drawbacks of such a system. The following two interesting remarks were made by the system developer:

1. The computer-based system for designing "excavator buckets" has been carefully de-

veloped and is highly modular, yet "keeping the program current" is a very tedious job. For example, sometimes when a variable value is changed, one has to check its occurrence everywhere in the program to determine the effect of this change.

2. Before the system is developed, one has to know how much flexibility should be incorporated in the system. This flexibility cannot be incrementally added to the system. However, this is a difficult requirement to meet because product designers cannot clearly specify at the beginning "what they want the system to do."

These remarks indicate that the design environment developed in this research should satisfy two important metrics: ***maintainability*** and ***extensibility.***

The use of Frame- and Constraint-Based Systems to develop the product and manufacturing facility models is one way in which the design environment satisfies the two metrics mentioned above. Extensions and modifications to the models are easier to make because structure, function, and relation are embodied in discrete frame objects (features) that are easy to manage and conceptualize. For example, if the form features "teeth" (e.g., spur and helical gear teeth), "thread," or "knurl" have to be created, then the inheritance hierarchy of the frame-based system underlying the feature-based model can be suitably extended to include these features. These features can be added by growing the inheritance hierarchy rooted under *Concentric.Features.* Constraint-Based Systems also contribute to a flexible and modular approach to developing the design environment by the use of Object Oriented Values (see Chapter 4).

Use of the MCKS paradigm for developing the reasoning subsystem is another way in which the design environment satisfies the two metrics. The MCKS system allows overlap or redundancy between various domain knowledge sources which need not be explicitly removed at system development time. Any conflict between knowledge sources can be

resolved at runtime through the development of competing "islands of solutions" on the blackboard (see Chapter 5).

Explicit representation of control and domain knowledge under the MCKS paradigm is another aspect of the design environment that allows it to be extended for use in different domains. While developing the knowledge sources to "ensure manufacturability" of Bearing Cage designs, it was noted that the knowledge possessed by the computer-based domain knowledge sources depend mainly upon the type of fixtures, cutting tools, machines, etc., available in a particular facility.[7.2] However, the control knowledge possessed by the manager was determined to be more facility specific and dependent upon the structure and layout of the facility. One of the advantages of the explicit representation of the two kinds of knowledge became evident when the process planner indicated that the manufacturing cells developed for making Pulleys were identical to the cells for Bearing Cages except for the absence of Speed-Grip fixtures.[7.3] Since no new components have been added to the Pulley facilities, there will be no changes in the domain knowledge (provided it is complete). However, additions to control knowledge would be necessary since the structure of the facilities has changed. The explicit representation of control knowledge makes it easier to add new knowledge at this level and extend the design environment to "ensure manufacturability" of pulley designs. At present this is only an observation; it has not been actually implemented and tested.

[7.2] The dependence of the machining-related concerns on the components of the facility has already been indicated in Chapter 3.

[7.3] All the fixtures in these manufacturing cells were Chucks.

7.1.2 Impact of Design Environment on Product Designer and Process Planner

The main impact of the design environment on the product designer is that it enables him to consider manufacturing concerns sufficiently early in the design task. Product design is still carried out in the same manner, except that by taking (wherever possible) a least commitment approach to design, the designer can try to decrease the number of machining-related concerns that arise. The design environment supports the least commitment approach in two ways:

1. The product feature inheritance hierarchies enable a selective abstraction process whereby portions of the part that are more abstract than others can be described by instances of features that are closer to the root of the inheritance hierarchies.

2. The Constraint-Based System with its capability to propagate interval values allows feature attributes to be intervals rather than a specific value.

The impact of the design environment has been greater on the process planner's task, primarily because his task has been moved "upstream" and is now considered concurrently with the product design task [102]. The development of the design environment has restructured the role of the process planner but has not completely eliminated him from the product development process. Due to the availability of this design environment, personal or computer-based interaction between product designers and process planners is not mandatory. The lead time required to ensure manufacturability of product designs is reduced, and process planners are required to develop only the detailed plan to process a part in a particular facility. Restructuring of the process planning activity has, however, introduced several new responsibilities for process planners. In the following sections, these new responsibilities for process planners are presented. The significant role of the design

environment in improving the productivity of the process planners while they carry out these responsibilities is emphasized.

Designing Special Manufacturing Facilities

The design of the special manufacturing facilities is an important and critical task for achieving the maximum reduction in cost under the Simultaneous Engineering concept for components manufactured in small and medium lot-sizes. 'Knowledgeable' process planners, because of their familiarity with the capabilities of the existing manufacturing cells, have the added responsibility of successfully performing this task. The availability of the design environment reduces the burden on process planners and enables them to concentrate on this crucial task.

Updating the Design Environment

Due to the dynamics of the manufacturing arena, another responsibility of the process planners is to constantly update the design environment to reflect the latest capabilities and machining-related concerns of the manufacturing facilities. Use of the model-based reasoning approach and the MCKS paradigm for developing the design environment is particularly beneficial for this task.

As mentioned before, explicit representation of control and domain knowledge makes it easy for process planners to incorporate new knowledge in the design environment. A careful separation of process-related domain knowledge from structural and functional knowledge about a manufacturing facility in model-based reasoning systems makes it easy for the process planner to update the design environment by changing only the manufacturing facility model. For example, capabilities of a manufacturing cell to maintain certain tolerances can change over time: this is a functional characteristic of the cell. However, it is always true

that parts with tolerances below the cell capability cannot be processed in the cell: this is process-related domain knowledge. Thus, as functional characteristics of the facility change, the process planner has to modify only the manufacturing facility model and does not have to change the domain knowledge.

Minimizing Total Number of Out-Sourced Parts

The Simultaneous Engineering concept for components manufactured in small and medium lot-sizes is achieved in two stages: it is not 'truly' simultaneous. This is bound to lead to sub-optimal solutions for certain kinds of parts. Part designs that cannot be made in any of the specially designed facilities and have to be out-sourced at a higher product development cost will continue to be developed. Under the new product development practice, another responsibility for the process planner is to keep the number of such out-sourced parts to a minimum. By constantly monitoring the type of parts that are out-sourced, process planners (because of their knowledge of the capabilities of the facility) have to suitably decide whether it is worthwhile to design new facilities or alter existing facilities to accommodate such components in the future. This is a very important task in ensuring the success of the two-stage approach to the Simultaneous Engineering concept. This is also a very time-consuming task and a large portion of the process planners time must be devoted to it. The development of the design environment is critical in enabling the process planner to perform this task more efficiently.

7.1.3 Knowledge Acquisition Bottleneck

AI-based systems have normally been used in the engineering domain to obtain productivity improvements by aiding or automating a particular task. This research demonstrates that the application of AI in such domains can be further extended to achieve still higher pro-

ductivity by using AI-based systems to efficiently restructure the manner in which certain tasks are performed. For example, as described in the previous section, the availability of the AI-based design environment resulted in a restructuring of the tasks performed by the process planner and enabled him to achieve significant productivity improvements. However, the development of the design environment introduces a significant challenge for the knowledge acquisition process.

The process of acquiring knowledge is much more complicated for an AI-based system developed to assist a restructured task. The main reason is the lack of domain experts performing the task in the manner dictated by the restructuring process. As a result, a knowledge engineer must develop a much greater understanding of the domain and must "extend" the available knowledge to make it applicable after the restructuring process. For example, development of the design environment requires that portions of the process planning task begin (if necessary) before the part design has been finalized. Process planning is not performed under such circumstances and, therefore, no suitable domain experts are available. The knowledge possessed by process planners must be extended (by the knowledge engineer with the help of product and process designers) to be applicable under the new context. The knowledge required to make part refinements that preclude certain machining-related concerns is an example of knowledge that has been generated this way.

7.2 Conclusions

A problem-driven approach was taken in this research to conduct a detailed study of the Simultaneous Engineering concept for components manufactured in small and medium lot-sizes. In the course of this research several problems were addressed and the solutions proposed for these problems enabled one to get a better understanding of the Simultaneous

Engineering concept. An important conclusion of this research is that the nature and scope of the Simultaneous Engineering concept mandates the use of such a rigorous problem-driven approach in different domains to fully achieve the goals of this concept. The main contributions of the study undertaken in this research can be briefly summarized as follows:

1. A cost-effective, two-stage approach has been developed to implement the Simultaneous Engineering concept for components manufactured in small and medium lot-sizes. It is important to note that both stages of the proposed approach are mandatory for obtaining the maximum reduction in product development cost.

2. A systematic procedure to identify and classify machining-related concerns has been established. This procedure significantly reduces the time required to delineate machining-related concerns by interviewing product designers and process planners.

3. The need for developing a computer-based design environment to assist the product designer in developing manufacturable product designs has been established. Model-Based Reasoning Systems were shown to provide a domain-independent framework for such a design environment.

4. The need for multiple product and manufacturing facility models to provide the most appropriate feedback to the product designer about machining-related concerns has been demonstrated. A feature-based model is required because "features" play an important role in the reasoning process involved while carrying out the product and process design tasks. A geometric model is necessary in order to determine the macroscopic effects of facility components[7.4] that cannot be determined by using only the feature-based model. A process performance model is necessary because they augment

[7.4] Fixtures, machines, cutting-tools, etc. (see Chapter 4).

the qualitative description of certain types of machining-related concerns to provide the designer a more quantitative description of the concern.

5. The need for explicitly reasoning about the concurrency between the product design and process planning tasks has been demonstrated. MCKS systems were shown to satisfy this requirement of the design environment. It was shown that the explicit product and manufacturing facility models and the MCKS system contributed to the modularity, flexibility, maintainability, and extensibility characteristics of the design environment.

6. The importance of using AI-based systems to complement rather than replace the activities of human experts in engineering applications was demonstrated. It was shown that the development of the AI-based design environment leads to an efficient restructuring of the process planning task and enables significant improvements in the productivity of the process planner.

7.3 Recommendations

Simultaneous Engineering is a fairly recent research topic. Many useful contributions have been made by this research in improving the understanding of this concept. However, enormous potential exists for further research. Ultimately, a more comprehensive framework for a Computer-Aided Simultaneous Engineering system needs to be developed. Figure 7.2 schematically represents one possible framework for such a system that could be developed through suitable extensions to this research. As described below, such a system can be developed by conducting future research in several different areas:

- The modularity, flexibility, maintainability, and extensibility characteristics of the design environment have been studied by implementing it for the concurrent product and process design of Bearing Cages. Future research should be conducted to study these characteristics of the design environment in other problem-domains such as Pulleys, Gears, or Cylinder Linings.

- An important limitation of the design environment is that it can provide feedback to the designer only about machining-related concerns; it cannot assist him in resolving the conflicts that arise between different product life-cycle concerns. Future extensions to this research could broaden the scope of product life-cycle concerns considered and provide assistance in conflict resolution by capturing "conflict resolution knowledge" [103].

- The development of the design environment has clearly established the important role of the process performance model in providing feedback to designers about machining-related concerns. This model complements the experience-based heuristics developed by domain experts. Currently the generative process performance model has not been integrated into the design environment but has been developed independently. Future work should consider the development of the *Cutting Parameters* K.S. to incorporate this model. Future research should also continue to further delineate the role of such models in the design environment and study their usage in conjunction with numerical optimization and machine learning techniques [104].

- It was shown in this research that the design environment enables the product designer to take the least commitment approach in order to minimize the number of machining-related concerns that arise. The drawback is that the computer-based knowledge sources may not possess the knowledge to take advantage of a designer's

least commitment strategy. Thus, if the strategy is not judiciously used, it could lead to an unnecessary increase in the total product development time. No detailed study has been conducted to assist the designer in using this strategy and enable him to minimize any unnecessary increase in product development time. Developing suitable means to do so would be an appropriate research direction for future extensions to this research.

- Given the efficacy of restructuring the process planning task, future extensions to this research should address the issue of developing knowledge acquisition tools and techniques that can ease the task of acquiring knowledge for the design environment. An important aspect of this work would be to develop suitable means of ensuring "reasonable completeness" for knowledge generated to be applicable after the restructuring process. Future research work should also focus on capturing product development rationale [105] to circumvent or alleviate the knowledge acquisition bottleneck described above. Such systems would require an explicit representation of product and manufacturing facility features, a requirement easily satisfied by the feature-based model.

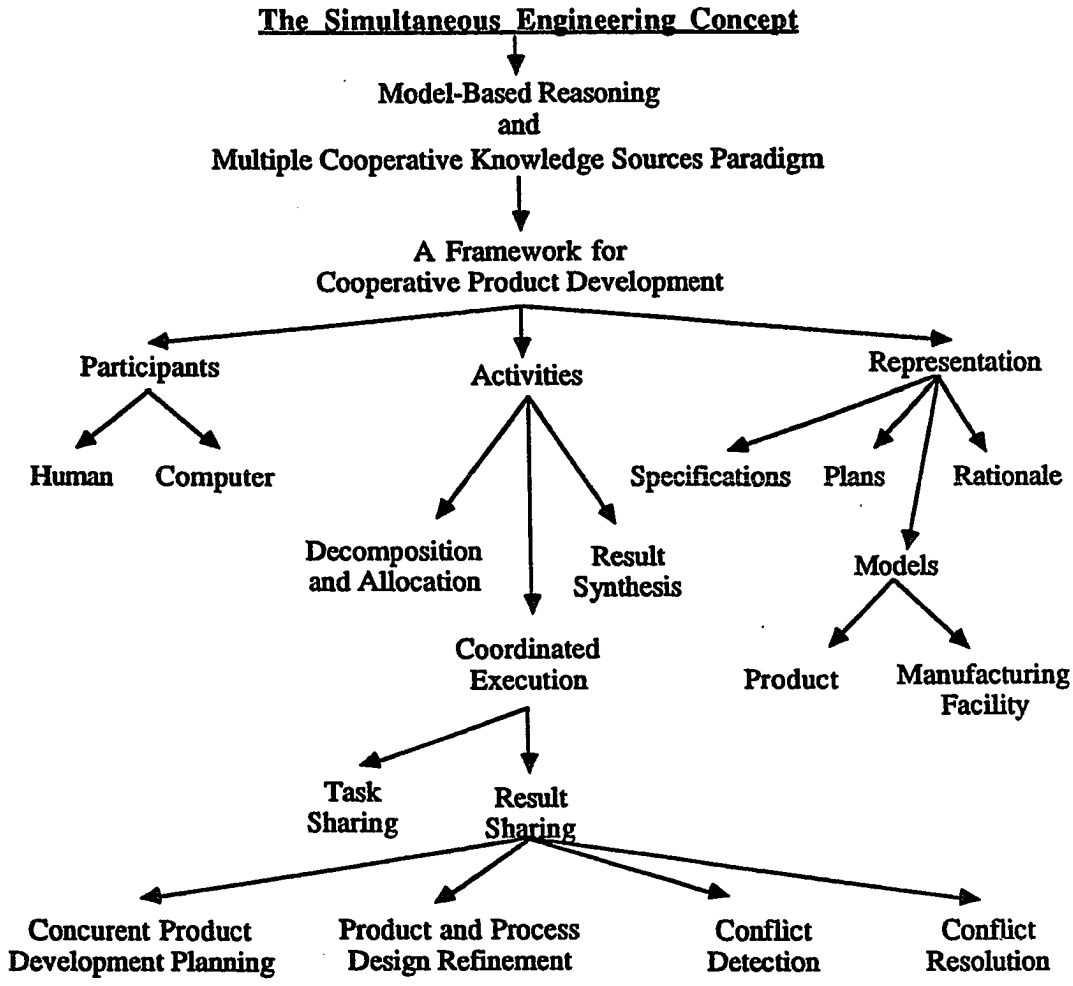

Figure 7.2: Extended Computer-Aided Simultaneous Engineering System

APPENDICES

Appendix A

PARAMETRIC DESCRIPTIONS OF FEATURES

A.1 Product Features

```
Feature:  OUTER.DIAMETERS

  Superclasses:          EXTERNAL.CONCENTRIC.FEATURES
  Member of:             CLASSES
  Subclasses:            STRAIGHT.OUTER.DIAMETERS
                         POSITIVE.NORMAL.OUTER.DIAMETERS
                         NEGATIVE.NORMAL.OUTER.DIAMETERS

  Member slot:           CLOSE.PRIMITIVE.FEATURE.DISPLAY
  Slot Characteristic:   USER-INTERFACE.METHOD
  Valueclass:            METHOD
  Values:                PRODUCT-LIBRARY>CONCENTRIC.FEATURES:
                         CLOSE.PRIMITIVE.FEATURE.DISPLAY!method

  Member slot:           COMPOUND.FEATURE
  Slot Characteristic:   BOOK-KEEPING.PARAMETER
  Valueclass:            EXTERNAL.COMPOUND.FEATURES
  Values:                Unknown

  Member slot:           CREATE.NEW.INSTANCE
  Slot Characteristic:   CREATION.METHOD
  Valueclass:            METHOD
  Values:                PRODUCT-LIBRARY>CONCENTRIC.FEATURES:
                         CREATE.NEW.INSTANCE!method

  Member slot:           CSG.NODE
  Slot Characteristic:   GEOMETRIC-MODEL.PARAMETER
```

```
Values:                 Unknown

Member slot:            DELETE.FEATURE.TO.THE.LEFT
Slot Characteristic:    DELETION.METHOD
Valueclass:             METHOD
Values:                 DELETE.FEATURE.TO.THE.LEFT

Member slot:            DELETE.FEATURE.TO.THE.RIGHT
Slot Characteristic:    DELETION.METHOD
Valueclass:             METHOD
Values:                 DELETE.FEATURE.TO.THE.RIGHT

Member slot:            DESIGN.ENVIRONMENT.PICTURE
Slot Characteristic:    USER-INTERFACE.PARAMETER
Valueclass:             KEEPICTURE.OBJECTS
Values:                 PRIMITIVE.FEATURES.PICTURE

Member slot:            DIAMETER
Slot Characteristic:    FEATURE.PARAMETER
Values:                 Unknown

Member slot:            ESTABLISH.SPECIAL.RELATIONSHIPS
Slot Characteristic:    CREATION.METHOD
Valueclass:             METHOD
Values:                 CONCENTRIC.ESTABLISH.SPECIAL.RELATIONSHIPS

Member slot:            EXTERNAL.DEPRESSIONS
Slot Characteristic:    BOOK-KEEPING.PARAMETER
Valueclass:             EXTERNAL.DEPRESSIONS
Values:                 Unknown

Member slot:            EXTERNAL.PROTRUSIONS
Slot Characteristic:    BOOK-KEEPING.PARAMETER
Valueclass:             EXTERNAL.PROTRUSIONS
Values:                 Unknown

Member slot:            FEATURE.NAME.STRING
Slot Characteristic:    USER-INTERFACE.PARAMETER
Values:                 "Outer Diameter"

Member slot:            FEATURE.TO.THE.LEFT
Slot Characteristic:    FEATURE.RELATION.PARAMETER
Valueclass:             (LIST.OF (UNION CONCENTRIC.FEATURES
                                        COMPOUND.FEATURES))
Values:                 Unknown

Member slot:            FEATURE.TO.THE.RIGHT
Slot Characteristic:    FEATURE.RELATION.PARAMETER
Valueclass:             (LIST.OF (UNION CONCENTRIC.FEATURES
```

```
                                        COMPOUND.FEATURES))
Values:                 Unknown

Member slot:            FILL.PARAMETERS.VALUE
Slot Characteristic:    CREATION.METHOD
Valueclass:             METHOD
Values:                 P-F.FILL.PARAMETERS.VALUE

Member slot:            FIND.PARAMETERS.TO.CHANGE
Slot Characteristic:    ACCESS.METHOD
Valueclass:             METHOD
Values:                 FIND.PARAMETERS.TO.CHANGE

Member slot:            GENERATE.CSG.NODE
Slot Characteristic:    GEOMETRIC-MODEL.PARAMETER
Valueclass:             METHOD
Values:                 CONCENTRIC.CSGNODE

Member slot:            GENERATE.INSTANCE.NAME
Slot Characteristic:    CREATION.METHOD
Valueclass:             METHOD
Values:                 GENERATE.INSTANCE.NAME

Member slot:            GEOMETRIC.TOLERANCES
Slot Characteristic:    SPECIAL.FEATURE.PARAMETER
Valueclass:             GEOMETRIC.TOLERANCES
Values:                 Unknown

Member slot:            GET.FEATURE.INSTANCE
Slot Characteristic:    FEATURE.INFORMATION.ACCESS.METHOD
Valueclass:             METHOD
Values:                 CONCENTRIC.GET.FEATURE.INSTANCE

Member slot:            IF.L.H.S.FEATURE
Slot Characteristic:    FEATURE.RELATION.PARAMETER
Values:                 RIGHT.END.DISTANCE

Member slot:            IF.R.H.S.FEATURE
Slot Characteristic:    FEATURE.RELATION.PARAMETER
Values:                 LEFT.END.DISTANCE

Member slot:            INNER.DIAMETER
Slot Characteristic:    FEATURE.PARAMETER
Values:                 Unknown

Member slot:            INSERT.NEW.FEATURE
Slot Characteristic:    REPLACEMENT.METHOD
Valueclass:             METHOD
Values:                 CONCENTRIC.INSERT.NEW.FEATURE
```

```
Member slot:          INSTANCE.COUNTER
Slot Characteristic:  BOOK-KEEPING.PARAMETER
Valueclass:           INTEGER
Values:               0

Member slot:          INTERNAL.DEPRESSIONS
Slot Characteristic:  BOOK-KEEPING.PARAMETER
Valueclass:           INTERNAL.DEPRESSIONS
Values:               Unknown

Member slot:          INTERNAL.PROTRUSIONS
Slot Characteristic:  BOOK-KEEPING.PARAMETER
Valueclass:           INTERNAL.PROTRUSIONS
Values:               Unknown

Member slot:          KIND.OF.PART
Slot Characteristic:  BOOK-KEEPING.PARAMETER
Valueclass:           KIND.OF.PARTS
Values:               Unknown

Member slot:          LEFT.END.DISTANCE
Slot Characteristic:  FEATURE.PARAMETER
Values:               Unknown

Member slot:          LEFT.FEATURE.CONSTRAINT
Slot Characteristic:  CONSTRAINT.METHOD
Valueclass:           METHOD
Values:               LEFT.FEATURE.CONSTRAINT

Member slot:          LENGTH
Slot Characteristic:  FEATURE.PARAMETER
Values:               Unknown

Member slot:          MACHINING.REQUIRED
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           (ONE.OF YES NO Y N)
Values:               Unknown

Member slot:          NEW.INSTANCE.NAME
Slot Characteristic:  BOOK-KEEPING.PARAMETER
Values:               OUTER.DIAMETERS-1

Member slot:          OPEN.PRIMITIVE.FEATURE.DISPLAY
Slot Characteristic:  USER-INTERFACE.METHOD
Valueclass:           METHOD
Values:               PRODUCT-LIBRARY>CONCENTRIC.FEATURES:
                      OPEN.PRIMITIVE.FEATURE.DISPLAY!method
```

```
Member slot:          REPLACE.FEATURE
Slot Characteristic:  REPLACEMENT.METHOD
Valueclass:           METHOD
Values:               CONCENTRIC.REPLACE.FEATURE

Member slot:          REPLACE.PARAMETER.VALUE
Slot Characteristic:  REPLACEMENT.METHOD
Valueclass:           METHOD
Values:               REPLACE.PARAMETER.VALUE

Member slot:          RIGHT.END.DISTANCE
Slot Characteristic:  FEATURE.PARAMETER
Values:               Unknown

Member slot:          RIGHT.FEATURE.CONSTRAINT
Slot Characteristic:  CONSTRAINT.METHOD
Valueclass:           METHOD
Values:               RIGHT.FEATURE.CONSTRAINT

Member slot:          RIGID.MOTION.PITCH
Slot Characteristic:  GEOMETRIC-MODEL.PARAMETER
Values:               Unknown

Member slot:          RIGID.MOTION.ROLL
Slot Characteristic:  GEOMETRIC-MODEL.PARAMETER
Values:               Unknown

Member slot:          RIGID.MOTION.X
Slot Characteristic:  GEOMETRIC-MODEL.PARAMETER
Values:               Unknown

Member slot:          RIGID.MOTION.Y
Slot Characteristic:  GEOMETRIC-MODEL.PARAMETER
Values:               Unknown

Member slot:          RIGID.MOTION.YAW
Slot Characteristic:  GEOMETRIC-MODEL.PARAMETER
Values:               Unknown

Member slot:          RIGID.MOTION.Z
Slot Characteristic:  GEOMETRIC-MODEL.PARAMETER
Values:               Unknown

Member slot:          SPECIAL.LEFT.FEATURE.CONSTRAINT
Slot Characteristic:  CONSTRAINT.METHOD
Valueclass:           METHOD
Values:               SPECIAL.LEFT.FEATURE.CONSTRAINT

Member slot:          SPECIAL.RIGHT.FEATURE.CONSTRAINT
```

```
  Slot Characteristic: CONSTRAINT.METHOD
  Valueclass:          METHOD
  Values:              SPECIAL.RIGHT.FEATURE.CONSTRAINT

  Member slot:         SURFACE.TEXTURE
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          SURFACE.TEXTURES
  Values:              Unknown


Feature:  CIRCULAR.HOLES

  Superclasses:        SYMMETRIC.HOLES
  Member of:           CLASSES
  Subclasses:          TAPERED.CIRCULAR.HOLES
                       STRAIGHT.CIRCULAR.HOLES

  Member slot:         AXIS.ANGLE
  Slot Characteristic: FEATURE.PARAMETER
  Values:              Unknown

  Member slot:         COMPOUND.FEATURE
  Slot Characteristic: BOOK-KEEPING.PARAMETER
  Valueclass:          COMPOUND.FEATURES
  Values:              Unknown

  Member slot:         CREATE.NEW.INSTANCE
  Slot Characteristic: CREATION.METHOD
  Valueclass:          METHOD
  Values:              PRODUCT-LIBRARY>NON-CONCENTRIC.FEATURES:
                       CREATE.NEW.INSTANCE!method

  Member slot:         CSG.NODE
  Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
  Values:              Unknown

  Member slot:         DIAMETER
  Slot Characteristic: FEATURE.PARAMETER
  Values:              Unknown

  Member slot:         ESTABLISH.REFERENCE.PRIMITIVE.FEATURES
  Slot Characteristic: CREATION.METHOD
  Valueclass:          METHOD
  Values:              FOR.THROUGH.INTERNAL.DEPRESSIONS

  Member slot:         FILL.PARAMETERS.VALUE
  Slot Characteristic: CREATION.METHOD
  Valueclass:          METHOD
```

```
Values:                 P-F.FILL.PARAMETERS.VALUE

Member slot:            FIND.PARAMETERS.TO.CHANGE
Slot Characteristic:    ACCESS.METHOD
Valueclass:             METHOD
Values:                 FIND.PARAMETERS.TO.CHANGE

Member slot:            GENERATE.CSG.NODE
Slot Characteristic:    GEOMETRIC-MODEL.PARAMETER
Valueclass:             METHOD
Values:                 CIRCULAR.HOLES.CSGNODE

Member slot:            GEOMETRIC.TOLERANCES
Slot Characteristic:    SPECIAL.FEATURE.PARAMETER
Valueclass:             GEOMETRIC.TOLERANCES
Values:                 Unknown

Member slot:            INSERT.NEW.FEATURE
Slot Characteristic:    REPLACEMENT.METHOD
Valueclass:             METHOD
Values:                 NON-CONCENTRIC.INSERT.NEW.FEATURE

Member slot:            KIND.OF.PART
Slot Characteristic:    BOOK-KEEPING.PARAMETER
Valueclass:             KIND.OF.PARTS
Values:                 Unknown

Member slot:            LENGTH
Slot Characteristic:    FEATURE.PARAMETER
Values:                 Unknown

Member slot:            MACHINING.REQUIRED
Slot Characteristic:    FEATURE.PARAMETER
Valueclass:             (ONE.OF YES NO Y N)
Values:                 Unknown

Member slot:            REFERENCE.CONCENTRIC.FEATURE.1
Slot Characteristic:    FEATURE.PARAMETER
Valueclass:             CONCENTRIC.FEATURES
Values:                 Unknown

Member slot:            REFERENCE.CONCENTRIC.FEATURE.2
Slot Characteristic:    FEATURE.PARAMETER
Valueclass:             CONCENTRIC.FEATURES
Values:                 Unknown

Member slot:            REPLACE.FEATURE
Slot Characteristic:    REPLACEMENT.METHOD
Valueclass:             METHOD
```

Values: NON-CONCENTRIC.REPLACE.FEATURE

Member slot: REPLACE.PARAMETER.VALUE
Slot Characteristic: REPLACEMENT.METHOD
Valueclass: METHOD
Values: REPLACE.PARAMETER.VALUE

Member slot: RIGID.MOTION.PITCH
Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
Values: Unknown

Member slot: RIGID.MOTION.ROLL
Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
Values: Unknown

Member slot: RIGID.MOTION.X
Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
Values: Unknown

Member slot: RIGID.MOTION.Y
Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
Values: Unknown

Member slot: RIGID.MOTION.YAW
Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
Values: Unknown

Member slot: RIGID.MOTION.Z
Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
Values: Unknown

Member slot: SURFACE.TEXTURE
Slot Characteristic: FEATURE.PARAMETER
Valueclass: SURFACE.TEXTURES
Values: Unknown

Member slot: X-COORDINATE
Slot Characteristic: FEATURE.PARAMETER
Values: Unknown

Member slot: Z-COORDINATE
Slot Characteristic: FEATURE.PARAMETER
Values: Unknown

Feature: DATUM.SURFACES

Superclasses: DATUMS

```
  Member of:            CLASSES

  Member slot:          CREATE.NEW.INSTANCE
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               PRODUCT-LIBRARY>DATUM.SURFACES:
                        CREATE.NEW.INSTANCE!method

  Member slot:          DATUM.TYPE
  Slot Characteristic:  FEATURE.PARAMETER
  Valueclass:           (ONE.OF A B C)
  Values:               Unknown

  Member slot:          FILL.PARAMETERS.VALUE
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               DATUM.FILL.PARAMETERS.VALUE

  Member slot:          GENERATE.INSTANCE.NAME
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               GENERATE.INSTANCE.NAME

  Member slot:          PRIMITIVE.FEATURE.INSTANCE
  Slot Characteristic:  FEATURE.PARAMETER
  Valueclass:           PRIMITIVE.FEATURES
  Values:               Unknown


Feature:  DATUM.TARGETS

  Superclasses:         DATUMS
  Member of:            CLASSES

  Member slot:          CREATE.NEW.INSTANCE
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               PRODUCT-LIBRARY>DATUM.TARGETS:
                        CREATE.NEW.INSTANCE!method

  Member slot:          DATUM.TYPE
  Slot Characteristic:  FEATURE.PARAMETER
  Valueclass:           (ONE.OF X Y Z)
  Values:               Unknown

  Member slot:          GENERATE.INSTANCE.NAME
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
```

```
  Values:                GENERATE.INSTANCE.NAME

  Member slot:           INSTANCE.COUNTER
  Slot Characteristic:   BOOK-KEEPING.PARAMETER
  Valueclass:            INTEGER
  Values:                0

  Member slot:           PRIMITIVE.FEATURE.INSTANCE
  Slot Characteristic:   FEATURE.PARAMETER
  Valueclass:            PRIMITIVE.FEATURES
  Values:                Unknown


Feature:  STRAIGHTNESS

  Superclasses:          SINGLE.FORM.TOLERANCES
  Member of:             CLASSES

  Member slot:           CREATE.NEW.INSTANCE
  Slot Characteristic:   CREATION.METHOD
  Valueclass:            METHOD
  Values:                PRODUCT-LIBRARY>GEOMETRIC.TOLERANCES:
                         CREATE.NEW.INSTANCE!method

  Member slot:           FILL.PARAMETERS.VALUE
  Slot Characteristic:   CREATION.METHOD
  Valueclass:            METHOD
  Values:                G-T.FILL.PARAMETERS.VALUE

  Member slot:           GENERATE.INSTANCE.NAME
  Slot Characteristic:   CREATION.METHOD
  Valueclass:            METHOD
  Values:                GENERATE.INSTANCE.NAME

  Member slot:           INSTANCE.COUNTER
  Slot Characteristic:   BOOK-KEEPING.PARAMETER
  Valueclass:            INTEGER
  Values:                0

  Member slot:           NEW.INSTANCE.NAME
  Slot Characteristic:   BOOK-KEEPING.PARAMETER
  Values:                GEOMETRIC.TOLERANCES-2

  Member slot:           PRIMITIVE.FEATURE.INSTANCE
  Slot Characteristic:   FEATURE.PARAMETER
  Valueclass:            PRIMITIVE.FEATURES
  Values:                Unknown
```

```
  Member slot:          TOLERANCE.VALUE
  Slot Characteristic:  FEATURE.PARAMETER
  Valueclass:           NUMBER
  Values:               Unknown


Feature:  PARALLELISM

  Superclasses:         RELATED.FORM.TOLERANCES
  Member of:            CLASSES

  Member slot:          CREATE.NEW.INSTANCE
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               PRODUCT-LIBRARY>GEOMETRIC.TOLERANCES:
                        CREATE.NEW.INSTANCE!method

  Member slot:          DATUM.SURFACES
  Slot Characteristic:  FEATURE.PARAMETER
  Values:               Unknown

  Member slot:          FILL.PARAMETERS.VALUE
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               PRODUCT-LIBRARY>RELATED.FORM.TOLERANCES:
                        FILL.PARAMETERS.VALUE!method

  Member slot:          GENERATE.INSTANCE.NAME
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               GENERATE.INSTANCE.NAME

  Member slot:          INSTANCE.COUNTER
  Slot Characteristic:  BOOK-KEEPING.PARAMETER
  Valueclass:           INTEGER
  Values:               0

  Member slot:          NEW.INSTANCE.NAME
  Slot Characteristic:  BOOK-KEEPING.PARAMETER
  Values:               GEOMETRIC.TOLERANCES-2

  Member slot:          PRIMITIVE.FEATURE.INSTANCE
  Slot Characteristic:  FEATURE.PARAMETER
  Valueclass:           PRIMITIVE.FEATURES
  Values:               Unknown

  Member slot:          TOLERANCE.VALUE
  Slot Characteristic:  FEATURE.PARAMETER
```

```
  Valueclass:           NUMBER
  Values:               Unknown


Feature:  SURFACE.TEXTURES

  Superclasses:         PRECISION.FEATURES
  Member of:            CLASSES

  Member slot:          CREATE.NEW.INSTANCE
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               PRODUCT-LIBRARY>SURFACE.TEXTURES:
                        CREATE.NEW.INSTANCE!method

  Member slot:          FILL.PARAMETERS.VALUE
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               SURFACE.TEXTURE.FILL.PARAMETERS.VALUE

  Member slot:          GENERATE.INSTANCE.NAME
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               GENERATE.INSTANCE.NAME

  Member slot:          INSTANCE.COUNTER
  Slot Characteristic:  BOOK-KEEPING.PARAMETER
  Valueclass:           INTEGER
  Values:               0

  Member slot:          NEW.INSTANCE.NAME
  Slot Characteristic:  BOOK-KEEPING.PARAMETER
  Values:               DESIGN.CHARACTERISTICS-1

  Member slot:          PRIMITIVE.FEATURE.INSTANCE
  Slot Characteristic:  FEATURE.PARAMETER
  Valueclass:           PRIMITIVE.FEATURES
  Values:               Unknown

  Member slot:          VALUE
  Slot Characteristic:  FEATURE.PARAMETER
  Valueclass:           (LIST.OF (ONE.OF 40 63 80 100 125))
  Values:               Unknown


 Feature:  CASTING.DIM.AND.TOL
```

```
Superclasses:          PRECISION.FEATURES
Member of:             CLASSES

Member slot:           BASIC.WALL.THICKNESS
Slot Characteristic:   FEATURE.PARAMETER
Valueclass:            NUMBER
Values:                Unknown

Member slot:           CORNER.RADIUS
Slot Characteristic:   FEATURE.PARAMETER
Valueclass:            NUMBER
Values:                Unknown

Member slot:           CREATE.NEW.INSTANCE
Slot Characteristic:   CREATION.METHOD
Valueclass:            METHOD
Values:                PRODUCT-LIBRARY>CASTING.DIM.AND.TOL:
                       CREATE.NEW.INSTANCE!method

Member slot:           DATUM.SURFACES.PROFILE
Slot Characteristic:   FEATURE.PARAMETER
Valueclass:            PROFILE.OF.A.SURFACE
Values:                Unknown

Member slot:           DATUM.TARGETS.PROFILE
Slot Characteristic:   FEATURE.PARAMETER
Valueclass:            PROFILE.OF.A.SURFACE
Values:                Unknown

Member slot:           EXTERIOR.DRAFT.ANGLE
Slot Characteristic:   FEATURE.PARAMETER
Valueclass:            NUMBER
Values:                Unknown

Member slot:           FILLET.RADIUS
Slot Characteristic:   FEATURE.PARAMETER
Valueclass:            NUMBER
Values:                Unknown

Member slot:           INTERIOR.DRAFT.ANGLE
Slot Characteristic:   FEATURE.PARAMETER
Valueclass:            NUMBER
Values:                Unknown

Member slot:           MACHINE.ALLOWANCE
Slot Characteristic:   FEATURE.PARAMETER
Valueclass:            NUMBER
Values:                Unknown
```

```
Member slot:          MINIMUM.WALL.THICKNESS
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           NUMBER
Values:               Unknown
```

A.2 Facility Features

```
Feature:  BEARING.CAGE.CELLS

  Superclasses:         CELL.FEATURES
  Member of:            CLASSES
  Members:              B.C.FIRST.CELL B.C.SECOND.CELL
                        B.C.THIRD.CELL B.C.FOURTH.CELL

  Member slot:          CELL.I.D.TOLERANCE.LIMITATION
  Slot Characteristic:  FEATURE.PARAMETER
  Valueclass:           NUMBER
  Values:               Unknown

  Member slot:          CELL.O.D.TOLERANCE.LIMITATION
  Slot Characteristic:  FEATURE.PARAMETER
  Valueclass:           NUMBER
  Values:               Unknown

  Member slot:          COMMODITY.TYPE.LIMITATION
  Slot Characteristic:  FEATURE.PARAMETER
  Values:               BEARING CAGE BEARING CAGES

  Member slot:          GENERATE.MACHINES.SLOTS
  Slot Characteristic:  CREATION.METHOD
  Valueclass:           METHOD
  Values:               FACILITY-LIBRARY>CELL.FEATURES:
                        GENERATE.MACHINES.SLOTS!method

  Member slot:          LARGEST.LENGTH.LIMITATION
  Slot Characteristic:  FEATURE.PARAMETER
  Values:               Unknown

  Member slot:          LOT.SIZE.LIMITATION
  Slot Characteristic:  FEATURE.PARAMETER
  Values:               Unknown

  Member slot:          MACHINE
  Slot Characteristic:  FEATURE.PARAMETER
  Valueclass:           MACHINE.FEATURES
  Values:               Unknown
```

```
Member slot:          MATERIAL.LIMITATION
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           MATERIAL.FEATURES
Values:               CAST.IRON

Member slot:          NUMBER.OF.MACHINES
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           NUMBER
Values:               Unknown

Member slot:          OVERALL.LENGTH.LIMITATION
Slot Characteristic:  FEATURE.PARAMETER
Values:               Unknown

Member slot:          PROCESSING.ORDER
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           (LIST.OF MACHINE.FEATURES)
Values:               Unknown

Member slot:          WEIGHT.LIMITATION
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           NUMBER
Values:               Unknown


Feature:  TURNING.TOOL.ASSEMBLY.1

  Member of:          TURNING.TOOL.ASSEMBLIES

  Own slot:           CUTTING.TOOL
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:         CUTTING.TOOLS
  Values:             TURNING.TOOL.1

  Own slot:           HEAD
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:         HEADS
  Values:             M6773

  Own slot:           RIGID.MOTION.PITCH
  Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
  Values:             Unknown

  Own slot:           RIGID.MOTION.ROLL
  Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
  Values:             Unknown

  Own slot:           RIGID.MOTION.X
```

```
  Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
  Values:              Unknown

  Own slot:            RIGID.MOTION.Y
  Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
  Values:              Unknown

  Own slot:            RIGID.MOTION.YAW
  Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
  Values:              Unknown

  Own slot:            RIGID.MOTION.Z
  Slot Characteristic: GEOMETRIC-MODEL.PARAMETER
  Values:              Unknown


Feature:  TOOL.MATRIX.1

  Member of:           TOOL.MATRICES

  Own slot:            A.POSITION.VALUE
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          NUMBER
  Values:              43.18

  Own slot:            B.POSITION.VALUE
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          NUMBER
  Values:              151.13

  Own slot:            C.POSITION.VALUE
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          NUMBER
  Values:              259.08

  Own slot:            GENERATE.STATIONS.SLOTS
  Slot Characteristic: CREATION.METHOD
  Valueclass:          METHOD
  Values:              FACILITY-LIBRARY>TOOL.MATRICES:
                       GENERATE.STATIONS.SLOTS!method

  Own slot:            NUMBER.OF.STATIONS
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          NUMBER
  Values:              12

  Own slot:            STATION-1
  Slot Characteristic: FEATURE.PARAMETER
```

```
Valueclass:           CUTTING.TOOL.ASSEMBLIES
Values:               FACING.TOOL.ASSEMBLY.1

Own slot:             STATION-10
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           CUTTING.TOOL.ASSEMBLIES
Values:               Unknown

Own slot:             STATION-11
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           CUTTING.TOOL.ASSEMBLIES
Values:               O.D.PROFILING.TOOL.ASSEMBLY.3

Own slot:             STATION-12
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           CUTTING.TOOL.ASSEMBLIES
Values:               GROOVING.TOOL.ASSEMBLY.1

Own slot:             STATION-2
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           CUTTING.TOOL.ASSEMBLIES
Values:               TURNING.TOOL.ASSEMBLY.1

Own slot:             STATION-3
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           CUTTING.TOOL.ASSEMBLIES
Values:               I.D.PROFILING.TOOL.ASSEMBLY.1

Own slot:             STATION-4
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           CUTTING.TOOL.ASSEMBLIES
Values:               I.D.PROFILING.TOOL.ASSEMBLY.2

Own slot:             STATION-5
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           CUTTING.TOOL.ASSEMBLIES
Values:               I.D.PROFILING.TOOL.ASSEMBLY.3

Own slot:             STATION-6
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           CUTTING.TOOL.ASSEMBLIES
Values:               Unknown

Own slot:             STATION-7
Slot Characteristic:  FEATURE.PARAMETER
Valueclass:           CUTTING.TOOL.ASSEMBLIES
Values:               Unknown

Own slot:             STATION-8
```

```
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          CUTTING.TOOL.ASSEMBLIES
  Values:              O.D.PROFILING.TOOL.ASSEMBLY.2

  Own slot:            STATION-9
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          CUTTING.TOOL.ASSEMBLIES
  Values:              Unknown


Feature:  SPEED.GRIP.1

  Member of:           SPEED.GRIPS

  Own slot:            ADAPTER
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          ADAPTERS
  Values:              PX552028

  Own slot:            BUSHING
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          BUSHINGS
  Values:              Unknown

  Own slot:            DRAWSCREW
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          DRAWSCREWS
  Values:              PX50

  Own slot:            MAXIMUM.INNER.DIAMETER.LIMITATION
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          NUMBER
  Values:              165.1

  Own slot:            MINIMUM.INNER.DIAMETER.LIMITATION
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          NUMBER
  Values:              41.27

  Own slot:            NOSE.PLATE
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:          NOSE.PLATES
  Values:              PX483473

  Own slot:            REPOSITIONING.CAPABILITY
  Slot Characteristic: FEATURE.PARAMETER
  Values:              (:eval (make-i :low 0.1016 :high 1.0e30))
```

```
  Own slot:             WORK.RING
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:           WORK.RINGS
  Values:               Unknown


Feature:  BUCK.CHUCK.1

  Member of:            BUCK.CHUCKS

  Own slot:             DIAMETER
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:           NUMBER
  Values:               381

  Own slot:             INTERMEDIATE.JAW
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:           INTERMEDIATE.JAWS
  Values:               INTERMEDIATE.JAW.1

  Own slot:             LENGTH
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:           NUMBER
  Values:               238.12

  Own slot:             MASTER.JAW
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:           MASTER.JAWS
  Values:               MASTER.JAW.1

  Own slot:             MAXIMUM.OUTER.DIAMTER.LIMITATION
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:           NUMBER
  Values:               304.8

  Own slot:             MINIMUM.OUTER.DIAMTER.LIMITATION
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:           NUMBER
  Values:               63.5

  Own slot:             NUMBER.OF.JAWS
  Slot Characteristic: FEATURE.PARAMETER
  Valueclass:           NUMBER
  Values:               3

  Own slot:             POSITIONING.OF.JAWS
  Slot Characteristic: FEATURE.PARAMETER
  Values:               EQUIDISTANT
```

```
  Own slot:              TOP.JAW
  Slot Characteristic:   FEATURE.PARAMETER
  Valueclass:            TOP.JAWS
  Values:                TOP.JAW.2 TOP.JAW.3 TOP.JAW.1


Feature:  S.C.LATHES

  Superclasses:          LATHES
  Member of:             CLASSES
  Members:               LATHE-1 LATHE-2 LATHE-3
                         LATHE-4 LATHE-5 LATHE-6

  Member slot:           FIXTURE
  Slot Characteristic:   FEATURE.PARAMETER
  Valueclass:            FIXTURES
  Values:                Unknown

  Member slot:           I.D.TOLERANCE.LIMITATION
  Slot Characteristic:   FEATURE.PARAMETER
  Valueclass:            NUMBER
  Values:                Unknown

  Member slot:           O.D.TOLERANCE.LIMITATION
  Slot Characteristic:   FEATURE.PARAMETER
  Valueclass:            NUMBER
  Values:                Unknown

  Member slot:           TOOL.MATRIX
  Slot Characteristic:   FEATURE.PARAMETER
  Valueclass:            TOOL.MATRICES
  Values:                Unknown

  Member slot:           TOOL.MATRIX.HOME.POSITION
  Slot Characteristic:   FEATURE.PARAMETER
  Valueclass:            NUMBER
  Values:                Unknown

  Member slot:           Y.STROKE.LENGTH
  Slot Characteristic:   FEATURE.PARAMETER
  Valueclass:            NUMBER
  Values:                Unknown

  Member slot:           Z.STROKE.LENGTH
  Slot Characteristic:   FEATURE.PARAMETER
  Valueclass:            NUMBER
  Values:                Unknown
```

Appendix B

DIMENSION OBJECT ORIENTED VALUE

B.1 Definition

```
(defvtype :name dimension
          :bit 5
          :v1 value
          :v1arg val
          :v2 tolerance
          :v2arg tol
          :maker make-dim
          :p dim-p
          :print-function
            (lambda (stream depth)
              (declare (ignore depth))
              (format stream "#<dim(~s), Tol(~s)>"
                 (value self)
                 (tolerance self))) )
```

B.2 Methods

```
(cmethod (+ dimension dimension) (d2)
 (make-dim :val (csend (value self) + (value d2))
   :tol (csend (tolerance self) + (tolerance d2))))

(cmethod (- dimension dimension) (d2)
 (make-dim :val (csend (value self) - (value d2))
   :tol (csend (tolerance self) - (tolerance d2))))

(cmethod (= dimension dimension) (d2)
```

```
 (and (csend (value self) = (value d2))
      (csend (tolerance self) = (tolerance d2))))

(cmethod (< dimension dimension) (d2)
 (< (+ (v-v2 (value self)) (v-v2 (tolerance self)))
       (- (v-v1 (value d2)) (v-v2 (tolerance d2)))))

(cmethod (<= dimension dimension) (d2)
 (<= (+ (v-v2 (value self)) (v-v2 (tolerance self)))
        (- (v-v1 (value d2)) (v-v2 (tolerance d2)))))

(cmethod (> dimension dimension) (d2)
 (> (- (v-v1 (value self)) (v-v2 (tolerance self)))
       (+ (v-v2 (value d2)) (v-v2 (tolerance d2)))))

(cmethod (>= dimension dimension) (d2)
 (>= (- (v-v1 (value self)) (v-v2 (tolerance self)))
        (+ (v-v2 (value d2)) (v-v2 (tolerance d2)))))

(cmethod (intersection dimension dimension) (d2)
 (make-dim :val (csend (value self) intersection (value d2))
   :tol (csend (tolerance self) intersection (tolerance d2))))

(cmethod (overlaps dimension dimension) (d2)
 (and (csend (value self) overlaps (value d2))
      (csend (tolerance self) overlaps (tolerance d2))))

(cmethod (subsumes dimension dimension) (d2)
 (and (csend (value self) subsumes (value d2))
      (csend (tolerance self) subsumes (tolerance d2))))

(cmethod (ok dimension nil) ()
 (when (and (csend (value self) ok)
    (csend (tolerance self) ok))
    self))

(cmethod (gen-zerop dimension nil) ()
 (make-dim :val (i 0) :tol (i 0)))

(cmethod (zerop dimension nil) ()
 (when (and (csend (value self) zerop)
    (csend (tolerance self) zerop))
    self))

(cmethod (zero-dimension-p dimension nil) ()
 (if (and (= 0 (i-low (value self)))
          (= 0 (i-high (value self))))
     t
     nil))
```

```
(cmethod (propagate-only-value dimension nil) ()
 (make-dim :val (value self) :tol (i 0)))

(cmethod (good-intersection? dimension dimension) (dim2)
  (csend (csend self intersection dim2) ok))
```

Appendix C

PARAMETRIC DESCRIPTION OF KNOWLEDGE SOURCES

```
KNOWLEDGE.SOURCES

  Member slot:        ACTION.TYPE
  Valueclass:         (ONE.OF INDIVIDUALLY TOGETHER)
  Values:             TOGETHER

  Member slot:        ASSERTION
  Values:             Unknown

  Member slot:        CHAINER.BREAK
  Valueclass:         (ONE.OF ON OFF)
  Values:             OFF

  Member slot:        COMMAND.MENU
  Valueclass:         METHOD
  Values:             GET.RULE.CLASSES.COMMAND.MENU

  Own slot:           COMMAND.MENU
  Valueclass:         METHOD
  Values:             GET.RULE.CLASSES.COMMAND.MENU

  Member slot:        CONCLUSION
  Values:             Unknown

  Member slot:        DELAYED.PREMISE
  Values:             Unknown

  Member slot:        DELETE!
  Valueclass:         METHOD
```

```
Values:             DELETE.RULESYSTEM.UNIT

Member slot:        EXTERNAL.FORM
Values:             Unknown

Member slot:        FETCH.PREMISE
Values:             FETCH.PREMISE.DEFAULT

Member slot:        FROM.BLACKBOARD
Valueclass:         BLACKBOARDS
Values:             Unknown

Member slot:        FROM.LEVEL
Values:             Unknown

Member slot:        GENERATE.INSTANCE.NAME
Valueclass:         METHOD
Values:             GENERATE.INSTANCE.NAME

Member slot:        INSTANCE.COUNTER
Valueclass:         INTEGER
Values:             0

Member slot:        NONMONOTONIC.PREMISES
Values:             Unknown

Member slot:        PARSE
Values:             DEFAULT.RULE.PARSER

Member slot:        PREMISE
Values:             Unknown

Member slot:        RATING.CONDITIONS
Valueclass:         LIST
Values:             Unknown

Own slot:           REINDEX
Valueclass:         METHOD
Values:             REINDEX

Member slot:        RULE.ABSTRACT
Values:             Unknown

Member slot:        RULE.TYPE
Valueclass:         (ONE.OF SAME.WORLD.ACTION NEW.WORLD.ACTION
                     DEDUCTION)
Values:             Unknown

Member slot:        STEP.BY
```

```
Valueclass:          (ONE.OF BY.PREMISE BY.RULE OFF)
Values:              BY.PREMISE

Member slot:         TO.BLACKBOARD
Valueclass:          BLACKBOARDS
Values:              Unknown

Member slot:         TO.FETCH.RULE
Values:              TO.FETCH.RULE

Member slot:         TO.INDEX.RULE
Values:              TO.INDEX.RULE

Member slot:         TO.LEVEL
Values:              Unknown

Member slot:         TO.UNINDEX.RULE
Values:              TO.UNINDEX.RULE
```

BIBLIOGRAPHY

[1] Ovens, W. D. and Dekker, D. L., "Design for Manufacturing: Starting at the Beginning," *Symposium on Advanced Topics in Manufacturing Technology*, ASME Winter Annual Meeting, Dec. 13-18, 1987, pp 1-10.

[2] Evans, B., "Simultaneous Engineering," *Mechanical Engineering*, Feb. 1988, pp 38-40.

[3] Tuttle, H. C., "Breaking the 'Wall' Between Design and Manufacturing," *Production*, May 1983, pp 63-66.

[4] Boothroyd, G., "Design for Producibility- The road to Higher Productivity," *Assembly Engineering*, March 1982, pp 42-45.

[5] Behuniak, J. A., "Design for Automation: The Competitive Edge," *Proc. 5th. Intl. Conf. on Assembly Automation*, Paris, 1984.

[6] Lu, S. C-Y. and Subramanyam, S., "A Computer-Based Environment for Simultaneous Product and Process Design," *Proc. of Advances in Manufacturing Systems, ASME Winter Annual Meeting*, PED.Vol.31, Chicago, Nov. 28-Dec. 2, 1988, pp 35-46.

[7] Boothroyd, G., "Design for Assembly- The Key to Design for Manufacture," Report #9, Department of Industrial and Manufacturing Engineering, University of Rhode Island, Kingston, RI.

[8] Fischer, W. R., "Design for Assembly," *Proc. of the IEEE Reliability and Maintainability Symposium*, New York, 1984, pp 409-411.

[9] Maczka, W. J., "GE has 'Designs' on Assembly," *Assembly Engineering*, June 1984, pp 16-18.

[10] Anon, K., "Design for Easy Assembly," *The Production Engineer*, Dec. 1982, pp 11-13.

[11] Boothroyd, G., "Design for Assembly: Making it Simple," *Mechanical Engineering*, Feb. 1988, pp 28-31.

[12] Lai, W., "Mechanical Design Simplification Using Functional Description Language," *Proc. of the 15th. NAMRC*, Bethlehem, PA, 1987, pp 615-620.

[13] Ulrich, K. T. and Seering, W. P., "Function Sharing in Mechanical Design," *Proc. of AAAI-88*, 1988, pp 342-346.

[14] Ishii, K., Adler, R. and Barkan, P., "Application of Design Compatibility Analysis to Simultaneous Engineering," *AI EDAM*, Vol. 2, 1988, pp 53-65.

[15] Ishii, K. and Barkan, P., "Design Compatibility Analysis– A Framework for Expert Systems in Mechanical System Design," *ASME Computers in Engineering*, Vol. 1, pp 95-102.

[16] Rehg, J., Elfes, A., Talukdar, S., Woodbury, R., Eisenberger, M. and Edahl, R., "CASE: Computer-Aided Simultaneous Engineering," *Intl. Conf. on Application of Artificial Intelligence in Engineering*, 1988, pp 339-359.

[17] Finger, S., Fox, M. S., Navinchandra, D., Prinz, F. B. and Rinderle, J. R., "The Design Fusion Project: A Product Life-Cycle View for Engineering Designs," Unpublished Report, Carnegie Mellon University, 1988.

[18] Lu, S. C-Y., Subramanyam, S., Thompson, J. B. and Klein, M., "A Cooperative Product Development Environment to Realize the Simultaneous Engineering Concept," *ASME Computers In Engineering Conference*, San Francisco, CA, Aug. 1989.

[19] Baskin, A. B., Lu, S. C-Y., Klein, M. and Stepp, R. E., "Integrated Design as a Cooperative Problem Solving Activity," *Proc. of 22nd. Intl. Conf. on System Sciences*, HI, Jan. 1989.

[20] Dewhurst, P. and Boothroyd, G., "Early Cost Estimating in Product Design," Report #11, Department of Industrial and Manufacturing Engineering, University of Rhode Island, Kingston, RI.

[21] Boothroyd, G., "Design for Machining," Report #3, Department of Industrial and Manufacturing Engineering, University of Rhode Island, Kingston, RI, 02881.

[22] Luby, S. C., Dixon, J. R. and Simmons, M. K., "Creating and Using a Features Data Base," *Computers in Mechanical Engineering*, Nov. 1986, pp 25-33.

[23] Luby, S. C. and Simmons, M. K., "Designing with Features: Creating and Using a Features Data Base for Evaluation of Manufacturability of Castings," *Proc. of the ASME Computers in Engineering Conference*, Chicago, IL, July 20-24, 1986.

[24] Ramalingam, S., "Expert Systems for Manufacturing: Examples of Tools to Assess Manufacturability," *Proc. of the 13th. NAMRC*, pp 411-417.

[25] Vaghul, M., Dixon, J. R., and Simmons, M. K., "Expert Systems in a CAD Environment: Injection Molding," *Proc. of the ASME Computers in Engineering Conference*, Boston, MA, 1985, pp 77-82.

[26] Rowe, G. W., "An Intelligent Knowledge-Based System to provide Design and Manufacturing Data for Forging," *Compter-Aided Engineering Journal*, Feb. 1987, pp 56-61.

[27] Cutkosky, M. R. and Tenenbaum, J. M., "CAD/CAM Integration through Concurrent Process and Product Design," *Proc. of Symposium on Computer Integrated Manufacturing, ASME Winter Annual Meeting*, Dec. 1987, Boston, MA, pp 1-10.

[28] DeVor, R. E., Kapoor, S. G., Hayashida, R. and Subramani, G., "A Methodology for the Simultaneous Engineering of Products and Manufacturing Processes," *Proc. of the 14th. NAMRC*, pp 542.

[29] Sutherland, J. W., DeVor, R. E., Kapoor, S. G. and Ferreira, P. M., "Machining Process Models for Product and Process Design," *Proc. of Society of Automotive Engineers and Earth Moving Industry Conference*, Apr. 12-14, 1988.

[30] Tong, C., "Toward an Engineering Science of Knowledge-Based Design," *Artificial Intelligence in Engineering*, Vol. 2, No. 3, 1987.

[31] Howe, A., Cohen, P. Dixon, J. and Simmons, M., "Dominic: A Domain-Independent Program for Mechanical Engineering Design," *Proc. of the First Intl. Conf. on Applications of Artificial Intelligence to Engineering Problems*, Southampton, England, Apr. 15-18, 1986.

[32] Dixon, J. R., Howe, A., Cohen, P. R. and Simmons, M., "Dominic I: Progress toward Domain Independence in Design by Iterative Redesign," *Engineering with Computers*, Vol. 2, 1987, pp 137-145.

[33] Dixon, J. R. and Simmons, M. K., "Expert Systems for Engineering Design: Standard V-belt Drive Design as an Example of the Design-Evaluate-Redesign Architecture," *Proc. of the ASME Computers in Engineering Conference*, 1984, pp 332-337.

[34] Kulkarni, V. M., Dixon, J. R., Sunderland, J. E., and Simmons, M. K., "Expert Systems for Design: The design of Heat Fins as an Example of Conflicting Subgoals and the Use of Dependencies," *Proc. of the ASME Computers in Engineering Conference*, 1985, pp 145-150.

[35] McDonald, M., "Constraint Based Design of Mechanical Systems of Components," M.S. thesis, University of Minnesota, Oct. 1985.

[36] Langrana, N. A., Mitchell, T. M. and Ramachandran, N., "Progress Toward a Knowledge-Based Aid for Mechanical Design," *Proc. of the Symposium on Integrated and Intelligent Manufacturing*, ASME Winter Annual Meeting, 1986.

[37] Brown, D. C. and Chandrasekaran, B., "Knowledge and Control for a Mechanical Design Expert System," *Computer*, IEEE, 1986, pp 92-100.

[38] Brown, D. C. and Chandrasekaran, B., "Expert Systems for a class of Mechanical Design Activity," *Knowledge Engineering in Computer-Aided Design*, Elsevier Science Publishers, B. V. (North-Holland), 1985, pp 259-290.

[39] Mittal, S. and Araya, A., "A Knowledge-Based Framework for Design," *Proc. of AAAI-86*, 1986, pp 856-865.

[40] Araya, A. and Mittal, S., "Compiling Design Plans from Descriptions of Artifacts and Problem-Solving Heuristics," *Proc. of Intl. Joint Conf. of Artificial Intelligence*, 1987, pp 552-558.

[41] Mayer, A. K. and Lu, S. C-Y., "An AI-Based Approach for the Integration of Multiple Sources of Knowledge to Aid Mechanical Engineering Design," *Journal of Mechanisms, Transmissions and Automation in Design, Trans. of the ASME*, Vol. 110, No. 3, 1988, pp 316-323.

[42] Rychener, M. D., Farinacci, M. L., Hulthage, I., Fox, M. S., "Integration of Multiple Knowledge Sources in ALADIN, an Alloy Design System," Proc. of AAAI-86, 1986.

[43] Link, C. H., "CAPP - CAM-I Automated Process Planning System," *Proc. 13th. Numerical Control Society Annual Meeting and Technical Conference*, Cincinnati, OH, March 1976.

[44] Schafer, G., "GT via Automated Process Planning," *American Machinist*, May 1980, 119-122.

[45] Doran, J. M., and Sechrist, N. W., "Computer-Aided Process Planning and Work Measurement," *15th. Numerical Control Society Annual meeting Tech. Conf.*, Chicago, IL, Apr. 9-12, 1978.

[46] Tulkoff, J., "Lockheed's GENPLAN," *18th. Numerical Control Society Technical Conf.*, Dallas, TX, May 1981, 417-421.

[47] CAM-I, "Functional Specification for an Experimental Planning System XPS-1," Computer Aided Manufacturing- International, Inc., Arlington, TX, Oct. 1980.

[48] Li, J., Han, C. and Ham, I., "CORE-CAPP – A Company-oriented Semi-generative Computer Automated Process Planning System," *Proc. 19th. CIRP Intl. conf. on Manufacturing Systems*, Penn. State, USA, June 1-2, 1987.

[49] Allen, D. K. and Smith, P. R., "Computer Aided Process Planning," Technical Report, Computer-Aided Manufacturing Laboratory, Brigham Young University, Provo, Utah, Oct. 15, 1980.

[50] Wysk, R. A., "An Automated Process Planning and Selection Program: APPAS," Ph.D. thesis, Purdue University, West Lafayette, IN, 1977.

[51] Kotler, R. A., "Computerized Process Planning- Part 1 & 2," *Army ManTech Journal*, Vol 4, No. 4 and No. 5, 1980.

[52] Chang, T. C., "Interfacing CAD and CAM - A Study of Hole Design," M.S. thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1980.

[53] Vogel, S. A. and Adlard, J. E., "The Autoplan Process Planning System," *Proc. of 18th. Int. Tech. Conf. NC Control Society*, 1981.

[54] Descotte, Y. and Latombe, J-C., "Making Compromises Among Antagonist Constraints in a Planner," *Artificial Intelligence*, Vol. 27, 1985, 183-217.

[55] Matsushima, K., Okada, N. and Sata, T., "The Integration of CAD and CAM by Application of Artificial Intelligence Techniques," *Annals of the CIRP*, Vol. 11, 1982.

[56] Brooks, S. L., "Applying Artificial Intelligence Techniques to Generative Process Planning Systems," M.S. thesis, May 1986.

[57] Nau, D. S. and Chang, T. C., "A Knowledge-Based Approach to Generative Process Planning," *Symposium of Computer-aided Intelligent Process Planning*, ASME Winter Meeting, Miami Beach, FL, 1985.

[58] Ham, I. and Lu, S. C-Y., "Computer-Aided Process Planning: The Present and the Future," *Annals of the CIRP*, Vol. 37, No.2, 1988, pp 1-11.

[59] Requicha, A. A. G., "Representations for Rigid Solids: Theory, Methods, and Systems," *Computing Surveys*, Vol. 12, No. 4, Dec. 1980, pp 437-464.

[60] Spur, I. G., Krause, I. F. L., "A Survey about Geometric Modeling Systems," *Annals of the CIRP*, Keynote Paper, 1979.

[61] Bradford, D., "Through the Labyrinth of Solids Modeling," *Mechanical Engineering*, March 1988, pp 30-34.

[62] Cunningham, J. J. and Dixon, J. R., "Designing with Features: The Origin of Features," *Proc. of the ASME Computers in Engineering Conf.*, San Francisco, CA, July 1988, pp 237-243.

[63] Eversheim, W. and Diels, "Changing Requirements for CAP - Systems Lead to a New CAP-Data Model," *Proc. 19th. CIRP Intl. conf. on Manufacturing Systems*, Penn. State, USA, June 1-2, 1987.

[64] Shah, J. J. and Rogers, M. T., "Functional Requirements and Conceptual Design of the Feature-Based Modeling System," *Computer-Aided Engineering Journal*, Feb. 1988, pp 9-15.

[65] Cutkosky, M. R., Tenenbaum, J. M. and Muller, D., "Features in Process-Based Design," *Proc. of ASME Computers in Engineering Conf.*, Oct. 1988.

[66] Klein, A., "A Solid Groove: Feature-Based Programming of Parts," *Mechanical Engineering*, March 1988, pp 37-39.

[67] Woodbury, R. F., "The Knowledge Based Representation and Manipulation of Geometry," Ph.D. dissertation, Department of Architecture, Carnegie-Mellon University, Pittsburgh, PA.

[68] Stefik, M. and Bobrow, D. G., "Object-Oriented Programming: Themes and Variations," *The AI magazine*, AAAI, 1986, pp 40-62.

[69] McKelvey, R. D., "VEGA2: Progress Report," Report, Department of Architecture, Carnegie Mellon University, Pittsburgh, PA.

[70] Nardi, B. A. and Simons, R. K., "Model-Based Reasoning and AI Problem Solving," *Proc. of Workshop on High Level Tools for Knowledge Based Systems*, Oct. 6-8, 1986, Columbus, OH.

[71] Smith, B., "Models in Expert Systems," *Proc. of Ninth Intl. Joint Conf. on Artificial Intelligence*, Los Angeles, CA, 1985, pp 1308-1312.

[72] Koton, P. A., "Empirical and Model-Based Reasoning in Expert Systems," *Proc. of Ninth Intl. Joint Conf. on Artificial Intelligence*, Los Angeles, California, 1985, pp 297-299.

[73] Craig, I. D., "Blackboard Systems," *Artificial Intelligence Review*, 1988, Vol. 2, 103-118.

[74] Erman, L. D. and Lesser, V. R., "A Multi-Level Organization for Problem Solving Using Many Diverse Cooperating Sources of Knowledge," *Proc. of Intl. Conf. on Artificial Intelligence*, Tbilisi, USSR, 1975, pp 483-490.

[75] Hayes-Roth, B., "A Blackboard Architecture for Control," *Artificial Intelligence*, Vol. 26, 1985, pp 251-321.

[76] Nii, P. H., "Blackboard Systems," *The AI Magazine*, Summer 1986, pp 38-52.

[77] Minsky, M., "A Framework for Representing Knowledge," *The Psychology of Computer Vision*, 1976, pp 211-277.

[78] Fikes, R. E. and Kehler, R. E., "The Role of Frame-based Representation in Reasoning," *Communications of the ACM*, Sept. 1985, pp 904-920.

[79] Steele, G. L., "The Definition and Implementation of a Computer Programming Language Based on Constraints," Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT, Aug. 1980.

[80] Herman, A. E. and Lu, S. C-Y., "A new Modeling Environment for Developing Computer-Based Intelligent Associates with an Application in Machining Operation Planning," Proc. of 16th. NAMRC, May 24-26, 1989.

[81] Kilhoffer, A. R. and Kempf, K. G., "Designing for Manufacturability in Riveted Joints," Proc. of Intl. Joint Conf. of Artificial Intelligence, 1986, pp 820-824.

[82] Cook, J., "Back to Simplicity," *Forbes*, Aug. 25, 1986, pp 32-34.

[83] Hyer, N. L. and Wemmerlov, U., "Group Technology and Productivity," *Harward Business Review*, July-Aug. 1984, pp 140-149.

[84] Forbus, K., "Intelligent Computer-Aided Engineering," *Artificial Intelligence*, Vol. 9, No. 3, Fall 1988, pp 23-36.

[85] "Romulus-D Focus Project," Internal Project Report, Caterpillar Inc., Peoria, IL, June 1988.

[86] Davis, R., "Diagnostic Reasoning Based on Structure and Behaviour," *Artificial Intelligence*, Vol. 24, 1984, pp 347-410.

[87] "Keeconnection: A Bridge Between Databases and Knowledge Bases," Intellicorp Technical Report, Intellicorp, Mountain View, CA, 1987.

[88] Balakumar, P., Robert, J-C., Kanade, T., Ikeuchi, K., Hoffman, R., "VANTAGE: A Frame-Based Geometric Modeling System, Programmer/User's Manual V1.0," Carnegie Mellon University, Pittsburgh, PA, June 1988.

[89] "Romulus-D: Introduction to FOL," Version 3, Shape Data Ltd., Cambridge, England, June 1988.

[90] Fu, N. J., DeVor, R. E., and Kapoor, S. G., "A Mechanistic Model for the Prediction of the Force System in Face Milling Operations," *Trans. of Jour. of Engg. for Industry*, ASME, 106, 1984, pp 81-87.

[91] Zhang, G. M., "Dynamic Modeling and Dynamic Analysis of the Boring Machining System," Ph.D. thesis, University of Illinois at Urbana-Champaign, 1985.

[92] Subramanyam, S. and Lu, S. C-Y., "An Integrated AI/OR Approach to Operation Planning Using Process Performance Models," *Proc. of the 16th. NAMRC*, Urbana, IL, May 24-26, 1988, pp 388-394.

[93] Kuhl, M. J., "The Prediction of Cutting Forces and Surface Accuracy for the Turning Process," M.S. thesis, University of Illinois at Urbana-Champaign, 1987.

[94] Hwang, C. L., and Md. Massud, A. S., Multiple Objective Decision Making- Methods and Applications, Springer-Verlag, Berlin Heidelberg, New York, 1979.

[95] Subramanyam, S., Lu, S. C-Y. and Zdeblick, W. J., "A Characterization of the Process Planning Task from an Artificial Intelligence Perspective," *Proc. of the 19th. CIRP Intl. Seminar on Manufacturing Systems*, June 1-2, 1987, pp 197-206.

[96] Doyle, J., "A Truth Maintenance System," *Artificial Intelligence*, Vol. 12, 1979, pp 231-272.

[97] Knowledge Engineering Environment Reference Manuals, Intellicorp, Mountain View, CA, 94040.

[98] Subramanyam, S. and Lu, S. C-Y., "Computer-Aided Simultaneous Engineering for Components Manufactured in Small and Medium Lot-Sizes," To be presented at *Symposium on Concurrent Product and Process Design*, ASME Winter Annual Meeting, 1989 and Submitted to *Journal of Engineering for Industry, Trans. of the ASME*, 1989.

[99] "Manufacturability of Bearing Cages," Internal Project Report, Caterpillar Inc., 1988.

[100] "Expert Systems vs D-Class," Internal Project Report, Caterpillar Inc., 1985.

[101] "Caterpillar Hydraulic Excavator Bucket 3D Modeler," Internal Project Report, Caterpillar Inc., 1988.

[102] Subramanyam, S. and Lu, S. C-Y., "The Impact of an AI-Based Environment for Simultaneous Engineering on Process Planning," Submitted to *International Journal of Computer Integrated Manufacturing*, 1989.

[103] Klein, M. and Lu, S. C-Y., "Run-Time Conflict Resolutions for Cooperative Design," *Proc. of AAAI Workshop on AI in Design*, Minneapolis, MN, Aug. 1988.

[104] Lu, S. C-Y. and Chen, K., "A Machine Learning Approach to the Automatic Synthesis of Mechanistic Knowledge for Engineering Decision Making," *Journal of Artificial Intelligence in Engineering, Design and Manufacturing*, Vol. 1, No. 2, 1987, pp 109-118.

[105] Thompson, J. B. and Lu S. C-Y., "Representing and Using Design Rationale in Concurrent Product and Process Design," Submitted to *Symposium on Concurrent Product and Process Design, ASME Winter Annual Meeting*, Anaheim, CA, Dec. 1989.

VITA

Sridhar Subramanyam was born on December 20, 1960. He was raised in the three major Indian cities of Bombay, New Delhi, and Madras. He graduated from the Indian Institute of Technology (IIT), Madras, in September of 1983 with a Bachelor of Technology degree in Mechanical Engineering. In his senior year at IIT, he was awarded the Institute Medal for Excellence in Athletics. He continued his studies at Syracuse University, Syracuse, and University of Iowa, Iowa City; in May of 1985 he received a Master of Science degree in Industrial and Management Engineering. During this period he received a Graduate Student Fellowship and worked as a Reseach and Teaching Assistant. In June 1985, he joined the Department of Mechanical and Industrial Engineering at the Univeristy of Illinois at Urbana-Champaign for his doctoral studies. Since then he has worked as a Research Assistant on several research projects with Allied Signal Aerospace, Caterpillar Inc., and Metcut Research Associates Inc. In the summer of 1987 and 1988, he worked as a Research Engineer in the Technical Center at Caterpillar Inc.

His technical interests include Simultaneous Engineering, Computer-Aided Engineering Automation, Cooperative Problem-Solving, Artificial Intelligence, and Operations Research. He is a member of the Alpha Pi Mu honor society, and is an active student member of the following professional societies: ASME, SME, AAAI and IIE.