WL-TR-93-4021

2

MEMORY DRIVEN FEATURE-BASED DESIGN

Y.H. PAO
F.L. MERAT
G.M. RADACK

CASE WESTERN RESERVE UNIVERSITY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
CLEVELAND, OH 44106-7221

JANUARY 1993

INTERIM REPORT FOR PERIOD FEBRUARY 1988-DECEMBER 1992

MATERIALS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7734

NOTICE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** January 1993	**3. REPORT TYPE AND DATES COVERED** Interim (Feb 1988 - Dec 1992)

4. TITLE AND SUBTITLE

Memory Driven Feature Based Design

5. FUNDING NUMBERS

C: F33615-87-C-5250
PE 62102F
PR: 2306
TA: P9
WU: 03

6. AUTHOR(S)

Pao, Y. H., Merat, F. L., Radack, G. M.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Case Western Reserve University
Electrical Engineering and Computer Science
Cleveland, OH 44106-7721

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Materials Directorate
Wright Laboratory
Air Force Material Command
Wright Patterson AFB OH 45433-7734

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

WL-TR-93-4021

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The act of design is a critical operation in the whole train of events of manufacturing. Improvement of the efficiency and effectiveness of the design process would certainly have a major effect on the entire manufacturing operation as a whole. Engineering design seems to be an admixture of synthesis and analysis activities with synthesis dominating the "conceptual design" and "embodiment design" phases of the activity; and analysis dominating the "problem definition" and "detailing" phases of the activity. In addition, in nearly all phases of the design procedure, the designer often finds it helpful to be able to recollect many other previous designs or pertinent portions of such similar designs; and to be able to visualize or sketch variations on a theme; and to be able to evaluate those comparatively. This is especially true for circumstances where the person has not had extensive personal experience in this matter. The research reported on in this document focuses on that aspect of design and planning. Namely, how stores of public (handbook type) knowledge and memory of private (episodic) knowledge can be and should be used effectively in combination with computer-based systems, in support of designers and fabricators.

14. SUBJECT TERMS

Memory   Neural Networks
Features   Computer Aided Design

15. NUMBER OF PAGES 95

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

FOREWORD

This report was prepared by Case Western Reserve University (CWRU), Cleveland, Ohio, under USAF Contract No. F33615-87-C-5250. This is an interim report summarizing formally a body of research carried out over a period of four years from February 1988 to December 1992. This report covers work carried out by CWRU faculty and students. However, this work was carried out in close contact with researchers in the Manufacturing Research Branch of the Materials Directorate, Wright Laboratory, on a daily basis with some of the work done on-site at the base. Work was administered by Dr. Steven R. LeClair. Research results to date have also appeared annually in reports generated by Dr. LeClair.

Over the period covered by this report, participants of the effort have included the following individuals in the following categories:

Faculty:	Yoh-Han Pao (P.I.), Frank L. Merat, Gerald M. Radack, Lee J. White, and Y. Takefuji.
Senior Research Associate:	Dejan J. Sobajic
Research Associates:	Wassim Hafez and Natarajan Balunsundra
Graduate Students:	Robert Delvalle, Kavous Roumina, Steve Weinroth, Steven Ruegsegger, Jeremy Jacobsohn, Kambiz Komeyli, Darlene Shei, Tom Kao, Amit Kohli, Bharat Mavinkere, Alok Mathus, Jaume Pujol-Busquets Pinana, Derya Ferendeci, and Jeffrey Knaus.

TABLE OF CONTENTS

Table of Contents (con't)

LIST OF ILLUSTRATIONS

List of ILLUSTRATIONS (con't)

LIST OF TABLES

ACKNOWLEDGMENTS

We acknowledge with great pleasure the pleasant and strengthening interactions and collaborations we have had with the other researchers on the RDS project including Air Force personnel at WL: Capt Charles Wright, Capt Chris Cartaya, Capt Tim Westhoven, Capt Baxter Turner, Capt Chris Madden, Mr Steven Ruegsegger, Mr Jeff Ashcom, and Mr. Alan Winn.

We appreciate the management leadership of Dr Walt Griffith and Dr Steven R. LeClair.

I. INTRODUCTION TO REPORT

This document reports on the progress made in a continuing research program on the roles of learning, memory, associative recall, and reasoning in design and manufacturing.

Over the years, co~ .derable progress has been made in the computer-based automation design and manufacturing tasks. Computer-Aided Design (CAD) systems are now used widely; and Computer-Aided Manufacturing, Computer-Aided Inspection and parametric engineering practices have combined with CAD to greatly increase accuracy and efficiency in the implementation of design intent.

Given this degree of automation, there is the opportunity to heighten its impact on productivity by adding intelligent system methodology to it; so as to facilitate use of that automation in the support of humans in the performance of cognitive and creative tasks, as well as routine tasks.

The act of design is a critical operation in the whole train of events of manufacturing. Improvement of the efficiency and effectiveness of the design process would certainly have a major effect on the entire manufacturing operation as a whole.

Engineering design seems to be an admixture of synthesis and analysis activities with synthesis dominating the "conceptual design" and "embodiment design" phases of the activity and analysis dominating the "problem definition" and "detailing" phases of the activity.

In addition, in nearly all phases of the design procedure, the designer often finds it helpful to be able to recollect many other previous designs or pertinent portions of such similar designs and to be able to visualize or sketch variations on the theme and to be able to evaluate those comparatively. This is especially true for circumstances where the person has not had extensive personal experience in this matter.

The research, which we report on in this document, focuses on that aspect of design and planning. Namely, how stores of public (handbook type) knowledge and memory of private (episodic) knowledge can be and should be used effectively in combination with computer-based systems, in support of designers and fabricators.

Specifically, we want to see how we can help human task performers benefit from previous experience and from group experience so that nearly similar product designs and processing plans can be modified easily and accurately; good designs and plans would then be obtained easily. Because the task performer would be alerted to potential problems, previous mistakes would not be repeated. Our expectations are that approach will result in dramatic improvements in the integration of design, process, inspection, and materials knowledge with the consequence that options can be formulated and choices can be optimized.

We believe that progress in these aspects of higher-level automation will be especially beneficial for small-scale odd-lot manufacturing where there is no opportunity to build-up experience over a lengthy run of mass manufacturing, and where there is no opportunity to spread the cost of elaborate-hard automation over large numbers of products.

To provide tools and an environment for carrying out such research, we participated in a substantial manner in the development of a major computer software system called the Rapid Design System (RDS). This document reports, in particular, on progress made in the development of three modules of the RDS: the design, the memory, and the inspection plan generation modules. The research, to date, has focused on computer-based intelligent information processing methodologies, but has also dealt with human cognitive processes in the context of design and manufacturing.

An overview of the RDS research effort and the RDS system architecture is presented in Section 2.

A central issue of intelligent systems for manufacturing is the matter of how designs might be described. Our work leads us to believe that there needs to be a feature-based description of a design at several levels. In our work, we start with an intermediate level, so to speak, in terms of features which come naturally to designers; for example, slots, pockets, holes, and so on but provide means for specifying these features in detailed geometry, as well as in more abstract categorical terms. The design module is positioned at the "basic-level" and is called the Feature-Based Design Environment. It is described in Section 3.

The memory module of RDS is an associative memory configured to have the mass storage and recall capabilities of computers and yet be compatible with the associative recall characteristics of humans. The Episodal Associative Memory is described in Section 4, and the Inspection Planning and Evaluation Module are described in Section 5.

2. OVERVIEW OF THE RDS PROJECT AND OF RDS SYSTEM ARCHITECTURE

Constraints on Research

The work reported in this document was carried out with some important constraints imposed on the course of the research.

On a strategic planning level, it was decided that despite the research nature of the program, it was also important that the results of the work ultimately be of practical significance. It was felt that if human task performers found RDS to be easy to use and helpful to them in their work, they would collaborate with us in our research and truly significant insights into the nature of human/computer-intelligent system interaction might be attained. By the same token, if RDS is truly useful then it might be adopted for use in small-scale manufacturing operations, and perhaps we should prepare for that technology transfer eventuality.

Towards those ends, numerous working alliances were built up with communities of fabrication and design personnel; and as a result our sets of features at different representation levels conform with those in common usage. In addition, they are compatible with the STEP/PDES (Standard for The Exchange of Product model data/Product Data Exchange using STEP) model.

Furthermore, dimensioning and tolerancing practices and inspection plan features are in conformance with ANSI standards. The same is also true of fabrication features and practices.

In the interest of ensuring software coherence and integrity, RDS makes use of and builds on commercially available software packages, whenever possible. Examples of such software are the CONCEPT MODELLER® from Wisdom Systems, N-NET® from AI WARE, and METCAPP® from the Institute of Advanced Manufacturing Science in Cincinnati.

The RDS comprises four modules: the Feature-Based Design Environment (FBDE), the Fabrication process planning module (FAB-PLAN), the inspection planning and evaluation module (IPEM), and the Episodal Associative Memory (EAM). In addition, we make use of a commercially available object-oriented data base called ITASCA® for PDES Research. This report covers only the FDBE, the EAM, and the IPEM modules.

We believe that our self-imposed constraints do not limit the mission, nor the scope of our research. They help to ensure that the research addresses significant issues, and evokes the interest and support of the manufacturing community. However, these constraints have certainly greatly influenced priorities in the scheduling of the different tasks. For example, our research and innovative results have been primarily in the design and implementation of the RDS as a tool and environment. Actual research into the role of a memory-driven, feature-based machine intelligent system for use into the support of the human cognitive task called design for integrated small scale manufacturing is just being made possible through the efforts of these past years.

Overview of Architecture and Functionalities

The systems architecture of RDS is object-oriented and modular. As mentioned previously in this document, we only report on aspects of our work on the FBDE, the EAM, and the IPEM modules. Work on other modules has been and will be described by the researchers involved.

Since RDS is being designed and implemented for use in support of human task performance at present, there is no need for elaborate systems control knowledge for intermodular communication.

The principal modules are shown schematically in Figure 1.

An overview of how the different modules might interact in the design mode is provided by schematic illustration shown in Figure 2. Different utilization paths would be more appropriate for the Fabrication Process Planner or for someone responsible for inspection.

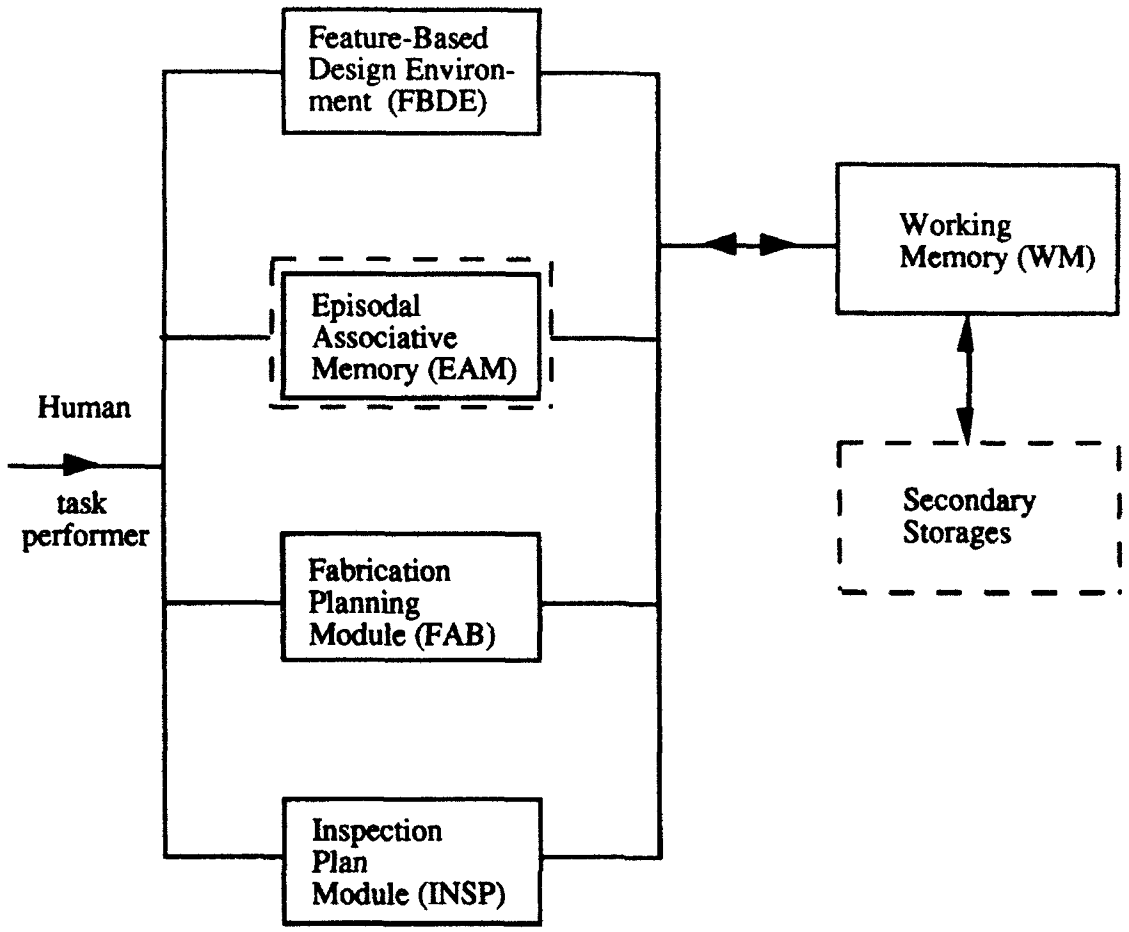

Feature-Based Design Environ-ment (FBDE)
Episodal Associative Memory (EAM)
Fabrication Planning Module (FAB)
Inspection Plan Module (INSP)
Working Memory (WM)
Secondary Storages
Human
task performer

Figure 1. Modular systems architecture of the Rapid Design System (RDS)

In the design mode, the user starts with a tentative design in the FBDE. He can ask the RDS if any similar design had been encountered in the past, or he can continue to carry out the newdesign entirely on his own. If the RDS is asked, the user enters into the EAM module where previous remembered designs are recalled on the basis of the cues posed by the present active entative design. The recalled information is not just in the nature of a mechanical design, but will have with it pointers to items such as Problems Encountered (and Solved), or Potential Problems, and so on.

The user can modify any one of those previous designs and adopt it for his own or he might incorporate parts of several designs into his active design. He can also examine the "Fabrication Features and Plan" part of the part model to see if problems might be expected there. This process of recall, modify, incorporate, and improve a process can be carried out iteratively until the user is satisfied, at which point he can store the entire new design in the Episodal Associate Memory.

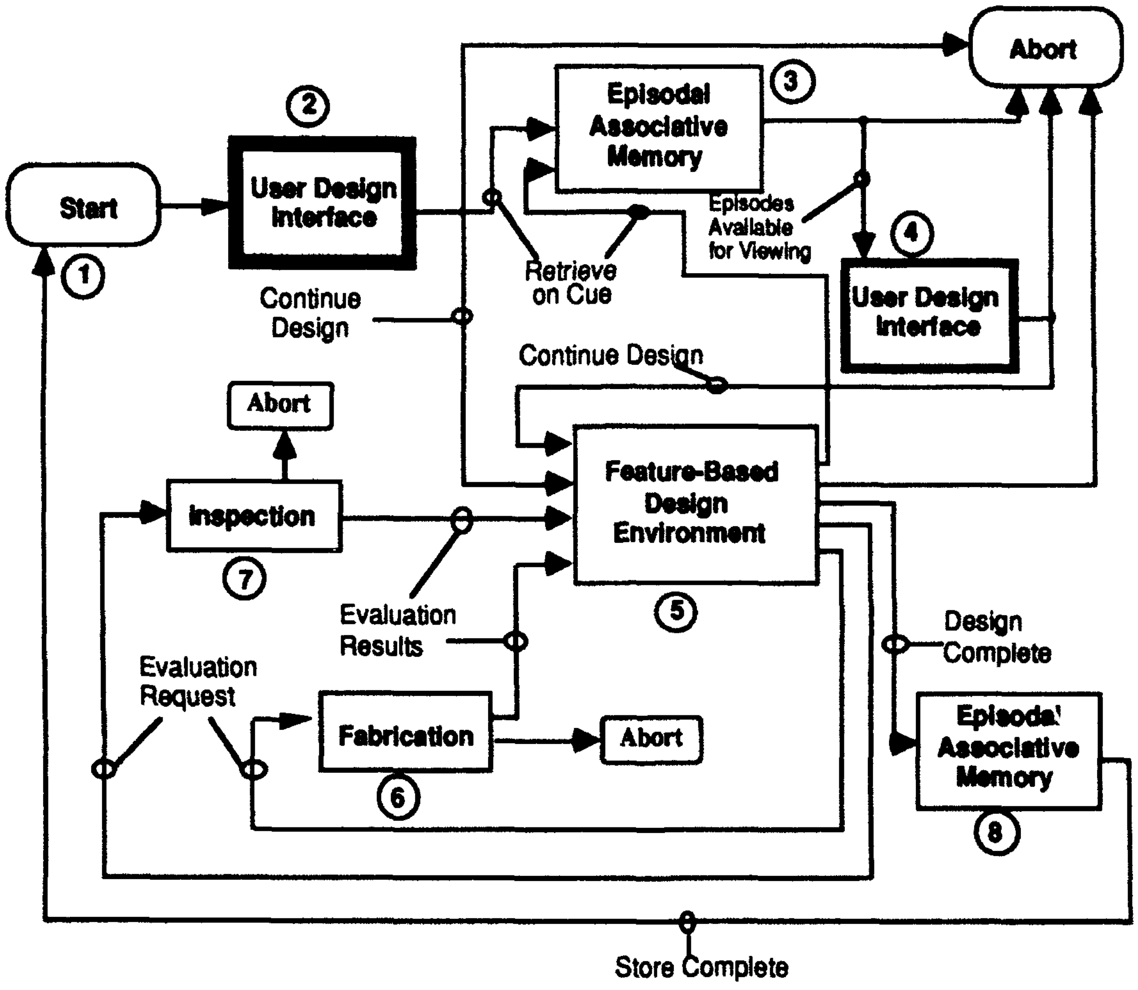


Figure 2. A possible scenario for use of the Rapid Design System (RDS)

Because of this vision of how a designer might prefer to function, we hypothesize that it is important that a designer be able to have access to large stores of previous designs relevant in one aspect or another to the design task that will be confronted.

Conceptually each "design" of a part is stored in the form of a flat file as illustrated schematically in Figure 3. The information contained in that file encompasses more than just the mechanical design of the part. The main categories of information and the items within each category are listed in Figure 3.

In actuality, such information is stored in mass secondary storage. In secondary storage, the designs can be stored in any one of a variety of alternate forms, in relational data bases for example. But in the recalled mode, the design is in the form of a flat file. The Episodal Associative Memory functions as an associative indexing interface so that a cue in the form of, for example, a geometrical design would automatically cause the appropriate pointers to be activated. The appropriate information is then retrieved for display and examination.

This foregoing account of the actions of the RDS associative memory is an accurate one of the currently implemented version, but it does not provide insight into the direction of the research. A wider view of the conceptual design of the RDS-EAM and of other advances made in supporting research can be obtained from the material of Section 4.

D0 Header Features • Host Assembly • Location • Part Function • Loading • Environmental • Materials • Critical Tolerances • Qualitative Topology • Qualitative Geometry	D2 Fabrication Features and Plan { Fabrication Feature (n) fixtures and positioning tools rate .... } : : :	Other Fields
D1 Design Geometry Features { Design Feature (n) function topology geometry dimensions tolerances .... } : : :	D3 Inspection Features and Plan { Inspection Feature (n) inspection mode inspection path inspection geometry .... } : :	Other Fields

Figure 3. Virtual "Flat" file storage format of design information

3. THE FEATURE-BASED DESIGN ENVIRONMENT

The Feature-Based Design Environment (FBDE) is an interactive, graphical environment for creating, manipulating, and editing a feature-based part model.

Motivation

Much interest has been focused recently on "design with features" [1,2,3,4]. In the context of computer aided design, features are "chunks" of the total part design which have some special meaning to the designer. Feature-based design is based on the premise that designers should be able to specify a part in terms of features which are meaningful to them instead of having to specify geometric primitives. The most obvious type of features is that of "form features." Form features represent certain geometric configurations on the surface of the part. Examples of form features include holes, slots and ribs. These various features have entered the designers' vocabulary either because they commonly arise in manufacturing (e.g., a hole is the result of a drilling operation) or because they have some special function (e.g., a rib is used to strengthen a thin section of a part).

Terminology

The RDS has been developed using object-oriented programming techniques. The following definitions come from the object-oriented programming world:

A *class* is a definition of data structure, similar to a record in a traditional programming language. A class contains *slots*, corresponding to components of a record in a traditional programming language. An *instance* of a class is a particular entity conforming to the class definition. There may be many instances of the same class existing at one time. A *method* is a specialized function which knows how to operate on instances of a particular class.

The *part model* is the collection of data which represents a part within the design system. Within the RDS, each feature type is represented by a class. The part model then consists of instances of these classes.

Design Paradigm

All designs begin with a distinguished feature called the "starting feature." The starting feature currently can be either a rectangular block, cylinder, or extrusion of an arbitrary two-dimensional profile. The designer designs by adding feature instances to the part model. The set of currently available features is shown in Figure 4. Features can be categorized into three types based on how they modify the geometry. *Positive features* (e.g., ribs) represent an addition of material, *negative features* (e.g., pockets) represent the removal of material, and *modifier features* (e.g., fillets) modify existing geometry, and can either cause addition or removal of material.

Each feature instance has a number of *attributes* which must be given values in order to fully specify the feature. Attributes are used to specify the internal dimensions of a feature (e.g., the radius and height of a cylinder), the relationship of a feature to other features, and other miscellaneous items (e.g., whether a hole is threaded). Each positive or negative feature has a *basic geometry*. For example, the basic geometry of a rectangular pocket is a block; the basic geometry of a hole is a cylinder. The basic geometry of a part is specified in a local coordinate system and is generally centered around the origin. The attributes of the feature which are used to specify its relation to other features can then be used to determine the position of the geometry feature in the world coordinate system.

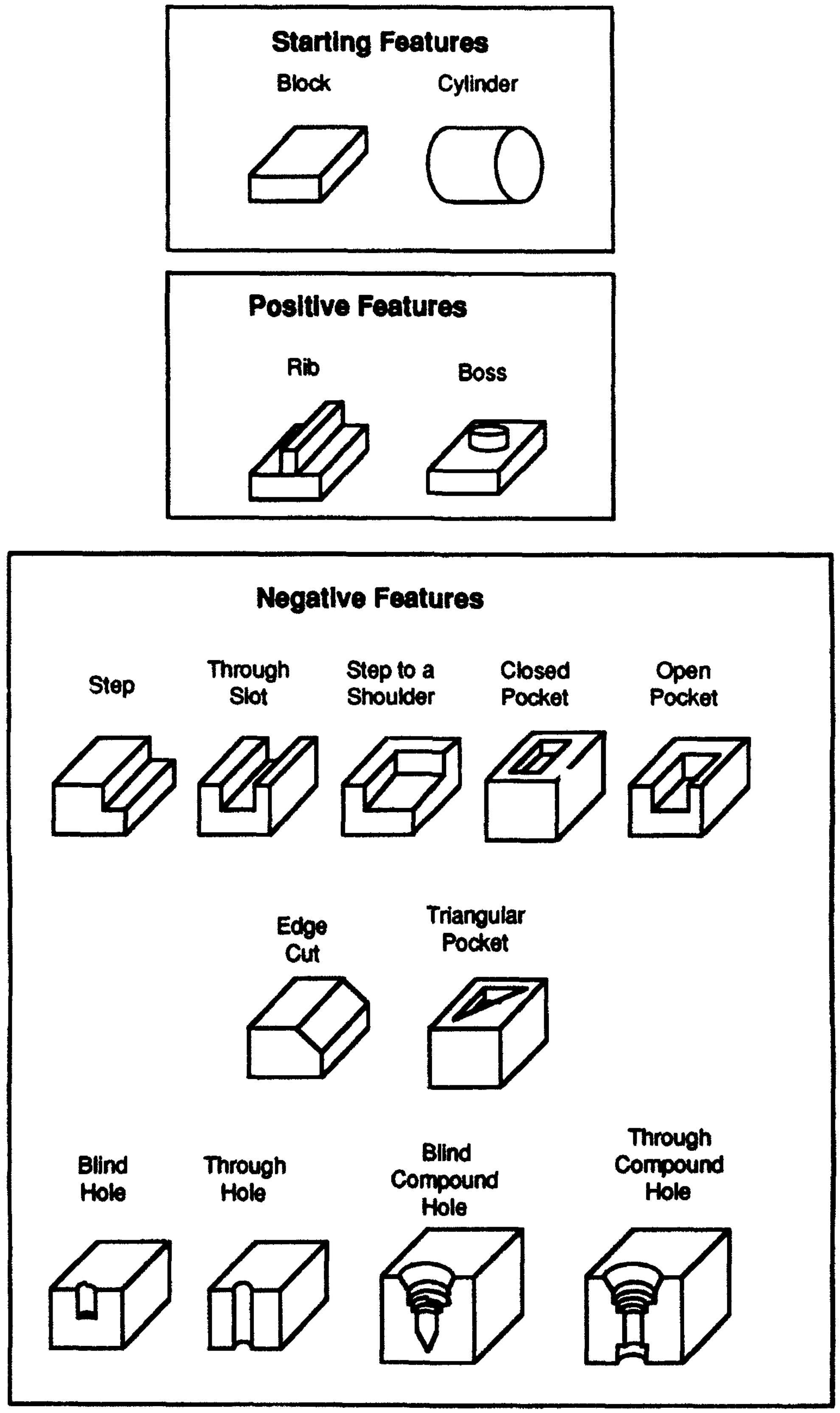


Figure 4. Features currently supported by the FBDE

Evaluating the Part Model

In order to determine the shape of the part, the part model must be *evaluated*. This is done within the solid model using Boolean operations: starting with the "starting feature," adding the material of the positive features, and removing the material of the negative features. The algorithm in pseudocode is as follows:

```
algorithm evaluate(part_model)
result ← transformed_geometry(starting_feature of part_model);
remaining_features ← sequence of all features of part_model except the starting_feature;
for each feature F in remaining_features do
        if F is a positive feature then
                result ← result ∪ transformed_geometry(F);
        elseif F is a negative feature then
                result ← result – transformed_geometry(F);
        elseif F is a modifier feature then
                result ← result modified by F;
        endif;
endfor;
return result;
end evaluate;
```

In the above algorithm, transformed_geometry(F) represents the basic geometry of F, transformed to the world coordinate system. "$\cup$" represents the set union operation, and "–" represents the set difference operation.

Positioning of Features

As mentioned earlier, most features have a basic geometry consisting of either a rectangular block or a cylinder. It might seem that there is no need to have more than one feature with any given basic geometry. To see why this is not the case, consider the case of a pocket and a step to shoulder, which both have a block as their basic geometry. The difference is that the pocket normally has four walls, while the step to shoulder has two walls, and thus can be thought of as modifying a *corner* of the geometry. Referring to Figure 5, we can give the location of the pocket by specifying surface A, and distances d_1 and d_2. On the other hand, we can determine the position of the step to shoulder by specifying surfaces B, C and A. This leads to the notion of "attachments."

A feature *F* is said to be *attached* to a surface *S* if one of the surfaces of (the basic geometry of) *F* is constrained to lie on *S*. In the case of negative features such as holes, pockets, and slots, *S* represents the surface of the material from which the feature is cut. In the case of positive features such as ribs and bosses, *S* represents the surface to which the feature is "glued." Note that this cutting or gluing operation is a logical operation which helps to conceptualize the shape of the part; there is not a requirement that the part actually be manufactured this way.

A surface to which a feature is attached is called an "attachment surface." Each positive or negative feature has at least one attachment surface. Some features have more than one attachment surface: a through hole has two (the top and bottom), and as we have already seen, a step to shoulder has three.

In the present implementation, an attachment surface is assumed to be a face of another feature instance. Therefore, to select the attachment surface, we can select an "attachment object" (a feature instance) and a face of (the geometry of) that object, called the "attachment face."

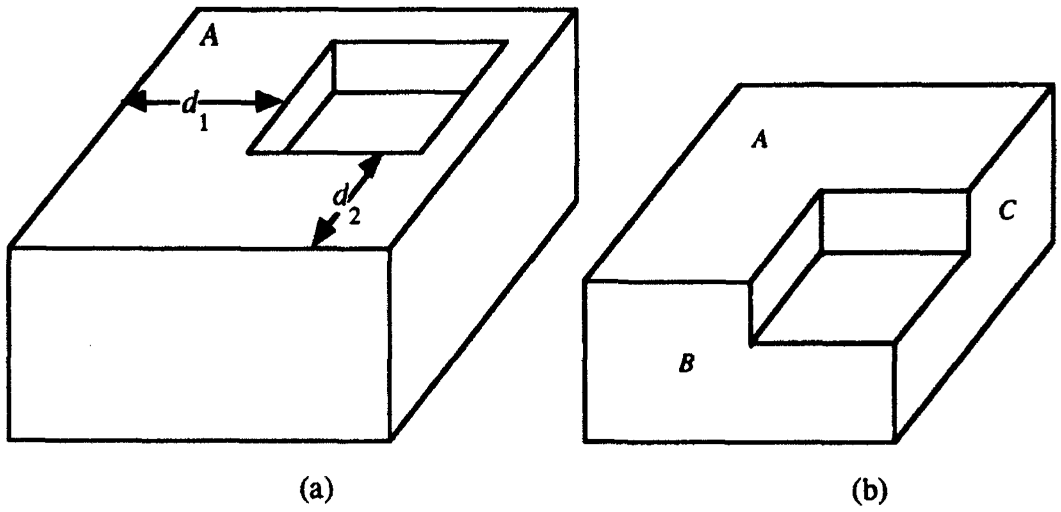


Figure 5. Positioning of: (a) A slot (b) A step to shoulder

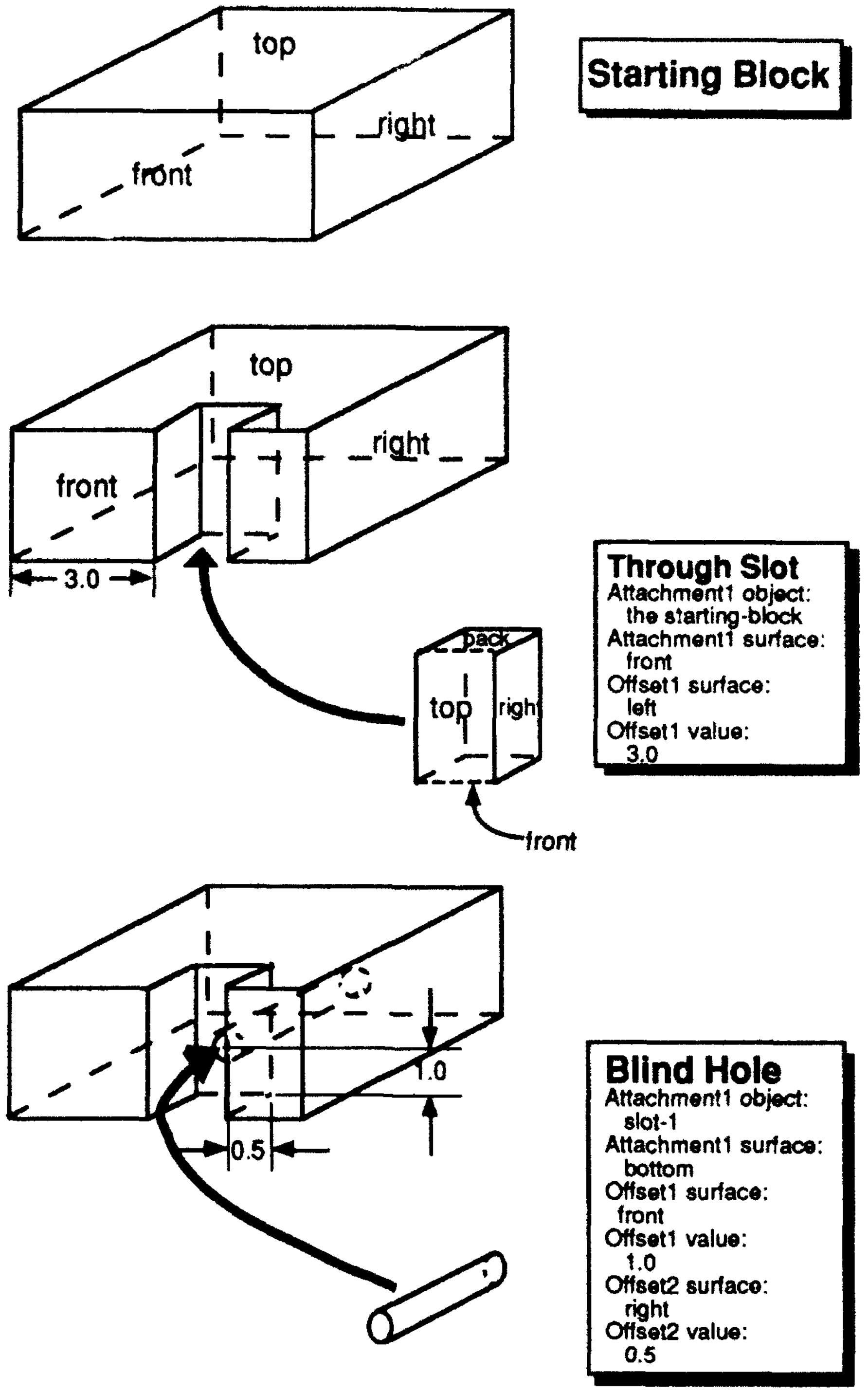
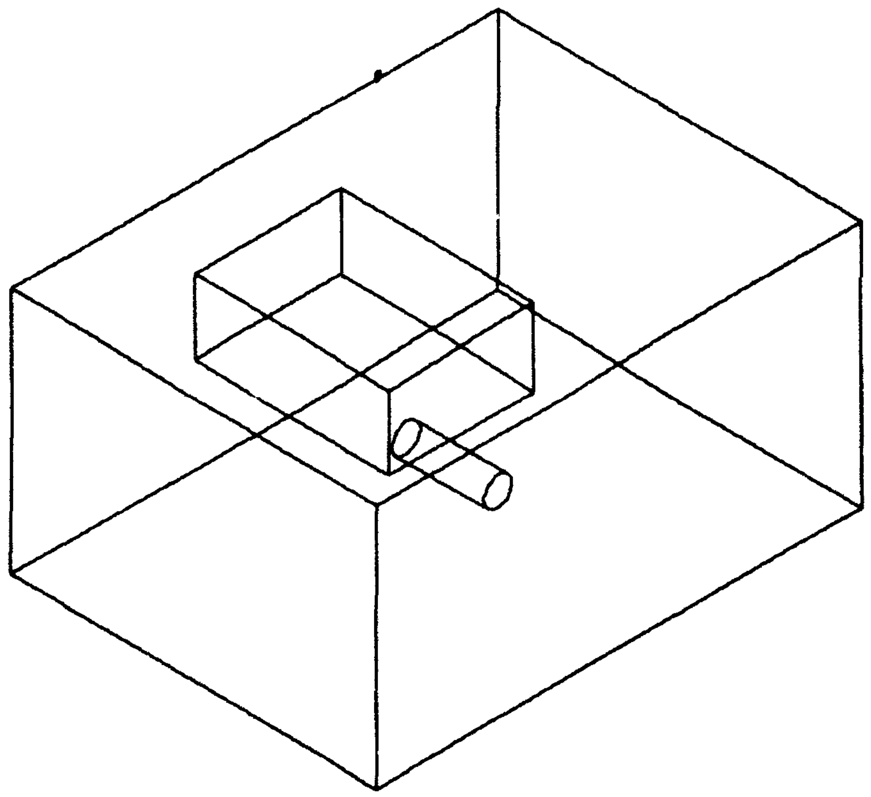


Figure 6. Starting block with a through slot attached to it,then a blind hole attached to the through slot



NAME	STARTING-BLOCK
SUPERIOR	(THE PART-MODEL)
DEPTH	20.0
WIDTH	15.0
HEIGHT	10.0
DRAW-COLOR	WHITE
FEATURE-TYPE-NAME	STARTING-BLOCK-CLASS
CONSTRAINT-CHECK-METHODS	()
FEATURE-CLASS	D1
VOLUME-TYPE	POSITIVE
STABILITY-TOLERANCE	0.5

Figure 7. A part design and its representation

DEFAULT-TOLERANCE	0.01
NAME	PKT-2
SUPERIOR	(THE PART-MODEL)
DEPTH	6.0
WIDTH	8.0
HEIGHT	3.0
DRAW-COLOR	WHITE
FEATURE-TYPE-NAME	POCKET-FEATURE
OFFSET1	3.2
OFFSET2	4.3
ROTATION	0
CONSTRAINT-CHECK-METHODS	(IS-ATTACHED-CONSTRAINT IS-CONNECTED-CONSTRAINT MIN-HEIGHT-CHECK MAX-HEIGHT-CHECK MIN-WIDTH-CHECK MAX-WIDTH-CHECK MIN-DEPTH-CHECK MAX-DEPTH-CHECK)
ATTACHMENT-OBJECT-LIST	(THE STARTING-BLOCK)
ATTACHMENT-FACE-LIST	(:TOP)
OFFSET1-OBJECT	(THE STARTING-BLOCK)
OFFSET1-FACE	(:LEFT)
OFFSET2-OBJECT	(THE STARTING-BLOCK)
OFFSET2-FACE	(:FRONT)
FEATURE-CLASS	D1
PICKABLE-CONTEXTS	(FBDE)
DISPLAY-CONTEXTS	(FBDE MAN EAM IPEM)
VOLUME-TYPE	NEGATIVE
CORNER-FILLET-RADIUS	0.05
BOTTOM-FILLET-RADIUS	0.05

NAME	TH-1
SUPERIOR	(THE PART-MODEL)
DRAW-COLOR	WHITE
HEIGHT	3.8
FEATURE-TYPE-NAME	PERP-THROUGH-HOLE-FEATURE
OFFSET1	2.0
OFFSET2	5.0
ROTATION	0.0
CONSTRAINT-CHECK-METHODS	(IS-ATTACHED-CONSTRAINT IS-CONNECTED-CONSTRAINT BOTTOM-FACE-IS-VALID MIN-DIAMETER-CHECK MAX-DIAMETER-CHECK MIN-DEPTH-CHECK MAX-DEPTH-CHECK)
ATTACHMENT-OBJECT-LIST	((THE STARTING-BLOCK) (THE PKT-2))
ATTACHMENT-FACE-LIST	(:RIGHT :RIGHT)
OFFSET1-OBJECT	(THE STARTING-BLOCK)
OFFSET1-FACE	:TOP
OFFSET2-OBJECT	(THE STARTING-BLOCK)
OFFSET2-FACE	:FRONT
FEATURE-CLASS	D1
PICKABLE-CONTEXTS	(FBDE)
DISPLAY-CONTEXTS	(FBDE MAN EAM IPEM)
VOLUME-TYPE	NEGATIVE
DIAMETER	1.2

Figure 7. **A part design and its representation (con't)**

Depending on the feature, there will be 0, 1 or 2 translational degrees of freedom after the attachment surfaces have been specified. This translation will be on the attachment surface. Figure 5 illustrates cases with 0 (part (b)) and 2 (part (a)) degrees of freedom. The translation amounts (i.e., d_1 and d_2) are called *offsets*. Offsets may be specified with respect to any edge which lies on the attachment face.

The use of attachment and offset faces is illustrated in Figure 6.

A sample design and its internal representation are illustrated in Figure 7.

Table 1: Geometric Dimensioning and Tolerancing (GD & T) Features

simple datum
compound datum
datum reference frame
flatness tolerance
circularity tolerance
cylindricity tolerance
straightness tolerance
angularity tolerance
parallelism tolerance
perpendicularity tolerance
concentricity tolerance
position tolerance
circular runout tolerance
total runout tolerance
location dimension
location tolerance

Dimensioning and Tolerancing Features

Form features specify the ideal geometry of the part. In order to specify the allowable variation of a manufactured part (which can never be perfect) from the ideal, dimensioning and tolerancing features are used. The set of dimensioning and tolerancing features used in the RDS is based directly on the information model contained in the draft PDES/STEP tolerance model [5], which in turn is based on the U.S. and international standards for Geometric Dimensioning and Tolerancing (GD&T) [6,7]. The GD&T features currently supported are listed in Table 1.

Constraint Checker

Incorporated into the FBDE is a mechanism for detecting constraint violations and reporting them to the designer. There are many different ways in which constraints arise in a CAD system. Some of these are listed below.

Topology constraints ensure that a feature is positioned so that it mates properly with the rest of the features in the part model. Any feature representing a negative volume must be positioned over a positive volume. This is illustrated in Figure 8, where the shaded area represents the cross section of a positive feature (e.g., block), and the empty rectangle represents a negative feature (e.g., slot). In (a), the slot is completely contained within the block. In (b), the slot is hanging in space, which is not meaningful. An ambiguous case is (c), where the slot feature has been positioned partially off the end of the block. Some researchers allow this as a method for creating a step feature. Our constraint mechanism considers it an error worth bringing to the attention of the user (who may choose to do it anyway).

Design rules can be represented as constraints. These rules can be classified into various groups, including rules that ensure manufacturability and inspectability. These constraints will depend on the equipment available for manufacturing and inspection. The design rules are intended for rapid checking of a possibly incomplete or even inconsistent design, so that the designer does not have to exit the FBDE and enter the manufacturing planning or inspection planning modules in order to check the feasibility of the design. These rules are not intended to do a complete check for manufacturability or inspectability; the fabrication or inspection planner must be used for that purpose.

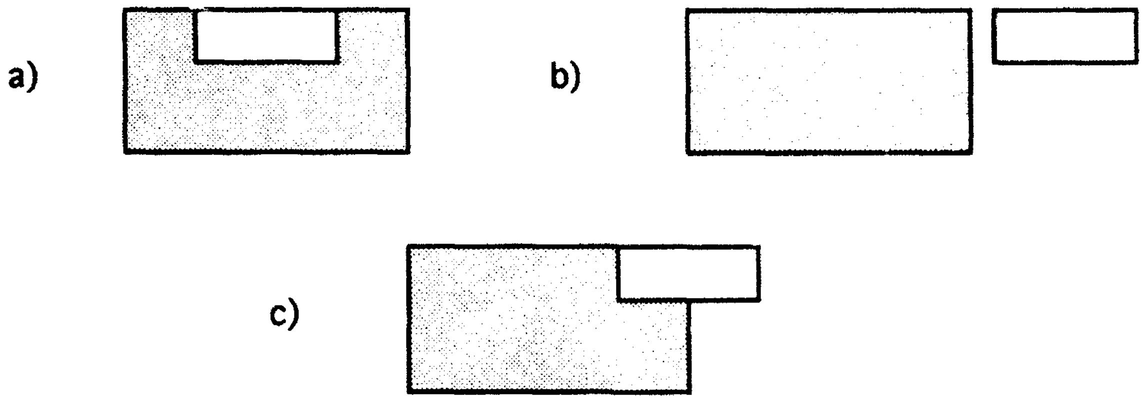


Figure 8. Topology constraint cases

Shop practice constraints represent the design standards that exist within an organization, including corporate, ANSI and international standards.

Consistency constraints ensure that different aspects of the design agree with each other. For example, a tolerance zone specified by a GD&T callout is intended to bound the position of a manufactured (i.e., imperfect) instance of a particular surface on the part. The constraint checker will ensure that the designed position of the surface falls within the bounds of the GD&T tolerance zone (this must be checked since in our data structure the GD&T callouts and the design-with features are separately manipulatable objects).

Completeness constraints ensure that all required information is present; for example, that all necessary attachments have been specified.

Constraint Mechanism

At certain stages of specification of a feature, such as when the user requests that the feature be redrawn, or when the feature is incorporated as part of the current design, the constraint checking mechanism is invoked.

The constraint checks are performed by sections of code called constraint check methods. Each class of design-with feature has a list of the constraint check methods germane to it, and these are applied sequentially to the feature instance. When a constraint check runs, it determines whether its constraint has been violated. If so, it notifies the constraint manager.

The constraint manager is a globally accessible programming object that provides services to the design environment and has as persistent data a list of constraint violation events (CVEs) currently applying to the design under construction. Each CVE is composed of information provided by a constraint check method when it has detected a violation. Services of the manager include the addition of a new CVE to the list, deletion of an existing CVE, or requests to recheck some subset of the existing CVEs. The manager is also responsible for maintaining the list of current constraint violations, which is displayed to the designer.

Data for the constraint check manager include:

- a list of currently active CVEs (the `constraint-list`)
- a list of strings currently displayed to the user as active CVEs (the `displayed-list`)
- a window for displaying the constraint violation messages to the user.

The CVE (Figure 9) is an important repository of information about the current state of the design. Each CVE is an object that contains:

- a string with enough information to describe the problem to the designer (the `text-description`)

- a list of design-with features which might, if changed, necessitate a re-evaluation of the constraint (the `affected-object-list`)

- a code to properly invoke the constraint checker that had originally detected the violation and perform the re-evaluation (the `recheck-incantation`).

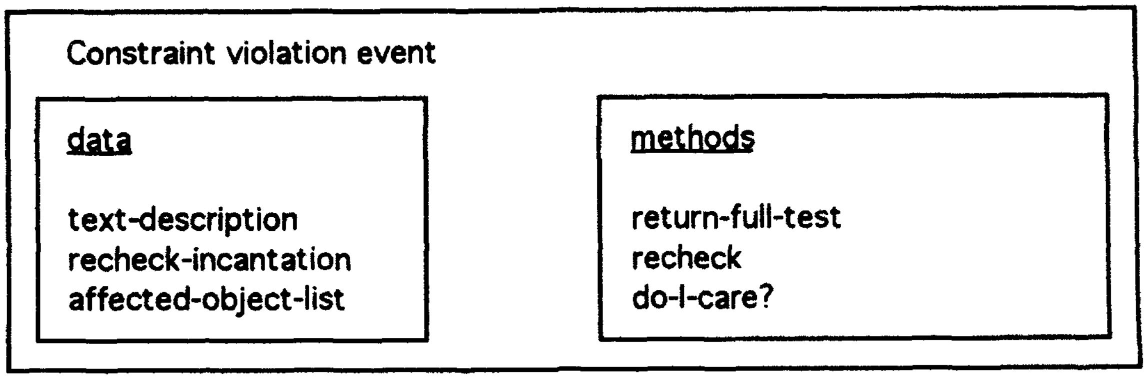


Figure 9. Constraint violation event

Constraint checking process. Constraints are rechecked for a feature whenever a change is made to its attributes. The mechanism is invoked by sending the `recheck-for-object` message to the constraint manager. The constraint manager first obtains a list of all CVEs that involve the feature being rechecked by sending `do-I-care?` messages to all CVE objects in the constraint list. All of the CVEs that respond affirmatively then receive recheck messages, which causes them to recheck themselves using the code constrained in their `recheck-incantation` data. If the CVE is shown to no longer apply, it is removed from the constraint list by the constraint manager.

After rechecking the existing CVEs, the constraint manager then steps through the `constraint-check-method-list` for the feature, sending each name in the list as a message. As stated before, these methods are then responsible for notifying the constraint manager if a new CVE has been discovered. The constraint manager checks for duplication as the CVE is reported—duplicates are not inserted into the constraint list.

References

1. Dixon, J.R., 1987, Expert systems for mechanical design: Examples of symbolic representations of design geometries. *Engineering with Computers*, 2, pp. 1-10.

2. Jakiela, M. and P. Papalambros, 1987, Design and implementation of an intelligent CAD system. American Society of Mechanical Engineers Design Automation Conference, Boston MA, September 27-30.

3. Libardi, E.C., J. R. Dixon, and M. K. Simmons, 1986, Designing with features: design and analysis of extrusions as an example, ASME Paper No. 86-DE-4.

4. Luby, S.L., J.R. Dixon, and M.K. Simmons, 1986, Designing with features: Creating and using a features database for evaluation of manufacturability of castings, *Proceedings of the ASME Computers in Engineering Conference*, Chicago IL.

5. External Representation of Product Definition Data, document number STEP DP 10303, Part 47, 1990.

6. Dimensioning and tolerancing, American National Standard Y14.5M–1982, American Society of Mechanical Engineers, New York NY, 1982.

7. Tolerances of form and of position, International Standards Organization, Parts I–III, documents R1101–1969, DIS2691–1972, and R1660–1971.

4. THE EPISODAL ASSOCIATIVE MEMORY

Overview and Summary

The function of the RDS Episodal Associative Memory (EAM) is to help the human task performer make use of accumulated knowledge and experience in the performance of new tasks; whether it be creating a new design, specifying a new fabrication plan, or detailing a new inspection procedure.

Work on the EAM has consisted of three related efforts; these being the conceptual design effort, a technology transfer effort, and a specific RDS-EAM implementation effort. These related but separate efforts are described briefly in the following.

The conceptual design effort has resulted in an understanding and an articulation of what functionalities are required if a memory is to perform in the intended EAM manner with the desired EAM capabilities. This understanding is valuable for this program and for all future intelligent systems research. Currently in literature, for example, associative memory is understood primarily in terms of simple clustering and retrieval. Our results indicate that such isolated capabilities are ineffective and largely irrelevant to serious augmentation of task performance. A realistic RDS-EAM conceptual design is described in the following.

The technology transfer effort consisted of making the RDS technology available to a program aimed at implementing a related computer-based task performance enhancement system named the Rapid Foundry Tooling System (RFTS). It had been decided that the RFTS would also be feature-based and memory-driven with use of the EAM scheme. This technology transfer effort proved very beneficial to development of the RDS-EAM because the task environment of the RFTS was rich in data and detail and the RFTS task provided data for demonstrating progress achieved in the RDS domain, but not demonstrable for lack of valid detailed examples. The RFTS effort was also valuable in illustrating the fact that different features become important depending on the nature of the task. In machining, the relationship of features to surfaces is important because of considerations of fixturing and tool set-ups; but in the RFTS task environment, there is little concept of a "starting block," much less that of surfaces. Instead the primary concern is with the direction of flow of the molten metal, the direction of cooling, and the relative sizes of sections. The RFTS task environment has enabled us to demonstrate true cross-context associative recall relating tasks, problems, causes, and cues. These matters are discussed briefly in the following paragraphs.

Finally, a specific software module, the EAM, has been implemented for the RDS itself. Although it is of limited capability, cognitively speaking, it is quite powerful and robust in its role as a utility module of the RDS. It assumes that a design is adequately described in terms of its qualitative geometry, consisting of a listing of all the geometrical features on all the surfaces of the "starting feature." The EAM module interacts seamlessly with the FBDE module. It can store a body of designs in the form of several clusters. It can add a new design to the memory or retrieve a previous design on the basis of similarity with the current design-the-cue. It can display the retrieved design as well as the current design. The overall system architecture is sound and the details of storage and search are largely irrelevant, because these can be changed at will. The system design is modular and object oriented, and so the present module can be integrated with other modules of the true EAM as it is evolved either in the RDS or in the RFTS domains.

Theory and Conceptual Design

The integrated design-and-fabrication task has traits of that which in humans would be called cognitive perception. In general, this type of task is difficult for humans as well as for machines but especially for machines.

It is known that for humans there are two modes of information processing: Perception and Action, and Language and Reasoning. In design and fabrication there is certainly a great deal of perception, but the task is not purely in the realm of Perception and Action. There is much variation in what is perceived. Rarely are designs exactly alike, and similarity is more meaningfully judged on abstracted bases rather than on the basis of surface-level features. Decisions specify what type of action is warranted, and these also have to be translated from the language-and-reason domain to the perception-and-action domain through the process of instantion.

In supporting basic research, we endeavored to find powerful means for uniting conventional AI methodology and pattern recognition/neural-net computing methodology. Substantial progress was made, and use of a unified information representation scheme makes it at least possible to discuss a conceptual design for a single realistically proficient EAM.

As shown in Figure 10, information processing starts with the concept memory (A), a component of the EAM. This is where relationships between *function* and *form* would be stored. Information on "task context" or "mission objectives" constitutes the cue to this memory. Search of the associative memory results in directions on how the perception inputs are to be processed and formatted and which of the resulting "features" are considered to be relevant. This information is passed on to the preprocessing and formatting module (B). That module "perceives" the proposed design or plan in the input form and processes the information to yield information in appropriate *feature* form.

Those features still need to be formatted and translated into a pattern representation. This task is carried out in the representations module (C). For example, a *set* of features is represented differently from a *sequence* of features and an episode may consist of a structure of sets or of sequences.

The properly formatted patterns can now be used as a cue for search of the appropriate associative memory. At step (D), both the mode of search and the retrieved results are influenced by the need of the task performer and related ideas of what constitutes similarity.

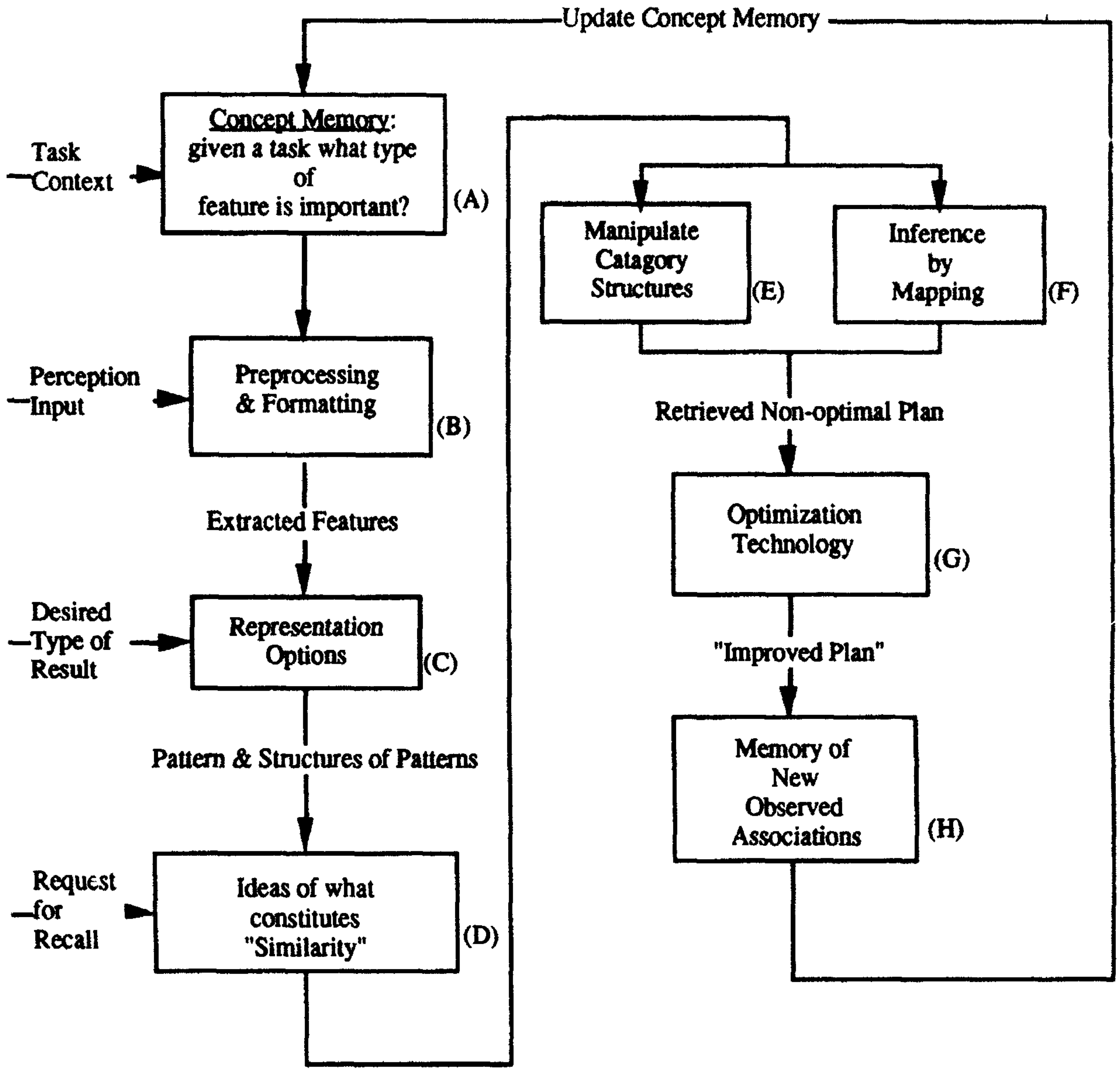

Update Concept Memory
Concept Memory: given a task what type of feature is important?
(A)
Task Context
Perception Input
Preprocessing & Formatting
(B)
Extracted Features
Desired Type of Result
Representation Options
(C)
Pattern & Structures of Patterns
Request for Recall
Ideas of what constitutes "Similarity"
(D)
Manipulate Catagory Structures
(E)
Inference by Mapping
(F)
Retrieved Non-optimal Plan
Optimization Technology
(G)
"Improved Plan"
Memory of New Observed Associations
(H)

Figure 10. Schematic illustration of the conceptual design of the EAM

The cognitive part of the task is carried out principally in modules (E) and (F) where extended tracing of structures of categories, inductive analogical reasoning, and deductive reasoning are carried out.

The end result of these activities is usually a nonoptimal design or plan. That plan can either be improved principally by means of human input or with the aid of an optimization module (G).

The improved plan and subsequent support or denial of that plan through use of experience are incorporated into existing associative memory structures.

The RDS-EAM effort has stimulated us to make significant progress in the enabling technologies needed for implementing the modules (A) to (G). Specifically, the advances include:

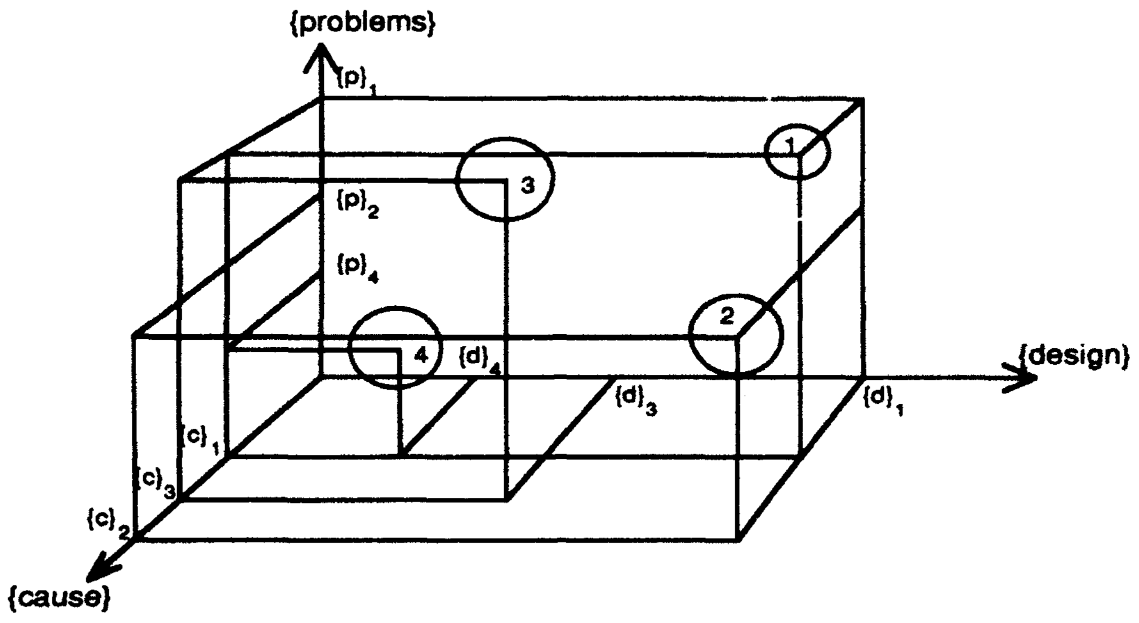
- a coarse-coded form of the Functional-Link net to provide uniform representation of diverse forms of patterns;
- patterns accommodated may be numeric or linguistic symbolic;
- patterns accommodated may be sets of nontemporal features or temporal relations squences of features;
- the same pattern space can be used for categorization, supervised learning of functional mapping, and optimization;
- categorization procedures which result in tighter and more meaningful clusters and also in hierarchical structures of clusters;
- learning of concepts.

These matters will be reported in length in subsequent reports and in technical publications.

Technology Transfer to the Rapid Foundry Tooling System (RFTS)

The RFTS is a system similar to the RDS but for a different task domain, that of helping pattern makers in metal-casting practice. The RFTS system is being developed by a commercial enterprise, AI WARE, Inc., for Kelly Air Force Base, San Antonio TX, starting with RDS technology.

In the case of RFTS, associative structures have been formed registering previously experienced associations between *design*, *problems*, and *cause*. This is illustrated schematically in Figure 11.



Retrieving Associations

Clusters formed from previous experience.
Retrieval on basis of {design} (partial cue) yields information on likely problems and causes.

Figure 11. Cluster cross-reference grid

For example, in Figure 11, a design similar to $\{d\}_1$ when used as a cue retrieves clusters 1 and 2 yielding the information that on previous occasions where problems $\{p\}_1$ and $\{p\}_2$ had been experienced, and that these had been diagnosed as being due to causes $\{c\}_1$ and $\{c\}_2$, respectively.

Similarly and more interestingly, further associative recalls would have yielded the information that the cause $\{c\}_1$ can also have problems of type $\{p\}_4$ in designs of type $\{d\}_4$ and so on. Using this type of prompting, the human operator can make large and seemingly unconnected jumps in context to get further insight between designs and problems and to arrive at ways for improving the one and to avoid the other.

Currently Implemented EAM Functionalities

A small part of the conceptual design of the RDS-EAM has been implemented for trial use together with the FBDE.

The functionalities and system architecture of that part of the EAM are described in this subsection. In detail, these are different in detail from what has been specified in conceptual design, but evolution from one to the other has been planned for. In the meantime, basic mechanisms for storage and associative retrieval of designs are now available for use and evaluation.

The current EAM can be described in terms of three "panes" at the top of an EAM window. The contents of the window display may differ considerably depending on the which pane has been activated.

The three main panes are shown in Figure 12. They are icons of the three states of the EAM.

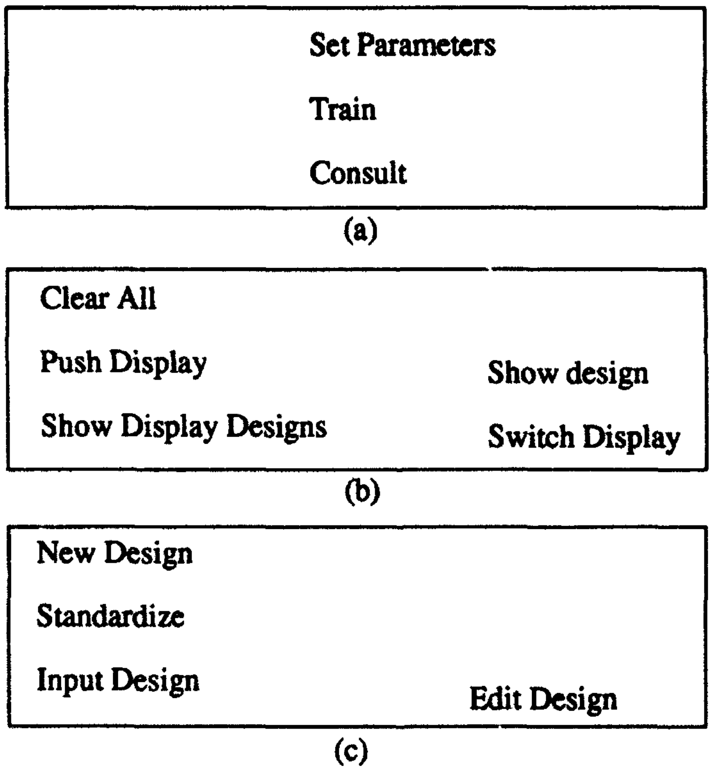


Figure 12. EAM window icons

1. The FBDE pane: This has a group of buttons which address the tasks of creating a new design, storing a part, or editing a design already in the EAM data base.

The buttons and the corresponding functionalities are as follows:

New Design:

Takes the user to the FBDE layout-the environment for creating new designs. (Figure 13).

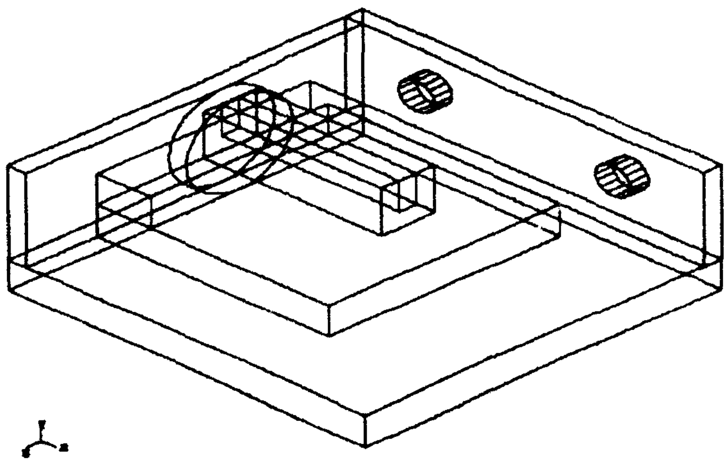


Figure 13. The FBDE accessed through EAM

Standardize:

If two parts are to be compared for similarity, it is necessary to have a standard "point of view." *Standardization* reorders faces so that: Height $\leq$ Depth $\leq$ Width and Top has more features than the Bottom, Left than Right, and Front than Back. To see how a part looks when two designs are being compared, we would standardize the part.

The part in Active Display is standardized and displayed in the Passive Display. However, we find that some designers do not like this to be done automatically, and so the standardization can also be accomplished in the background for assessing similarity in a manner transparent to the user. In other words, some designers find it irritating to have a design, which they have a certain mental image, retrieved and displayed to them in another unexpected orientation; even though the latter might be considered to be the standardized orientation. Accordingly, the preference is to store and retrieve as indicated by the designers; but match in automatically standardized form.

Input Design:

With use of this button, it is possible to retrieve a design in memory by specifying some combination of data base name--a part-type or part-name.

Edit Design:

This allows for porting of the part in the Active pane to the FBDE so that editing actions can be carried out.

2. The DISPLAY pane: The set of buttons here controls what is shown in the two display windows: The Active Display and the Passive Display (Figure 14).

The Display pane buttons determine what is shown in the Active and Passive panes. Only the part in the Active pane (upper LEFT) may be edited; for example, add/delete features.

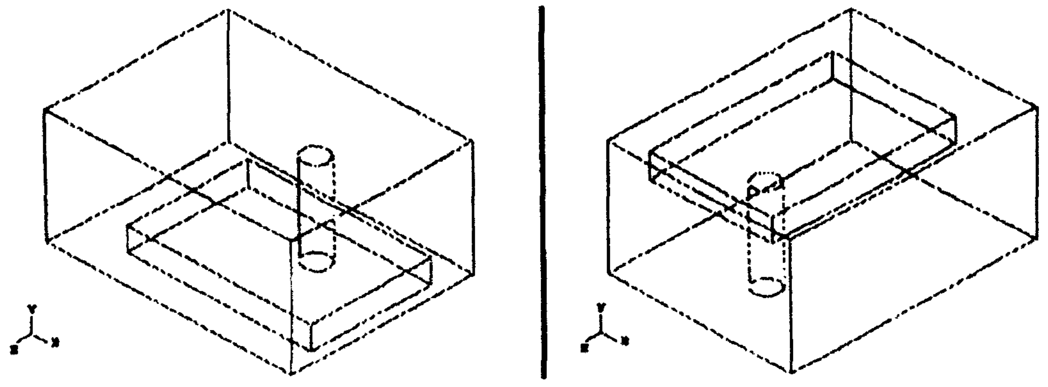


Figure 14. The EAM working environment

Various options available in this pane are:

Clear all:

Clears both active and passive panes.

Show Design:

To display current design in active display pane.

Push Display:

When clicked, design in active display pane is copied and displayed in passive display pane. The previous design in passive display pane is destroyed.

Show Display Designs:

Same as show design except both ACTIVE as well as PASSIVE designs are redrawn.

Switch Display Designs:

Changes the designs in ACTIVF and PASSIVE displays.

3. NNET pane: Storage and recall functionalities are activated by buttons on this pane.

The two principal functionalities in this pane are:

Train:

Refers to the formation of clusters, namely the action of grouping parts together on the basis of how much they resemble each other when described in standardized orientation. The measure of similarity is based on the number of common features they have in common on the various faces. The cluster radius is one of the train parameter described in the foll^wing.

Consult:

Refers to search of existing memories formed previously in "train" mode.

However, before we can be used in either consult or train mode, the user needs to have set the value of some parameters. The way to do this is through use of the set parameters facility.

Set Parameters

The two types of parameters are shown when we click on the button:

Consult:

This refers to the form of input that is going to be used as cue for the associative search. If the part just designed (and so, in the active display) is to be the cue pattern, the option PART-MODEL is chosen. On the other hand, if there is only an approximate idea of the desired pattern; for example, a part of the form of a flange with a rib attached to the top and four holes through the starting block from the top face to the bottom face, a more formal cue can be formed with use of the create-pattern facility as follows.

Create-Pattern:

Once this is selected, a pane containing the names of six faces (Top, Front, Left, Bottom, Back, Right) is displayed. For each face name, a list of all features is shown (Figure 15). For the example of a part with one rib and four through holes, the user fills in 1 for rib and 4 for through the holes.

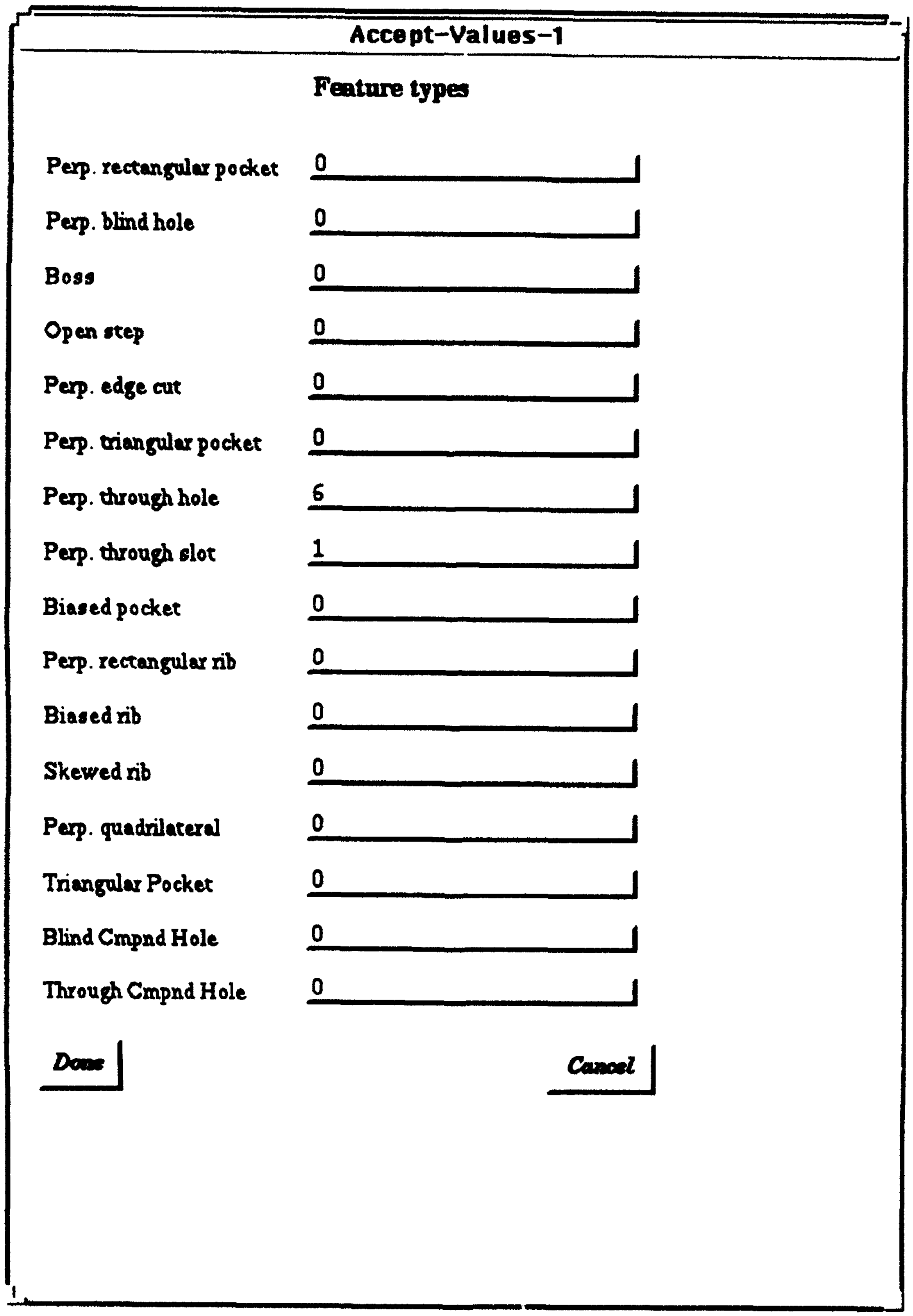
After the features have been specified on all the faces, "Done" is clicked.

Features are counted only once. For example, a through hole from the top surface to the bottom surface of the starting block would be counted only one as having been attached to the top surface. This is a result of an ordering of the list of faces the system considers in order: Top, Front, Left, Bottom, Back, Right.

Train:

Once the train option is selected, the parameters to be set are (Figure 16):

System Name:
A name for the trained system.

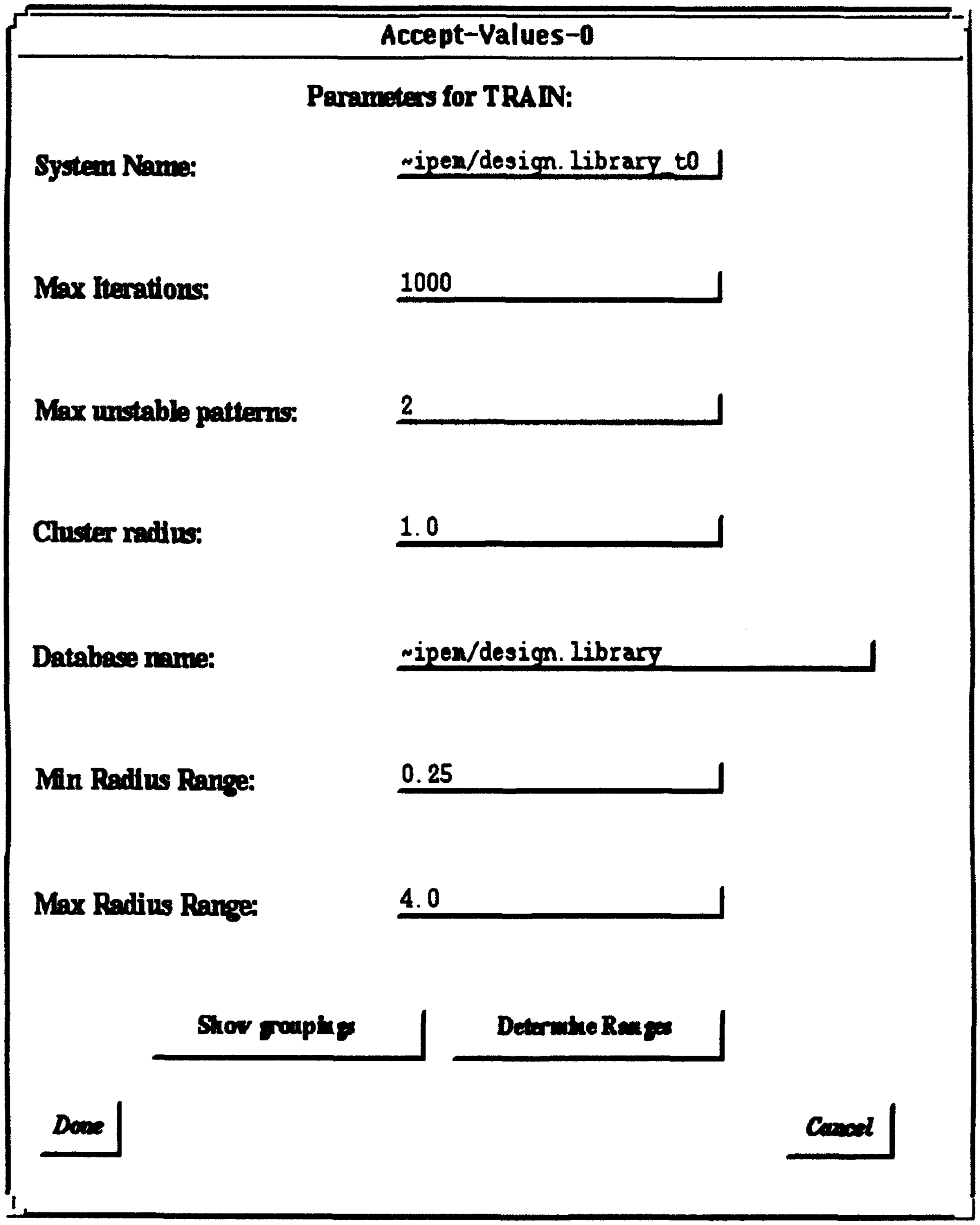


Accept-Values-1

Feature types

Perp. rectangular pocket	0
Perp. blind hole	0
Boss	0
Open step	0
Perp. edge cut	0
Perp. triangular pocket	0
Perp. through hole	6
Perp. through slot	1
Biased pocket	0
Perp. rectangular rib	0
Biased rib	0
Skewed rib	0
Perp. quadrilateral	0
Triangular Pocket	0
Blind Cmpnd Hole	0
Through Cmpnd Hole	0

Figure 15. Create pattern facility template for search cue in Consult mode


Accept-Values-0

Parameters for TRAIN:

System Name:	~ipem/design.library_t0
Max Iterations:	1000
Max unstable patterns:	2
Cluster radius:	1.0
Database name:	~ipem/design.library
Min Radius Range:	0.25
Max Radius Range:	4.0

Figure 16. Template for setting parameters values in Train mode

Max Iterations:

The training uses an iterative algorithm and can sometimes be time consuming. However, if the user is satisfied with a memory suitable for quick, nonoptimum searches, the number of iterations in training can be lowered by requiring the system to terminate its compuations after a set number of cycles. This could be used for "what if..." inquiries by the user.

Alternatively, increasing the number of maximum iterations usually results in more accurately structured memories.

Max Unstable Patterns:

"Unstable Pattern" refers to the dilemma the system faces with a part that it could classify in either of two clusters equally well: a given part C looks similar to part A and looks equally similar to part B, but part A and part B are clearly distinct. The term "Max Unstable Patterns" refers to the number of such patterns which can be tolerated in a result with for example, the cluster of part A and the cluster of part B both containing part C. When this parameter value is set to zero, all clusters will be distinct without any overlap.

Cluster Radius:

This is an abstract measure of the closeness of the parts in a cluster to each other. "Abstract" refers to the fact that there is no physical attribute of a part which relates directly to this measure.

If the cluster radius is set to nearly zero, any search would result, at most, in a retrieval of one design--namely itself--if it is in the data base. Otherwise, there is no associative recall. Setting it to a characteristically large value would place all parts in a single cluster and all designs would be retrieved regardless of cue. An intermediate value is suggested for a starting value.

Database Name:

This is the data base into which the parts to be clustered are stored. This is the same name that is referred to in "save part" and "set data base name" in the FBDE layout. This field must be valid to continue.

Train

If there is a set of parts in a data base and the user would like to carry out an associative search with respect to the data base, the user needs to first execute Train so that the system will have the associative memory structure necessary for the search.

Consult

Once the parameters have been configured (see Set Parameters above) and the memory has been trained the system is available for consulting. The consult mode is called up by clicking this button.

The retrieved part names will be displayed in the part selection pane (lower-right pane). The parts retrieved are arranged in such a way that the closest matches are near the top with closeness decreasing with the order down the list.

The list of retrieved designs may be scrolled by moving the mouse along the right-hand border (white bar) of the pane. Clicking on the triangular regions at the top of the border results in scrolling all the way to the top. Similarly, clicking on the lower triangle results in scrolling to the bottom of the list.

To look at a part, the user can point at the name of the part in the list and select it with a left click of the mouse. This will display the part in the Passive (upper right) pane.

Comments of Bibliographical Nature

This RDS-EAM research is concerned with the question of how remembrances of designs, fabrication, and inspection plans might be effected so that useful information can be recalled appropriately, as needed, on future occasions.

A multitude of issues arise if this seemingly straightforward task is attempted. In a sense this is also why this present work is related to a number of other efforts which overlap the present work in so far as objectives are concerned, but differ in approach and methodology.

To start with, Group Technology is well known to idustry. That practice advocates the grouping together of similar parts so as to reduce set-up times, tool changes, and so on. Description of parts in terms of features and recognition on the basis of *similarity* were investigated by Filho [1] using methods of pattern recognition.

The question of what "features" are valid ones has turned out to be a bit of a red herring, as we will explain.

Various agencies, technical societies, and committees labored over the years to develop standard procedures for describing mechanical parts in terms of features. The STEP/PDES model is the most recent outcome of such sustained efforts. With STEP/PDES, the Form Feature Information Model is spelled out in some detail and so is the underlying geometric representation. However, the Applications Features (for the various processes, such as machining, casting, etc.) are left to the task domain specialist with the proviso that it should be possible to map or relate the Applications Features to the Form Features. This also means that there is no "correct" set of features. Task performers deem some features to be more natural than others. Those are the features that researchers might find suitable for use in their intelligent systems.

Actually, selection of a feature set for a specific domain would fall within the scope of an AP, which is done by a committee, not an individual specialist.

This means that how designs and plans are to be described to computers needs to be considered carefully so that they might be remembered appropriately, in terms of groups of similar items, with varying criteria of similarity depending on whether functional, conceptual, or detailed design is of interest.

The present work is entirely compatible with the official stance of the STEP/PDES viewpoint as evidenced by the schematic illustration exhibited in Figure 10.

Aside from the Group Technology, pattern recognition research mentioned previously, there have also been Artificial Intelligence research efforts focused on design and process planning. These efforts also deal with the use of features, measures of similarity, questions of associativity and reasoning to establish analogic similarity. The latter type of similarity may be of a nature very different from that established by pattern recognition on the basis of surface level features.

Examples of reports on the use of Artificial Intelligence techniques in dealing with feature-based design include Zhao and Maher [2] on analogical reasoning for the design of buildings, Maher [3] on the synthesis and evaluation of preliminary designs, and Maher *et al.* [4] addressing the issue of creative design.

Examples of reports on feature-based Artificial Intelligence process planning studies include Ansaldi *et al.* [5] on integration of AI techniques, and CAD solid modeling for process planning; Descotte and Latombe [6] describing a problem solver that plans how to machine mechanical parts; Fridshal [7] on automatic generation of NC instructions from geometric models; Iwata and Sigimura [8] also on process planning; Nau and Gray [9] on the use of hierarchical knowledge clustering in process planning; Nau and Karithi [10,11] on feature interactions in process planning; Mantyla *et al.* [12], Mantyla and Opas [13], and van Houten *et al.* [14] all on feature-based process planning.

The EAM research is consonant with the objectives of these AI studies. Experience teaches that these AI systems, which tend to be primarily rule-based and utilize sequential processing of linguistic symbolic expressions, are suitable for dealing with reasoning and planning at higher conceptual levels but are limited in power when dealing with numerical or graphical information. Design and process planning are tasks of mixed nature involving both reasoning and large amounts of data. The present approach, therefore, augments AI techniques with neural-net computing and pattern-information processing methodologies.

Others have also reported on the possibility of using neural-net technology for the storage and retrieval of aspects of feature-based design. Examples of reports of such efforts are the MS Thesis by Kamarthi [15] on the application of neural networks for component design data retrieval, the report by Kamarthi *et al.* [16] on that same topic, and the discussion by Kumara and Ham [17] on the use of Associative Memory and self-organization in conceptual design. These are interesting accounts, but differ from the present approach in illustrating principally pattern matching on the basis of detailed graphics which tend to not capture the essence of "similarity."

Over the years, there have been struggles with the issues of associative memory, measures of similarity, and the question of how to manage remembering and recollecting on the basis of similarity [18]. There is a large body of literature on these matters.

This present research activity is influenced by the work of Minsky [19], who pointed out in his K-lines model of memory that "disposition" is more important than proposition. In other words, in associative recall it is not only the surface features of the cue which are of interest. Sometimes these surface features remind us of a certain theme or form a certain "disposition." It is the automatic formation and utilization of a dynamic memory capable of handling these matters which is challenging. In other words, as mentioned previously, the issue is not only associative recall but also abstraction accompanied by generalization and associative recall at different levels of abstraction.

The development of the EAM is also influenced by the Dynamic Memory ideas of Schank [20], by the episodic memory ideas of Kolodner [21], and by the Case-based planning approach of Hammond [22].

Although the present research is related to these various previous works, there are significant differences and innovations.

As in other works, self-organized neural-net associative memories are used to achieve storage and associative retrieval. In the software implementation, clustering and search are based on similarity defined in terms of Euclidean distances. However, in our research version, self-organization is hierarchical and a coarse-coded representation [23] of the situation allows for use of similarity measures which incorporate additional ideas from Tversky and Hutchinson [24].

The present work has made major advances in this matter by combining the connectionistic coarse-coding suggestions of Hinton and Touretzky with the Functional-Link net processing methodology of Pao and his colleagues [25-28]. The result is a technology capable of supporting information-connectionistic processing, in a unified manner, of pattern-formatted information in linguistic symbolic, as well as, numeric form. In addition, this new format supports neural-net computing of the forms of unsupervised learning, supervised learning, and optimization, all in one format--that of the flat net.

The net result of these advances is that it will be possible to describe designs and fabrication plans, which might be quite complicated (not just in the form of one-flat pattern), and store such information in associative memories. These memories can then be searched on the presentation of cues. These cues can be quite complex with embedded structures or can be partical cues. In addition, experience is constantly processed so that concepts are formed [28] and meaningful categorization is achieved.

REFERENCES

1. Filho, E.V.G., 1988, Computer-aided group technology part family formation based on pattern recognition techniques, Ph.D. Thesis, Department of Industrial and Management Systems Engineering, The Pennsylvania State University.

2. Zhao, F. and M.L. Maher, 1988, Using analogical reasoning to design buildings. *Engineering with Computers*, Vol. 4, No. 3, pp. 107-19.

3. Maher, M.L., 1989, Synthesis and evaluation of preliminary designs. *Proceedings of the Fourth International Conference on the Applications of Artificial Intelligence in Engineering*, p. 3-14, Computer Mechanical Publications, Southampton UK.

4. Maher, M.L., F. Zhao, J.S. Gero, 1989, An approach to knowledge-based creative design, *NSF Engineering Design Research Conference*, pp. 333-346.

5. Ansaldi, S., L. Boato, M. Del Canto, F. Fusconi, and F. Giannini, 1989, Integration of AI techniques and CAD solid modelling for process planning applications, *Proceedings of Computer Applications in Production and Engineering (CAPE '89)*, pp. 351-364, F. Kimura and A. Roisstadas (Eds.), North-Holland Publishing Co., Amsterdam.

6. Descotte, Y. and J.-C. Latombe, 1981, GARI: A problem solver that plans how to machine mechanical parts, *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-81*, pp. 766-772.

7. Fridshal, R, 1984, Automatic generation of NC instructions from geometric models, *Proceedings of the Symposium on Computer Integrated Manufacturing*, ASME Winter Annual Meeting.

8. Iwata, K. and N. Sigimura, 1984, A knowledge based computer aided process planning system for machining parts, *Proceedings of the Sixteenth CIRP International Seminar on Manufacturing*, pp. 83-92.

9. Nau, D.A. and M. Gray, 1986, SIPS: An application of hierarchical knowledge clustering to process planning, *Integrated and Intelligent Manufacturing, Proceedings of the Winter Annual Meeting of ASME*, PED-Vol. 21.

10. Nau, D.S. and R.R. Karithi, 1989, Using a feature algebra for reasoning about geometric feature interactions, Eleventh International Joint Conference on Artificial Intelligence (IJCAI) August.

11. Nau, D.S. and R.R. Karithi, 1990, Handling feature interations in concurrent design and manufactuirng, Manufacturing International '90, March, Atlanta GA.

12. Mantyla, M. and J. Opas, 1988, HutCAPP--A machining operations planner. *Proceedings of the International Symposium on Robotics and Manufacturing Systems (ISRAM)*, pp. 901-910, ASME, New York NY.

13. Mantyla, M., J. Opas, and J. Puhakka, 1987, A prototype system for generative process planning of prismatic parts, *Modern Production Management Systems, Proceedings of AOMS'87*, A. Kusiak, (Ed.), pp. 599-611, North-Holland Publishing Co., Amsterdam.

14. van Houten, F.J.A.M., A.H. van't Erve, and H.J.J. Kola, 1989, Part: A feature based CAPP System, 21st CIRP International Seminar on Manufactuirng Systems, Stockholm, Sweden.

15. Kamarthi, S.V., 1990, An investigation into the application of neural networks for component design retrieval, Unpublished MS Thesis, Department of Industrial and Management Systems Engineering, The Pennsylvania State University.

16. Kamarthi, S.V., S.T. Kumara, F.S. Yu and I. Ham, 1990, Neural networks and their applications in component design data retrieval, *Journal of Intelligence Manufacturing*, Vol. 1, pp. 125-140.

17. Kumara, S.R.T. and I. Ham, 1990, Use of associative memory and self-organization in conceptual design, *Annuals of CIRP*, Vol. 40, pp. 117-120, August .

18. Pao, Y.H. and G.P. Hartoch, 1982, Fast memory access by similarity measure, *Machine Intelligence 10*, J. Hayes, D. Michie and Yoh-Han Pao (Eds.), Edinburgh University Press.

19. Minsky, M., 1980, K-Lines: A theory of memory, *Cognitive Science*, Vol. 4, pp. 117-133.

20. Schank, R.C., 1982, Dynamic Memory, *A Theory of Reminding and Learning in Computers and People.*, Cambridge University Press, New York NY.

21. Kolodner, J., 1980, *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Ph.D. Dissertation, Yale University, New Haven CN.

22. Hammond, K.J., 1989, *Case-Based Planning*, Academic Press, Inc., San Diego CA.

23. Touretzky, D.S. and G.E. Hinton, 1986, A distributed connectionist production system. Carnegie-Mellon University report, CMU-CS-86-172, Pittsburgh PA.

24. Tversky, A. and J. Hutchinson, 1986, Nearest neighbor analysis of psychological spaces. *Psychological Review*, Vo. 93, No. 1, pp. 3-22.

25. Pao, Y.H., 1989, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading MA.

26. Pao, Y.H., 1989, Functional-link nets: Removing hidden layers, *AI Expert*, Vol. 4, pp. 60-68, Miller Freeman Publications, San Francisco CA.

27. Yoh-Han Pao and Yoshiyasu Takefuji, 1991, Functional-link net computing: Theory, system architecture, and functionalities, Special issue of *Computer* on Computer Architectures for Intelligent Machines, IEEE Computer Society Press, Vol. 3, pp. 76-79, also published in Readings in Computer Architectures for Intelligent Systems (expanded version), J. Herath (ed.), in press.

28. Yoh-Han Pao and Wassim Hafez, 1992, Analog computational models of concept formation, *International Journal of Analog Integrated and Signal Processing*, Special Neural-Net Issue on Analog VLSI Neural Networks, Kluwer Academic Publishers, Boston MA. (in press).

5. The Inspection Planning and Evaluation Module

The Inspection Planning and Evaluation Module (IPEM) takes a part model—consisting of a specification of the part geometry as well as dimensioning and tolerancing information—and produces an inspection plan, which gives detailed instructions as to how to inspect a manufactured part to determine whether it is within tolerance. An inspection plan might consist of printed instructions to be manually performed by an inspector, or it might consist of code that can be executed by automated inspection equipment such as a computer controlled coordinate measuring machine (CMM). Ideally, the inspection plan will consist of a combination of both manual and automatic steps, so that each aspect of a part is inspected in the optimal way in order to achieve the minimum inspection time plan. Inspection planning for CMMs is emphasized in this report because of the Defense Manufacturing Modification Facility (DMMF) use of such a machine.

The RDS is a feature-based CAD system in which "features" are used to describe the geometry and tolerancing of a part. More details of this representation can be found in [1]. In this section, the representation of information as features is extended to inspection process planning where a set of tolerance geometries, both physical and conceptual, is created and compared in order to determine the usability of a manufactured part. The inspection features represent methods and detailed procedures for evaluating tolerances in conformity with industrial practices. Tolerances are specified according to the Geometric Dimensioning and Tolerancing standard [2] and can be either coordinate tolerances or geometric tolerances. Special emphasis has been placed on geometric tolerances in this report.

Inspection of a given part can be distilled into measurement of tolerance geometries. The physical measurement that creates a geometry is represented by a specific task such as moving the CMM probe to a surface and touching a select sequence of points or manually fitting a radius gauge to the curvature of a fillet. The task instructions are contained in data objects called "inspection plan fragments." A single tolerance geometry can often be measured in multiple ways resulting in the generation of many inspection plan fragments.

The overall inspection process planning consists of generating all possible inspection plan fragments for each tolerance in the design and combining them into an overall time-efficient inspection plan. An algorithm for inspection process planning is described and applied to a sample part. Special considerations which are important to inspection planning for CMMs (such as the generation of collision free inspection probe paths) are briefly described. Once an inspection process plan is generated it is translated into executable code or instructions for a computer controlled CMM.

Motivation

It can be quite time-consuming for an inspector to understand a drawing and determine how to inspect a part. Programming CMMs is also time-consuming, tedious, and highly susceptible to human error. This overhead is particularly critical when it is spread over a small number of parts. An automated inspection system can reduce the time between part design and final inspection, cutting costs and allowing better response.

By generating the inspection plan when the part is being designed, the system can aid the designer in producing part designs that do not require unnecessary inspection procedures and eliminate confusion between the designer and inspector over part inspection requirements.

Background

Geometric Dimensioning and Tolerancing (GD&T)

The inspection process is driven by tolerances specified by the designer. Tolerances are modeled on the ANSI Y14.5 standard [2] and can be of two types: coordinate tolerances (also known as plus/minus [±] or two point) and geometric tolerances.

A coordinate tolerance, such as the tolerance on the distance between the left and right surfaces in Figure 17, requires that two point measurements be taken and the distance between the points be evaluated. These are usually shown on an engineering drawing as a dimension with an associated ± tolerance. If this distance is within the specified variation, then the tolerance is satisfied. Coordinate tolerances are used to specify relationships between two surfaces or lines and can be measured in any pose (position and orientation).

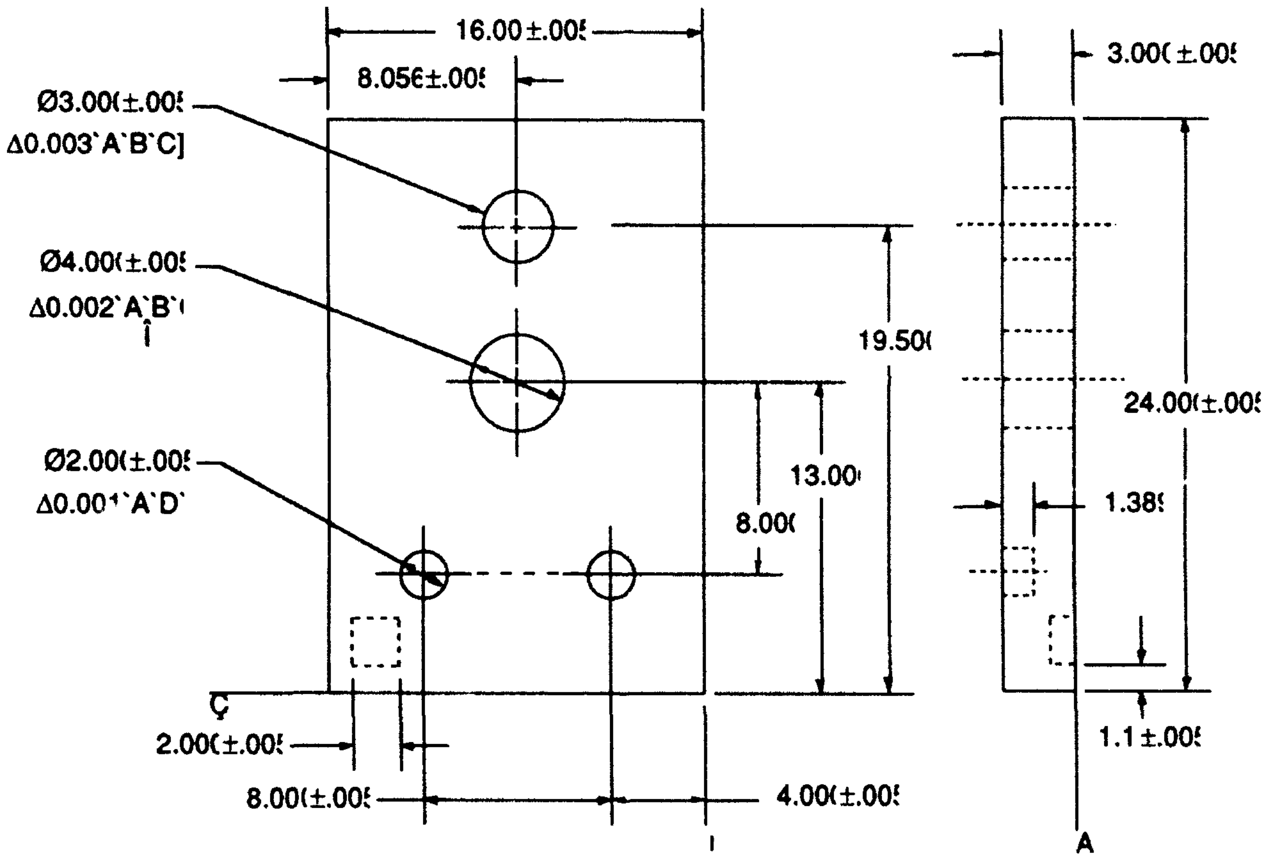


Figure 17. Sample part dimensioned and toleranced according to GD&T standard

Geometric tolerances, such as the position tolerances on the holes in Figure 17, are more complex and can be used to specify allowable variations in the pose of a geometric feature or can be used to specify allowable variation in the intrinsic properties of a feature, e.g., cylindricity of a hole. Geometric tolerances such as those of position and angularity must be defined in Datum Reference Frames (DRFs). These are coordinate systems whose origin and orientation are uniquely specified by three datums. Datums can be either surfaces or lines (e.g., planes or the axes of holes). Points can also be used as datums but will not be considered here.

A DRF specifies the part coordinate system to be used for both measurement and evaluation of the part. For example, the datums labeled A, B, and C in Figure 17 define the datum reference frame indicated as $\boxed{A}\boxed{B}\boxed{C}$, which must be measured prior to verifying the hole position tolerances. Figure 17 contains two different datum reference frames, ABC and ADC. It also contains four positional tolerances with two referring to the large through holes and two referring to the smaller blind holes.

Coordinate Measuring Machines

A coordinate measuring machine is typically a gantry-type robot with three orthogonal degrees of freedom [See Figure 18]. The "arm" of the CMM is equipped with a touch probe to make point measurements of the surface to be inspected. In general, the motion of the CMM is characterized by two distinct commands: a "MOVE" command is used to move the probe at a high speed between measurements. It is important that this movement be collision-free. A "MEAS" command moves the probe very slowly in the specified direction until the probe tip contacts the part. This command is used to acquire measurement data.

Simple touch probes are fixed in orientation and may consist of nothing more than a precision rod with a spherical tip. More sophisticated probes are similar to a robot wrist with two rotational degrees of freedom (See Figure 26).

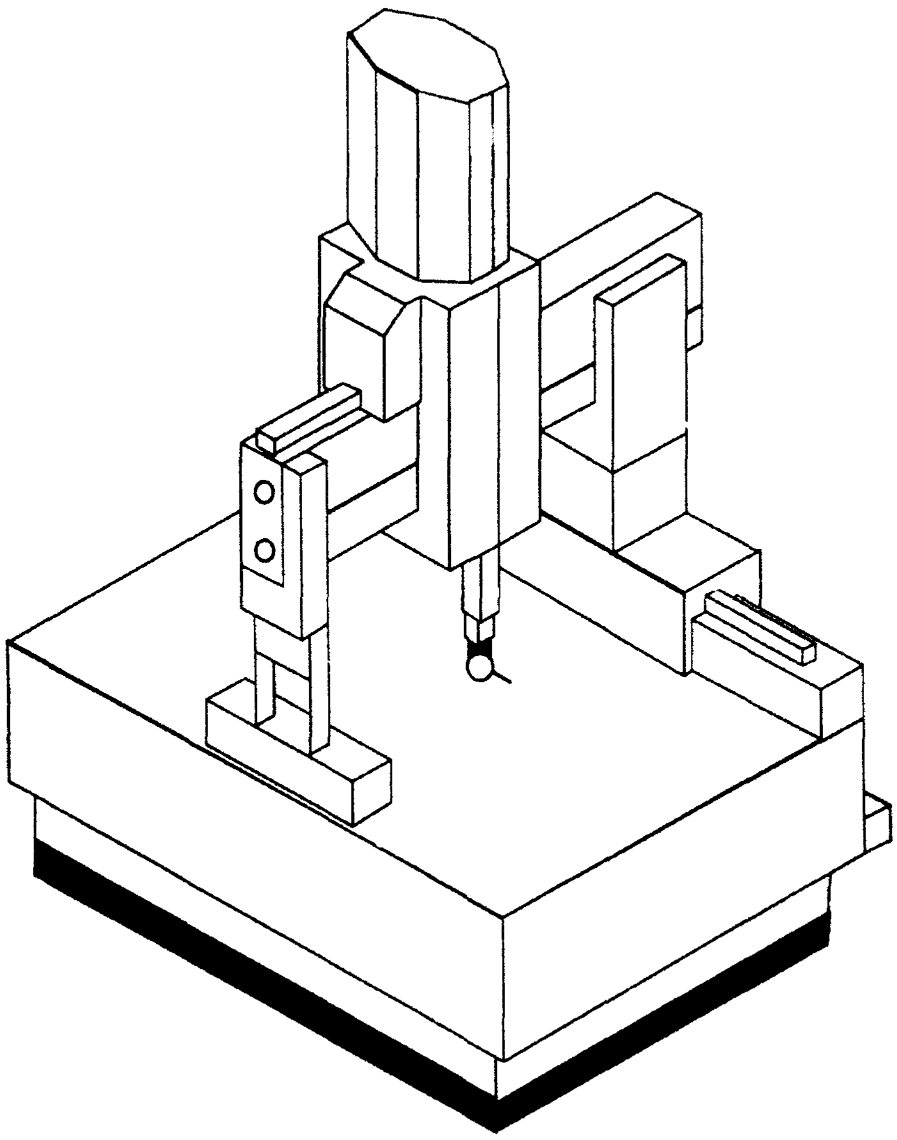


Figure 18. LK MicroVector coordinate measuring machine

<u>Inspection</u>

Inspection consists of performing manual and/or automatic operations, or tasks, to evaluate the specified tolerances. The complete sequence of such operations, when executed, will result in the statement that a part meets or fails to meet all specified tolerances. Additional information such as which tolerances were not met, machining variation for process control, and the like, may also be provided.

Datum measurements and evaluations must be done before any geometric tolerances relate to a datum reference frame (e.g , the position tolerance with the ADC datum reference frame in Figure 17). In general, when the primary datum (the first datum specified in a callout symbol, such as A in Figure 17) is a plane, the part will be placed with its primary datum in contact with the inspection table. The inspection table is measured to determine the primary datum. This corresponds to the concept known as "hard gauging" and is the basis of inspection techniques derived from ANSI Y14.5.

ANSI Y14.5 inspection techniques were developed for "hard" gauges using surface plates, fences, plugs and the like. Inspection procedures using CMMs can only sample the surface at a finite number of points and evaluate the part geometry based upon those measurements. This process has been called "soft gauging" and can result in two basic problems in CMM inspection: 1) where to sample the surfaces and 2) how to interpret the sample measurements. We have used accepted industrial practices for the inspection and evaluation of tolerances in designs produced with the Rapid Design System (RDS). Hybrid procedures requiring CMM measurements from manually placed fences or plugs (for example, see [3]) are not implemented in the present version of the IPEM.

Inspection at the DMMF Quality Control (QC)

The IPEM team members have had many meetings with experts at the DMMF QC. We have observed their practices and techniques, asked about rules of thumb that they use, interviewed them about their expertise, and queried about what they expect from the IPEM. Their critique on our view of CMM inspection and how we are automating it has been solicited and is a foundation for the way IPEM is presently implemented.

The motivation for the Inspection Planning and Evaluation Module of the RDS is obvious if one spends an afternoon in the QC room of the DMMF. The present inspection process begins by the inspectors receiving the part to be inspected and the accompanying blueprints. Hours are spent in understanding the geometry of the part from the blueprints and the relationship of the geometry to the tolerances. Many times the tolerances are incorrect and/or created without inspectability in mind. Once the inspection strategy is formulated, the inspector must enter the CMES code for inspection of a part into the PC controlling the CMM using a word processor. This is very tedious because the CMES instructions require an array of 3-D data points (in thousandths of an inch). These commands specify the movement of the probe. Any typographical error in the coordinate points of the probe path could cause it to crash into the part or another object (e.g., a clamp) and damage a very expensive CMM probe-head! For many tolerances, the inspector presently evaluates the probed inspection point measurements with a pocket calculator because the CMES evaluation process is cumbersome and difficult to remember. The inspector is forced to spend most of his time doing routine trigonometry and algebra (which a computer could do easily, quickly, and accurately) rather than using his expertise to solve the difficult problems which are less frequent, but demand the inspector's time and attention.

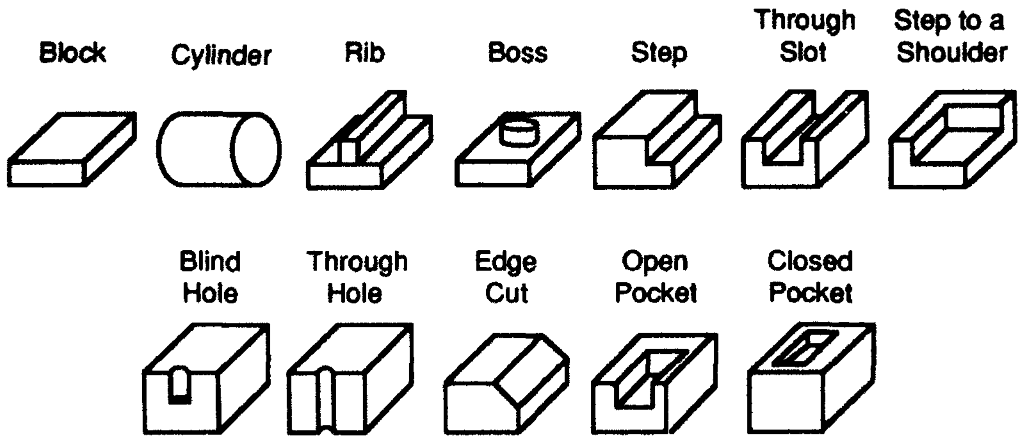


Figure 19. Form features supported by IPEM

Representation

The RDS is intended to support "feature-based design," in which parts are described in terms of "features" [1]. Features can be other than purely geometric, with each discipline such as inspection having its own set of features as described below.

Form Features (D1)

Form features define the form or shape of the part. Typical examples of form features include holes, pockets, ribs, bosses, slots, chamfers, and fillets. Many of these features can be further classified into subtypes. For example, holes can be blind or through, countersunk or not, with cone-shaped, flat, or spherical bottom (if blind). Form features can replace the geometric primitives (e.g., lines, curves, surface patches) used in traditional CAD systems to describe the shape of a part. Figure 19 shows the set of form features supported by the inspection planning module.

In general, a single form feature will replace several geometric elements. For example, a "blind hole with flat bottom" feature would replace two geometric elements: a cylindrical surface and a flat disk-shaped surface. This does not include any fillets. In traditional systems, fillets and chamfers are normally not represented explicitly as geometric elements. They are added to a drawing as textual notes. In the RDS they are distinct geometric features.

GD&T Features (D4)

There are various formalisms for dimensioning and tolerancing mechanical designs. Most designers have used the ± dimensioning convention exclusively. This convention is being supplanted by ANSI Standard Y14.5 which is becoming the notation of choice for dimensioning and tolerancing in engineering diagrams [2]. The notation specified by the ANSI Y14.5 Standard, usually called "Geometric Dimensioning and Tolerancing," or more simply, GD&T, is intended to specify, unambiguously, the allowed variation in shape of a manufactured part; this is not true of ± dimensioning [4]. More fundamentally, GD&T provides a richer framework than conventional ± dimensioning for the designer to indicate functionality by precisely specifying a relationship to be measured. The reader wishing more information about GD&T is referred to the standard [2] or to any of the available textbooks such as [5]. A simple example of the dimensioning and tolerancing of a block with four holes according to the GD&T standard is shown in Figure 17.

Commercial CMM software is readily applicable for performing commands on features such as those shown in Figure 19. For example, simple commands will interpret a set of measurements as a hole and will compute the position and diameter of the hole from the measurements. The CMM community has developed a standard software language called DMIS (Dimensional Measurement Interface Specification, [6]) which allows the user to define the nominal points at which an object is to be inspected (i.e., measured by a CMM probe) and then automatically interpret those measurements in terms of a user-defined tolerance zone. The LK CMM used by the DMMF uses a proprietary language called CMES [7] which is somewhat similar to DMIS. Because the DMMF is the end-user of the IPEM, the inspection planning output is in CMES. However, neither language interprets measurements in a strict GD&T sense, e.g., allocating bonus tolerances in interpreting measurements. They do not allow for the specification of interacting tolerances, which are commonly found in engineering designs (for example, MMC is the most commonly used modifier in mechanical designs using GD&T).

RDS inspection features based on GD&T and a subset of GD&T features are presently implemented in the RDS are shown graphically in Figure 18. GD&T features are subdivided into dimensions, datums, and tolerances. A dimension is an intrinsic property of a form feature or a relationship such as a distance between form features. A datum is an ideal reference line or plane which is used in the evaluation of a form feature measurement.

Datums can be located by appropriate datum positioning which makes physical contact with the high points of the reference surface (refer to the ANSI Standard for a precise definition) or approximated from CMM measurements. Diameter refers to the diameter of a hole feature; position refers to the location of any form feature; straightness defines a maximum deviation from the center of an axis (straightness can apply to either an edge of a block; rib or prism feature or to the surface of a cylinder); angularity is used to specify the angle between surfaces; perpendicularity is a specific instance of angularity where the angle is assumed to be 90°; and parallelism is another specific instance of angularity where the angle is assumed to be zero. In general, these implemented callouts either tolerance an intrinsic property of a form feature (such as straightness,flatness, or diameter), or a relationship between form features (such as a dimension). Relationships between surfaces belonging to an individual feature may be controlled by intrinsic attributes of that feature or by explicit GD&T callouts; relationships between surfaces belonging to different features must be specified using GD&T callouts.

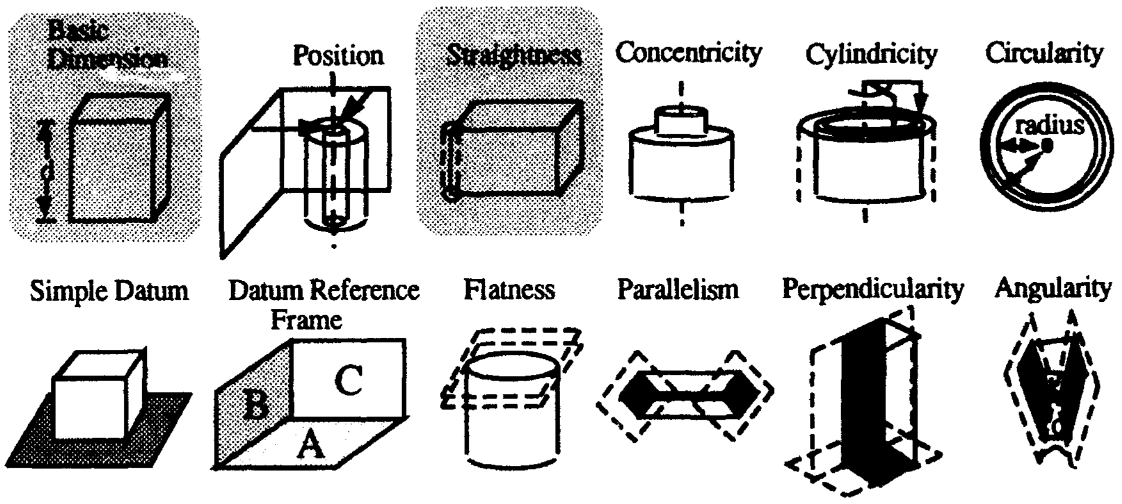


Figure 20. GD&T features supported by the IPEM (except for shaded objects)

Inspection Features (D3)

Inspection features are created as a result of the part designer's geometric and tolerance specifications. Where design (D1) features represent specific geometries, and GD&T (D4) features represent specific geometric tolerance zones, inspection features generate procedures for creating geometric constructs and evaluating measurements for a specified tolerance. Inspection features are similar to manufacturing features (D2) in that both are generated from a synthesis of the data in D1 and D4 features. Manufacturing features use geometric and tolerancing information to create a feature instance describing tool selection, cutting angles, speeds and feeds. IPEM constructs inspection features (which prescribe procedures to measure and evaluate tolerances) from the same geometric and tolerance information. An important distinction between the way manufacturing features and inspection features have been implemented is found in the object class definitions. Manufacturing features (such as "surface, cutter parallel" and "surface, cutter perpendicular") include in their definitions how the feature is to be manufactured. Inspection features use a more general class definition where specific inspection procedures are indicated by feature attributes rather than class definitions.

Inspection feature objects include:

- a definition of the toleranced geometry,
- a measurement attribute (how the inspection measurements are to be performed for this geometry),
- an evaluation attribute (how to evaluate the inspection measurements for the specified tolerance), and
- a comparison attribute (a boolean operation to determine if the evaluated measurements satisfy the tolerance).

The D3 feature geometry is determined by the assignment of the tolerance information to the part geometry. This assignment implies the creation of basic geometric elements such as a **point, axis,** or **surface**, the result is a set of basic inspection features. For example,

- tolerancing the offset dimensions used to position a hole creates a **point** where the hole center belongs,
- tolerancing the perpendicularity of a hole to a surface requires the creation of an **axis** to determine the angular hole pose with respect to that surface and
- tolerancing the position of a slot requires the creation of a **surface** to determine the slot position.

A complete list of basic geometries that represent inspection features upon which the IPEM currently operates is shown in Figure 21. The "virtual" features shown are geometries that are not an explicit component of the part geometry but are necessary to evaluate GD&T tolerances. For example, a designer using the RDS will explicitly specify a hole (as represented by its bounding surfaces) but the axis of that hole is not explicit in the RDS part model. However, for purposes of evaluating GD&T features the axis must become a separate geometric element upon which boolean geometric operations such as intersect or union can be performed.

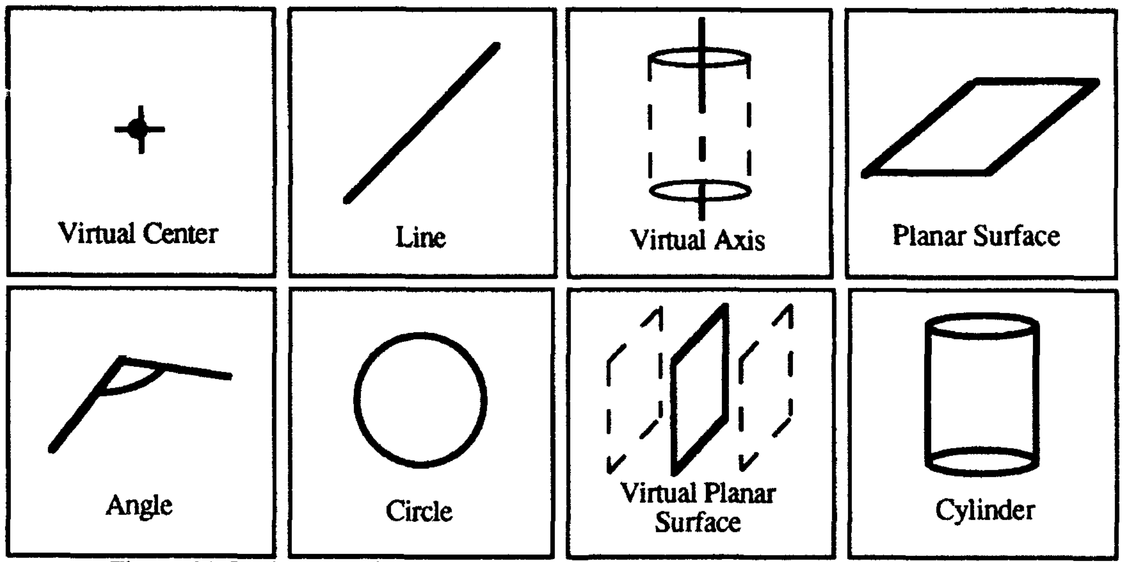
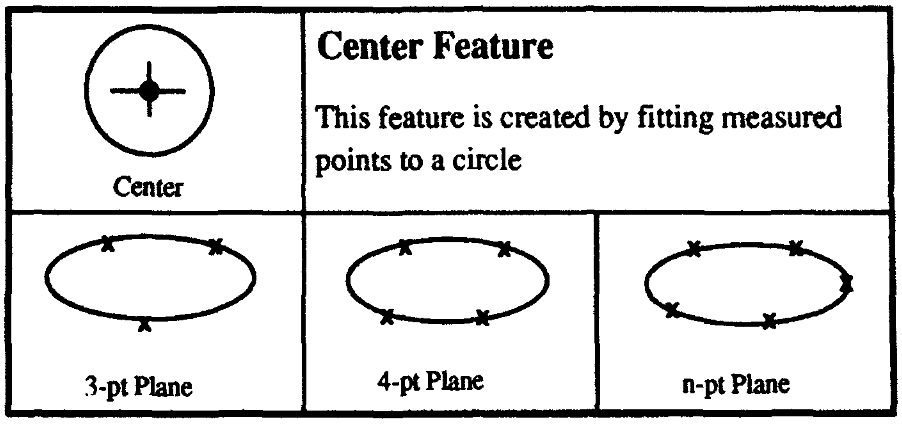


Figure 21. Basic inspection features that correspond to GD&T tolerance requests.

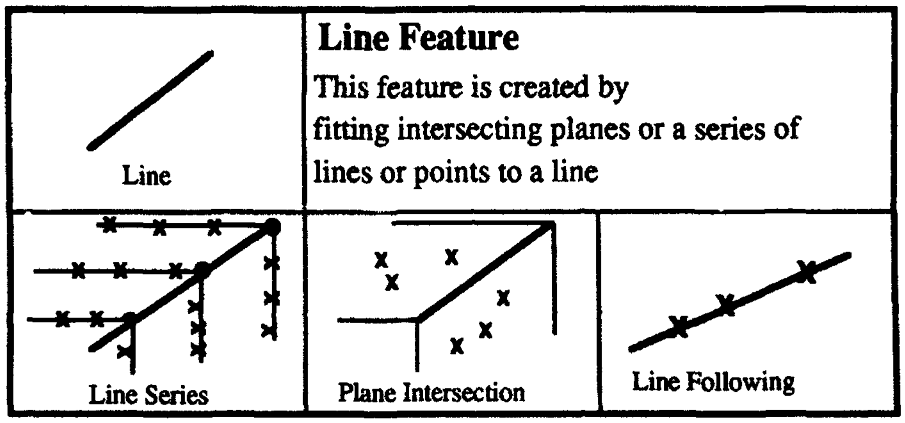
Although the inspection features can be represented by the simple range of geometries shown in Figure 21, each expand into a variety of measurement and evaluation methods. The measurement technique is a function of the precision of the specified tolerance. The present IPEM implementation has concentrated upon the tolerances (±0.001 inch or larger) measurable with the LK Micro Vector 80 CMM used by the DMMF QC Department. The measurement task primarily consists of moving the CMM probe near the surface to be inspected and performing a sequence of probe touches (surface measurements) at calculated surface points. Local constraints on the measurement procedures include tolerance type, limited access of the probe to the surface, or varying degrees of surface continuity, i.e., the surface to be inspected may consist of several spatially disconnected surfaces.

Tolerance type constraints determine the number and configuration of points based on the D1-D4 feature combination, samples of which are shown in Figure 22. Accessibility of the CMM probe to the inspection surface is being treated as part of the inspection setup generation (to be described later in this section). Spatially disconnected surfaces will be treated as part of the intersecting feature problem (also to be described later in this section). Proper implementation of these constraints requires an extension of the data structures used in the IPEM. Specifically, each geometric surface needs to maintain a bi-directional pointer to a nominal toleranced surface.

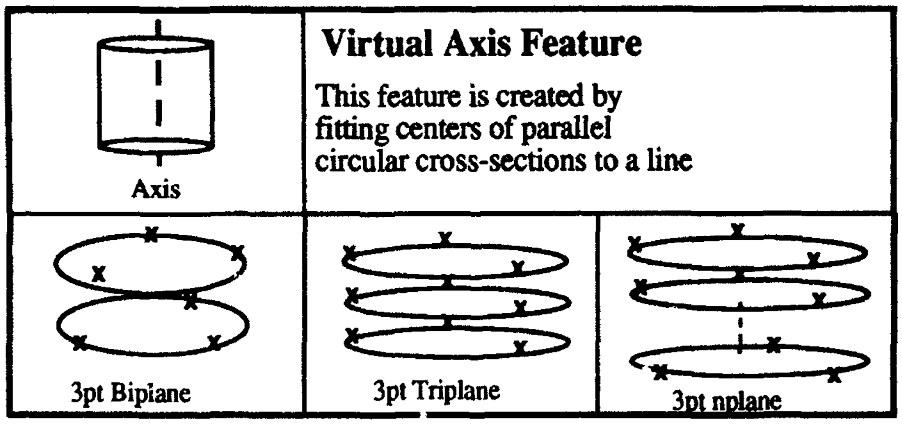
At present, only the tolerance type constraint is implemented. For the most part, the measurement techniques retrieve the minimum number of points needed to establish the inspection geometry feature. However, some tolerances such as flatness have no specific minimum number of inspection points. Some of the constraints specific to the inspection features of Figure 21 are three noncollinear points to measure or evaluate a plane, four noncollinear points for a plane or planar circle, or four noncollinear points in two or more planes to measure and evaluate a cylindrical surface. Figure 22 illustrates methods of creating the basic inspection features that arise from the geometric tolerances in Figure 21. Possible numbers of points and configurations needed to construct them are indicated.


Center Feature

This feature is created by fitting measured points to a circle

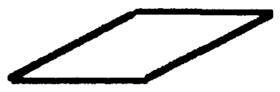
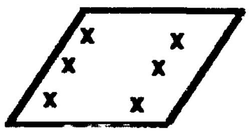
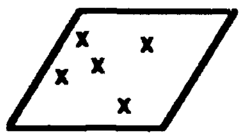

Line Feature

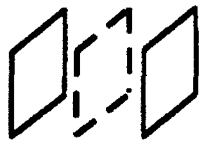
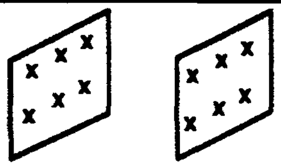
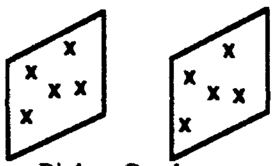
This feature is created by fitting intersecting planes or a series of lines or points to a line


Virtual Axis Feature

This feature is created by fitting centers of parallel circular cross-sections to a line

Figure 22. Detailed inspection feature geometries

 Surface	**Planar Surface** This feature is created by fitting 4 or more points to a plane	
 Grid (>3)	 Random (>3)	

 Virtual Plane	**Virtual Planar Surface** This feature is created by fitting a mid-plane to 2 sets of planar points	
 Biplane Grid	 Biplane Random	

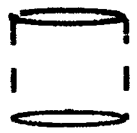
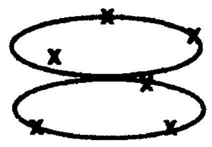
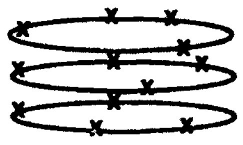
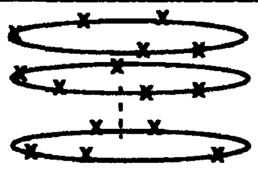
 Virtual Cylinder	**Cylinder Feature** This feature is created by fitting perimeters of parallel circular cross-sections to a cylinder	
 3pt Biplane	 4pt Triplane	 npt nplane

Figure 22. Detailed inspection feature geometries (con't)

Measurement variations that are not shown include constructs to evaluate the exact fit of a geometry to a tolerance zone. A good example is measuring and evaluating the flatness of a surface. A large number of points (10 or more) is needed to create a more accurate planar fit.

Figure 22 presents a detailed description of inspection features; however, these features are the result of a particular combination of design and GD&T features. The inspection feature that is most appropriate to a specific toleranced geometry, i.e., a particular geometry and GD&T feature combination, is presented below. Table 2 indicates the specific geometry that will be generated by the IPEM to evaluate the GD&T feature (listed down the side) applied to the appropriate design feature (listed across the top).

Inspection Plan Features

The dimensions and corresponding inspection features described in Figures 5 and 6 represent specific instances of toleranced geometries. However, in order to produce an inspection plan which is an ordered (in a time optimal sense) sequence of inspection operations with no redundancies, i.e., unnecessary operations, much more information than the simple geometries of Table 2 is needed. The class of features, called Inspection Plan Fragments (IPFs), implemented in the IPEM includes the specification of the geometry shown in Table 2 as well as additional information:

- how the part surface is to be measured,
- how the part is to be oriented to permit inspection of that geometry,
- how the measurements are to be evaluated,
- how the evaluated geometry is to be compared with the tolerance specification, and
- how the inspection of that particular toleranced geometry is to be placed into an overall part inspection plan

Table 2. Inspection feature geometry as a function of design and GD&T features

D1 → *D4↓*	*Hole*	*Boss*	*Pocket*	*Through Slot*	*Open Step*	*Step-to-Shoulder*	*Edge Cut*	*Rib*	*Single Surface*
Flatness	NA	NA	NA	NA	NA	NA	NA	NA	P Surf
Straightness	NA	NA	NA	NA	NA	NA	NA	NA	NA
Circularity	Circle	Circle	NA	NA	NA	NA	NA	NA	NA
Cylindricity	Cylinder	Cylinder	NA	NA	NA	NA	NA	NA	Cyl Surf
Perpendicularity	V Axis	V Axis	VP Surf P Surf	VP Surf P Surf	VP Surf P Surf	VP Surf P Surf	P Surf	VP Surf P Surf	P Surf
Angularity	V Axis	V Axis	VP Surf P Surf	VP Surf P Surf	VP Surf P Surf	VP Surf P Surf	P Surf	VP Surf P Surf	P Surf
Parallelism	V Axis	V Axis	VP Surf P Surf	VP Surf P Surf	VP Surf P Surf	VP Surf P Surf	P Surf	VP Surf P Surf	P Surf
Position	V Center V Axis	V Center V Axis	V Axis VP Surf	VP Surf	NA	NA	NA	Axis VP Surf	V Axis P Surf
Concentricity	V Axis	V Axis	NA	NA	NA	NA	NA	NA	V Axis
Primary Datum	V Axis	V Axis	NA	NA	NA	NA	NA	NA	P Surf V Axis
Secondary Datum	V Axis	V Axis	NA	NA	NA	NA	NA	NA	P Surf V Axis
Tertiary Datum	V Center	V Center	NA	NA	NA	NA	NA	NA	P Surf V Axis

Legend	
NA	Not Applicable
V	Virtual
P	Planar
VP	Virtual Planar
Cyl	Cylindrical
Surf	Surface

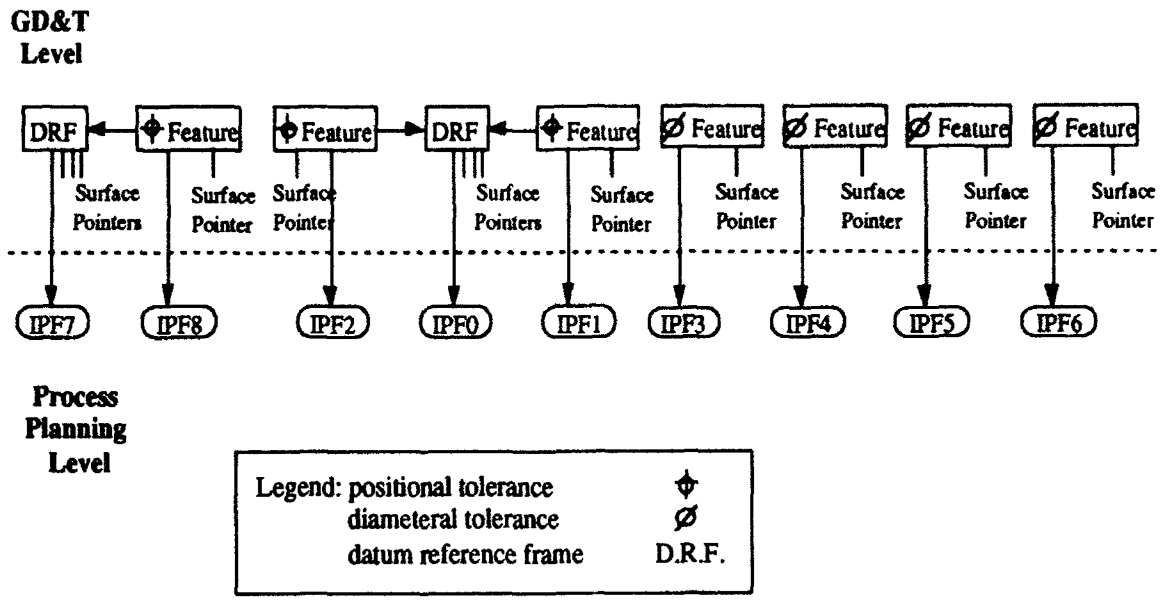


Figure 23. Relationship between GD&T and inspection plan fragments (for example of Figure 17)

Inspection Plan Fragments are generated on a one-to-one basis from the GD&T features. Specifically, each Inspection Plan Fragment is created by linking each instance of a GD&T feature class to an Inspection Plan Fragment [8]. Figure 23 shows the relationships between the IPFs and the GD&T features for the part in Figure 17. The positional tolerances on the large through holes are linked to the GD&T datum reference frame ABC through their specification. This relationship is shown as horizontal links in Figure 23. Since the modules of the RDS are relatively independent interacting only through the part model or the Episodal Associative Memory, this structure allows the IPEM to create feature instances which can be manipulated and processed into an inspection process plan without modifying or otherwise altering any design (D1), tolerance (D4) or manufacturing (D2) features. This isolation is shown by the dashed horizontal line in Figure 23.

Each IPF corresponds to a possible valid inspection procedure for the associated tolerance, and specifies:

- the type of the tolerance,
- a pointer to the GD&T feature which generated this IPF,
- the geometry (as per Table 2),

- a list of the coordinates of the points to be sampled and the orientation of the CMM probe,
- how the measurements are to be evaluated, and
- how the evaluation is to be tested for the specified tolerance.

The IPFs generated from the GD&T features are valid when they do not interact. For the purposes of this report, interacting form features are defined to be features which intersect geometrically in such a way that not all feature surfaces are present in their entirety. Work is in progress to detect interacting form features and modify the IPFs as appropriate. This will be described later in this section; however, this work has not been finished and the following discussion will relate only to form features which do not interact.

For each tolerance feature, separate IPFs can be generated for different CMM probes and probe orientations, and manual inspection tools (such as depth micrometers and plug gauges). Manual techniques such as inserting plugs into holes for CMM measurements are also excluded in this IPF expansion. Even though many IPFs can be generated for each toleranced feature, only one IPF needs to be executed for each tolerance. How redundant IPFs are eliminated from the inspection process plan will be described under the pr ɔcess planning algorithm. Manual procedures are not presently implemented.

An IPF logically contains the following objects, each of which corresponds to one of the inspection features listed above:

- meaurement request: a link to the surface of a feaure to be measured, the number of poings to be measure, and any constraints on those point, e.g., that they be noncollinear. Specifically this describes the geometries of Table 2.
- evaluation request: a high-level representation of an evaluation to be performed.
- comparison request: a high-level representation of a comparison to be performed.

The planner organizes the "request" objects listed above into a process plan. The process planner is responsible for eliminating redundant requests to the same surface(s), modifying the IPFs in the case of interacting features, and organizing the requests in a time-efficient manner. This will be described after a brief description of the IPEM feature class structure.

Evaluation Requests

These are data objects which contain information on how measurement data should be evaluated. Based on specification from the designer, the touch probe sample point data can produce geometries based on the following possible methods:

- Least Square Fit [9]: the most commonly used method; fitting a geometric object to sampled points.
- Fitting points to mathematical equations for creation of various geometries: Line, Circle, Plane, and Sphere.
- Maximum Inscribed or Minimum Circumscribe. ror circular surfaces only. Maximum Circumscribed: fitting the largest circle to set of points from without (outside); Minimum Inscribed: fitting the smallest circle to a set of points from within (inside).

These methods range from general to specific. The present IPEM's evaluation method is based on Least Square Fit, since our CMM language (CMES) makes use of it exclusively.

The output of the evaluation process will be a geometric object corresponding to the entries of D1-D4 combinations in Table 2. Eventually, IPEM will offer the designer or inspector a choice of the evaluation method. However, this requires an interface with the CMM to supplement and/or bypass the default evaluation procedures of the active programming language.

Comparison Requests

Comparison requests are also data objects but of a different kind. These compare the evaluated geometries with the tolerance constraint geometries (referred to as D4 features, see Figure 20). The constraint geometries are those which create the allowable tolerance zones within which the evaluated D3 geometries should fit.

The comparison task is proposed to be a Boolean operation. That is, either the part passes inspection or it does not. This process could also be expanded to determine the degree of failure, informing the inspector how it could be made to pass the inspection. For instance, a hole with a smaller diameter than specified can be bored again to meet the specification. This assumes that the part is retrievable. On the other hand, a hole bored too large cannot be salvaged.

IPEM Class Structure

Most, if not all, of the class definitions of IPEM are based on CLOS (Common Lisp Object-Oriented System) rather than the class definition structure of the Wisdom Systems Concept Modeler (CM), a solid modeling commercial package used as a programming base. The reason behind this action is that objects created by the Concept Modeler are modifiable by the user, where CLOS objects are not. Users are not allowed access to the internal objects of the system; hence, the CLOS class structure was chosen.

The class names, attributes and descriptions of all object classes used in the IPEM are listed below in the following format:

Class name Description
Attribute name

Insp-Planner	Global inspection plan
IPFS	List of all inspection plan requests of the part
Stable-Resting-Surfaces	List of stable exterior surfaces
Measurement-Rqst-List	Unordered list of all surfaces needing inspection
Setup-List	Unordered list of all setups (orientations) for the part
Sequence-List	Unique and (semi-) optimum sequence of inspection tasks
Global-CMM-Code	Instance of CMM-inspection-plan, the generic CMM code (i.e., metacode)
Standoff	Default offset value used for CMM path calculations
Intersecting-Features?	Indication for consideration or nonconsideration of intersecting form features for the process plan (i.e., a flag)

IPF	Inspection plan fragment (D3 feature)
Name	Name of the tolerance
Type	Type of the tolerance, i.e., position, angularity, circularity
Tolerance-Link	A pointer to the tolerance instance which generated this IPF
MR-List	List of measurement requests needed to inspect/evaluate the tolerance
ER	Evaluation request
CR	Comparison request
Evaluation-Constraints	Constraint placed on evaluation process
Independent-CMM-Code	CMM-metacode for a D1/D4 combination

Measurement-Request	Surface measurement information for an IPF
Geom-to-Inspect	Geometry of the surface to inspect
Surface-Type	Type of the surface, e.g., planar, or cylindrical
Number-of-Pts	Total number of inspection points needed for this surface
Insp-Pts-List	List of xyz-coordinates of the inspection points
Constraint-on-Pts-list	List of constraints placed on inspection points e.g., random, co-planar
M-Vector	Move vector for any of the sample points
P-Vector	Direction of approach for CMM probe
IPF-Link	A pointer to corresponding IPF(s)

Setup	Orientation information: one per stable surface
Geom-of-Resting-Surface	Geometry of the stable surface
MR-List	List of measurement request inspectable in this orientation
Measurements-in-DRF-list	Information to create a virtual coordinate frame in which tolerances are associated with the datum reference frame that establishes the origin they reference
Stable-Surface-Orientation-Matrix	Transformation matrix needed to transform the part so that the resting surface of it is lies on the x-z plane
VCF-List	List of Virtual Coordinate Frames (i e., a set of three

	mutually perpendicular geometries, for example, a datum reference frames)
Measurements-in-DRF	Inspection points contained in a setup
DRF-pointer	Memory location pointer
DRF-name	External design name; i.e., ABC,ADE,etc.
DRF-type	Abbreviation of datum geometries; PPP, PAP,etc. (P= planar, A= axis)
Datum-Resting-Surface	Geometry of the part resting surface
DRF-I-Depend-On	DRF that must be measured and evaluated before this DRF can be established
DRFs-Depend-On-Me	DRFs that require this DRF to be measured and evaluated before they can be established
DRF-MR-List	Measurement requests to establish this datum reference co-ordinate frame
Individual-Type-MRs	Measurement requests for "individual" type tolerances
Related-Type-MRs	Measurement requests for "related" type tolerances
Final-MR-order	Measurement requests sequenced in a minimum distance probe path within a datum reference co-ordinate frame
Pts-and-Distance	Euclidean distance between two measurement requests
Request-Name	Name of this Measurement Request
E-Dist	Euclidean distance from the last point measured of the previous measurement request and first point of the present one
Anchor-pt	Point from which the Euclidean Distance is calculated
CMM-Inspection-Plan	Overall CMM code for the part; one per part
Inspector	Inspector's name
Part-name	Name of the part to be inspected
Part-description	Text description of the part to be inspected
Date	Date the CMES code was written
Metacode-list	Slot contains the list of all the `CMM-metacode` instances. It is this list that gets generated into CMM instructions.
Clear-list	List that contains the information to move the probe a safe distance from the part where there are no obstacles that might cause a collision.

	Offset	Standard offset value where the probe is positioned before it takes a point.
	CMM-language	Defines which generator to call for the CMM instruction output (CMES or DMIS)
CMM-Metacode		Information needed by CMM regarding D1/D4 combination
	Feature-type	Encrypted form-feature type (e.g., TH = through-hole).
	Tolerance-type	Encrypted tolerance-feature (e.g., POS = position).
	DRF	Name of the datum reference frame.
	Approach-vector	Direction (X,Y,Z) points should be taken in.
	POS-info	List of information about where the feature is located. It differs for each type of feature to be inspected.
	TOL-info	List of information about the tolerance values placed on the feature.
	Num-points	Number of points to be inspected.
	Feature-name	String with the feature name from the FBDE. Used for comments in the CMES file.
	Save-meas?	Whether or not the surface to be inspected has been inspected before and is therefore redundant (not presently used).

Inspection Planning Algorithm

The steps for generating a part inspection plan are graphically illustrated in Figure 24 and described in greater detail below:

Step 1 Check Appropriateness Of D4 Tolerance Assignments

Since it is theoretically possible to assign any GD&T tolerance (D4 features) to any form, feature (D1) there needs to be a mechanism for avoiding erroneous situations. For example, placing a cylindrical tolerance on a rib makes no sense. Therefore, the "Allowable D1-D4 Specification" table, which contains information regarding GD&T tolerances and form features, is accessed each time a D4 feature is to be created. The entries of this table are "Y," " NA," and "NS." "Y" indicates that D1-D4 combination

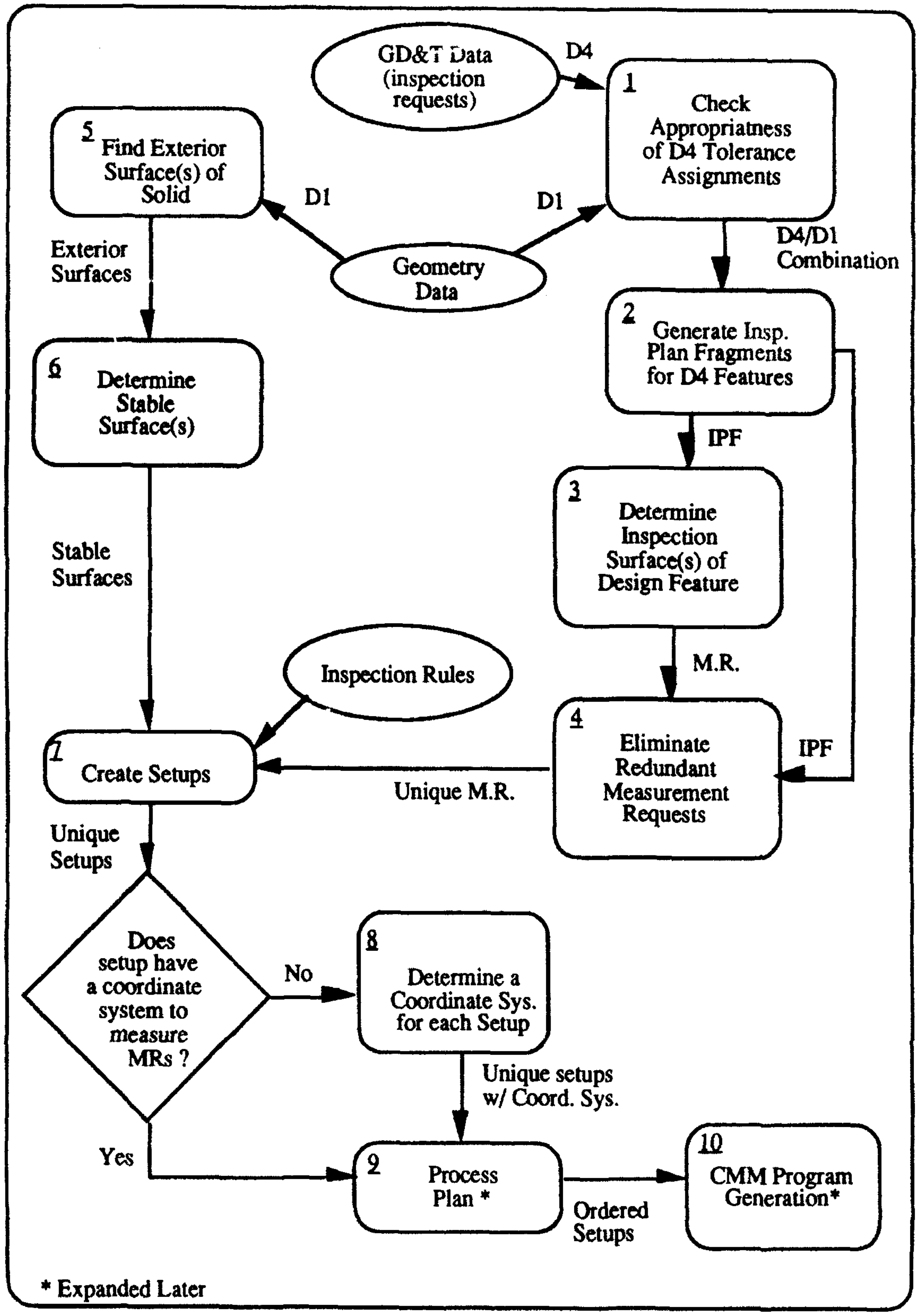

GD&T Data (inspection requests)
D4
1 Check Appropriatness of D4 Tolerance Assignments
5 Find Exterior Surface(s) of Solid
D1
D1
Geometry Data
Exterior Surfaces
D4/D1 Combination
2 Generate Insp. Plan Fragments for D4 Features
6 Determine Stable Surface(s)
IPF
3 Determine Inspection Surface(s) of Design Feature
Stable Surfaces
Inspection Rules
M.R.
4 Eliminate Redundant Measurement Requests
IPF
7 Create Setups
Unique M.R.
Unique Setups
Does setup have a coordinate system to measure MRs ?
No
8 Determine a Coordinate Sys. for each Setup
Unique setups w/ Coord. Sys.
Yes
9 Process Plan *
Ordered Setups
10 CMM Program Generation*
* Expanded Later

Figure 24. Inspection planning algorithm.

is appropriate. "NS" means that as far as the GD&T convention is concerned the D1-D4 combination is not standard but the system allows it anyway. "NA" stands for not allowable. The check for appropriateness is performed at the user interface level. A message is provided to the user if he/she selects a combination which results in an "NA" or "NS" entry. The present implementation of this is shown in Table 3.

Table 3. Allowable D1-D4 specifications

D1's → *D4↓*	*Blind / Through Hole*	*Boss*	*Pocket*	*Through Slot*	*Open Step*	*Step-to-Shoulder*	*Edge Cut*	*Rib*
Flatness	NA	NA	NA	NA	NA	NA	NA	NA
Straightness	NA	NA	NA	NA	NA	NA	NA	NA
Circularity	Y	Y	NA	NA	NA	NA	NA	NA
Cylindricity	Y	Y	NA	NA	NA	NA	NA	NA
Perpendicularity	Y	Y	NS	Y	Y	Y	Y	NS
Angularity	Y	Y	NS	Y	Y	Y	Y	NS
Parallelism	Y	Y	NS	Y	Y	Y	Y	NS
Position	Y	Y	Y	Y	NA	NA	NA	NS
Concentricity	Y	Y	NA	NA	NA	NA	NA	NA

Step 2 Generate Inspection Plan Fragment for D4 Features

Each D4 feature initiates an Inspection Plan Fragment which is a set of specifications about how a D4 instance is to be inspected/evaluated. An IPF of the inspection feature class and its attributes were described under **REPRESENTATION**.

Step 3 Determine Inspection Surface(s) of Design Feature

Before the measurement request attribute of an IPF can be instantiated, the surface(s) of the form feature that will actually be inspected (measured) must be determined. For instance, when a position tolerance is placed on a blind hole, only the cylindrical side surface is to be inspected. A position tolerance does not require any measurements of the bottom hole surface. Each D1-D4 combination specifies a certain set of D1 surface(s) to be inspected. This step specifies the correct set of surfaces by identifying the existing surface(s) of each design feature. A measurement request instance is then generated for each specified surface. These MRs are then linked to the IPF which was generated for the D4 feature instance.

Step 4 Eliminate Redundant Measurement Requests

In the process of generating measurement requests for all datums and tolerances, multiple requests may have been generated for inspection of the same surface. For example, a blind or through hole inspected for perpendicularily and cylindricity will initially produce two IPFs with identical measurement requests. Since the information needed to evaluate both tolerances can be obtained from a single pass over inspection points on the same surface, only one set of points is required. Hence, these two MRs can be merged into one MR, provided that enough points are sampled to satisfy the requirements of both tolerances.

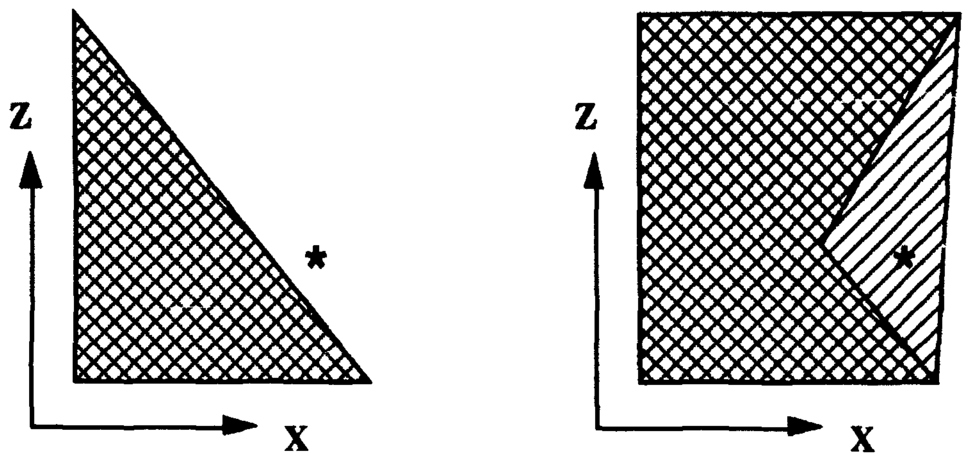
Step 5 Find Exterior Surface(s) of Solid

At some point in the inspection process the design model needs to be placed on the CMM table for inspection. For this to happen one needs to determine the flat exterior surfaces upon which the model can be placed. Some of these exterior surfaces may not be able to support the design model without the use of fixtures; therefore, it's necessary to determine which of these surfaces are stable. Hence, an exterior surface is either intrinsically stable or is considered unstable. The unstable surfaces may be designated as resting surfaces if fixturing is to be used. Currently no provision for fixturing is assumed.

Presently, exterior surfaces are identified by an offsetting method. A positive and negative offset surface is created parallel to each flat surface. These offset surfaces are then tested for intersections with the solid model. If both offsets intersect the solid model, the original surface (from which offsets were generated) will not become a candidate for stability test, otherwise the original surface will be tested.

Step 6 Determine Stable Surface(s)

The stability test involves the transformation of the candidate surface to the X-Z plane followed by creation of a two-dimensional convex hull area for the surface in question [10]. Projection of the center of the mass of the solid model onto the X-Z plane will determine the surface stability. Figure 25 illustrates this method (here, the part would be sitting on the page with the shown surface down).



(a) Unstable Surface (b) Stable Surface

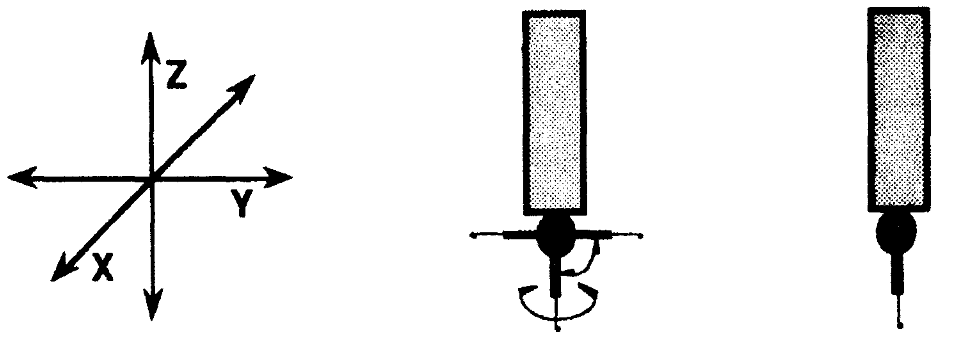
* Projected center of mass ▨ Convex Hull area ▧ Surface area

Figure 25. Surface stability using a 2-D convex hull algorithm

Step 7 Create Setups

Each stable surface establishes a possible part orientation for measuring the part in this step. Each surface to be inspected (i.e., one MR per surface) is assigned to a particular stable part orientation. This is done by orienting the part such that it is resting on an identified stable surface and then identifying which of the surfaces to be inspected can be accessed by the CMM in this part orientation. The set-up surface and list of accessible MRs are stored in the set-up class object. There exists one setup per stable surface. It's possible that a surface for inspection is inspectable in more than one setup. This redundancy is removed in later steps. Simple heuristics are used for determining which surfaces are inspectable in a particular orientation. For instance, only those inspection surfaces visible from a vantage point directly above the surface are accessible.

Currently the accessibility test applies to a probe holder with zero degrees of freedom (d-o-f) (Figure 26c). That is, the CMM model is capable of movement in three directions, as in a **xyz** coordinate frame (Figure 26a). Typically, a probe holder would add 1 or 2 additional degrees of freedom (Figure 26b). Our model does not add degrees-of-freedom beyond the basic **xyz** movement. The current probe model is merely an extension of the robot arm in the z-direction.



(a) simple xyz movement. **(b) 2-degree-of-freedom probe holder.** **(c) extension, but no additional d-o-f**

Figure 26. CMM movement configurations.

The resulting heuristic, as mentioned, tests the space directly above each inspection point for collision with the part. If an obstruction is detected, the surface on which the inspection point is attached is not inspectable in this part orientation.

It is possible that the space directly above an inspection point be free from obstruction although not accessible to the probe if the probe has to go through a small opening on its way to the inspection point. This is possible where two through slots are aligned across an open channel. To avoid an erroneous affirmation from the z-axis accessibility test, a geometric simulation of the CMM probe is created. For each point deemed accessible in the initial test, the probe simulation is positioned at that point as it would be if the actual CMM was taking the measurement. If an intersection between the part-model solid and the probe geometry is determined, the point accessibility is suspect. However, the case where the surface is accessible but the point placement is causing an intersection, the point is moved to a convenient location, accessible to the CMM probe. Again, an intersection test is performed. If the part solid and probe still collide, the surface is deemed inaccessible. Otherwise, it is accepted.

This test will be expanded, in later versions of the IPEM, to include probe holders which add additional degrees-of-freedom (as shown in Figure 26b). Additional considerations to guarantee a collision-free path between inspection points and surfaces are also planned. This matter involves allowing avoidance of clamps and part features.

Step 8 Establishing a Coordinate Frame **in each Setup**

A CMM requires a reference point from which to locate points on the physical part. The CAD representation of the part comes with a reference frame, but it is not possible to transfer the electronic part origin to the CMM. The CMM origin must be located in the CMM frame of reference. The only way to establish the CMM origin is to manually manipulate the touch probe to specific points on the physical part.

The object of the manual manipulation is to establish the origin of a three-dimensional coordinate frame. The software being used (CMES) establishes the origin by intersecting three ortho-normal vectors from the tactile data the operator provides. This can be accomplished in a number of ways by manipulating the touch probe to establish the simple geometric entities; plane, axis, or vertex. The required intersecting vectors are created using the 3-2-1 isostasies principle. This principle says that any object can be completely fixed in space by the simultaneous contact on three, two, and one point, respectively, on each of three mutually perpendicular geometries. Essentially, what CMES requires is to locate these points on the appropriate geometries of the machined part with it fixed to the inspection table. Typical geometric combinations are:

- three planes (abbreviated PPP, see Figure 27),
- a plane, an axis, and a plane (abbreviated PAP, another is PPA)
- a plane, and two axes (abbreviated PAA)

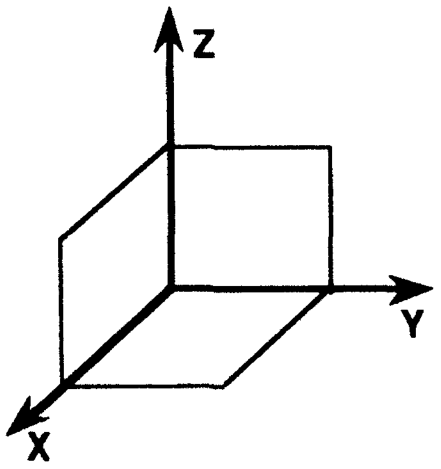


Figure 27. Coordinate system formed by intersecting three mutually perpendicular planes

In GD&T tolerancing, the geometries mentioned would be referred to as datums and, the coordinate frame created called a datum reference frame (DRF). It is this coordinate frame within which location tolerances, a member of the "related" tolerance category, are specified (e.g., position, concentricity). Orientation tolerances, also related tolerances, are not evaluated with respect to a DRF, but may need a single datum (e.g., perpendicularity, parallelism). Form tolerances, categorized more generally as "individual" tolerances, need no reference entity at all (e.g., flatness, straightness).

As long as the part is oriented on the CMM inspection table so that all three of the required datums are accessible, a datum reference frame can be established by manually manipulating the CMM probe. It is also possible to establish a coordinate system without using specified datums or DRFs. The inspector may create his own by simply choosing the proper combination of geometries. There would then be the additional step of transforming the coordinates from the intended DRF origin to the created one. The functionality to do this automatically is not yet implemented.

Step 9 Associating Measurement Requests with Datum Reference Frames

Presently, only DRFs are used to establish the coordinate frame in a setup. A valid setup is one in which all datums (of a DRF) can be accessed. The tolerances and other DRFs that are linked to the "setup" DRF (and are accessible), are grouped in the "measurement-in-DRF-list" property within the setup object. The measurement-in-DRF feature contains specific details about each DRF accessible in the orientation represented by the setup. The critical information being:

- The DRF datums must be located by a manual manipulation of the CMM probe. Otherwise, which DRF datum(s) is located in the course of evaluating a tolerance (such as position, perpendicularity, etc.).

- Which tolerances, accessible in a given part orientation, are related tolerances and which are individual.

This information is essential to the order in which part surfaces are probed. First of all, the DRF used to locate the part on the CMM table initially, must be probed manually. Therefore, the first action taken in the implementation of any inspection probe path plan must be to establish the physical location of the part using a DRF whose datums are of the individual type.

Secondly, if there are DRFs with datums that have related tolerances associated with them, the new coordinate frames cannot be established until those tolerances are evaluated. Hence, some general rules of inspection order become evident:

1) the manual probing to establish the part location and a datum reference frame
2) tolerances that callout the initial DRF
3) the establishment of a DRF with related datum tolerances from list in step 2
4) tolerances that callout the DRF from 3

This sequencing is recursive at two levels: through each setup, and through the number of DRFs occurring in 3. This primary sequencing is performed on each valid setup. The next step is to sequence the minimum number of setups, remove the redundant ones, and sequence the measurement request within each setup.

Step 9 Process Plan

A process plan consists of two sub-processes: 1) an ordering of setups, and 2) an ordering of individual measurement requests within a setup. This can be seen as a global ordering and a local ordering, respectively. In global ordering, the goal is to find the minimum number of setups for complete inspection of the manufactured part. This is accomplished by first identifying the setup with the largest number of measurement requests (MRs), say setup X. Then, the MRs specified in X are removed from all other setups. This may lead to a situation where several setups may lose some or all of their MRs. Setups with no remaining MRs are eliminated from the process plan and the remaining setups are ordered according to the size of their MR attribute.

Local ordering requires a logical grouping of MRs with the DRF that each MR references. If there are two (or more) DRFs in a setup, the measurement and evaluation of the second DRF's datums must precede the measurement of the tolerance which make use of the second DRF. Finally, within each grouping of MRs there is a spatial, "next closest" ordering.

MRs are grouped within the "measurement-in-DRF" object. The first MRs to be probed are those pertaining to the DRF datums. Establishing a DRF requires a surface probing order of primary, secondary, tertiary datums in a 3-2-1 point touch sequence. From the final tertiary point, the Euclidean Distance (ED) between this point and all sample points of every other MR in the current DRF coordinate system grouping is calculated. This grouping includes MRs from the related and individual type lists. The closest MR to the one just inspected is identified and placed next on the "final-MR-list." This is repeated until all MRs of each grouping have been processed. This procedure is performed on all DRFs in the present setup.

The spatial sequencing process could be optimized upon by using a "traveling salesman" type algorithm, where minimizing travel time between target surfaces is the highest priority [11]. It will be possible to incorporate local optimization as the IPEM's higher order planning functions are completely developed.

Step 10 CMM Program Generation

Once the setups and MRs are ordered, the actual commands for the CMM are generated. The goal is to produce an ASCII file according to the specifications of CMES (Co-ordinate Measurement Software[11]), the CMM operating language.

CMES is the proprietary language of LK Tool of Cincinnati Milacron Co., the makers of the CMM which was purchased by the DMMF QC for inspecting the manufactured products from the DMMF. This CMES text file will be transported (via modem, disks, or Ethernet) to the controlling PC of the CMM to be executed for inspection of the part.

First the MR feature objects are translated into the metacode feature objects. This translation to metacode is not a one-to-one ratio. The metacodes have different functions within the inspection plan: some measure and evaluate; some measure, save the points and evaluate; and some use saved points from other measurements to only evaluate. These differences arrive from the removal of redundant MRs, described above, where two tolerances on the same surfaces can use the same measurement points, just evaluated differently based on their tolerance. Once created, the metacode can be saved to disk for future retrieval.

Before the metacode is translated into the desired CMM code, the inspector can perform an inspection plan similar to the inspection plan on the workstation. The simulator overlays the CMM probe path on top of the part model and checks for any intersections (collisions). This allows the inspector to get a feel for the inspection plan before it is executed, showing where the probe is scheduled to travel, warning the inspector of possible fixtures or a cluttered table-top. The simulation also allows the checking of efficient programming and gives the inspector opportunity to fine tune the inspection plan, if desired.

For each DRF in a metacode object produced by the process planner, the origin of the coordinate system is determined and translated into the coordinates of the sample point to the origin of the setup datum reference frame. Prior to this translation the sample point coordinates are based on the coordinate system of the solid modeler, with its origin located at the center of the part model. Using the measurement requests of each tolerance (contained in the IPF), a series of CMM-metacode instructions is produced. This metacode is currently generated only for a subset of GD&T tolerance/design feature combinations, e.g., a position tolerance on a hole. The metacode is an intermediate description of the CMM commands and is independent of the actual language which operates the CMM. Even though the CMM purchased by the DMMF QC uses the proprietary language CMES, most other commercial CMMs have adopted the ANSI Standard DMIS (Dimensional Measurement Interface Specification [8]) language. The use of an intermediate metacode allows the IPEM output to be easily altered to produce DMIS instructions. Once the instructions are created for a part, they are saved in the part model data base for future retrieval.

Algorithm for Detection of Intersecting Design Features

One of the major problems of the existing IPEM is that it only works for nonintersecting features. Many real-world part designs do not follow this design constraint. The first step in handling the inspection of intersecting form features is, naturally, to determine which toleranced form features actually intersect with other form features. This process is useful for either masking out such features from the inspection process, if so desired, or in determining how to handle these "intersecting features." The algorithm for detecting intersecting toleranced features presented here is in its preliminary stages. At present, no attempt is made at detection of intersecting features by the inspection planning algorithm.

Overall Description of Algorithm

The intersecting feature detection process consists of checking the geometry (or surface) associated with a tolerance feature for intersection with other form feature geometries. Excluded from the check are the geometries for the starting block and the form feature to which a toleranced surface belongs. For example, the bottom surface of a through-slot should not be checked for intersection with the sides of the through slot or for intersection with itself.

The first step of the algorithm consists of constructing two lists; one contains all geometries associated with a form feature and the other contains all geometries associated with a tolerance feature. The second step involves checking each toleranced surface for intersections with the surfaces of other form features. For each pair of geometries that intersect, an output list is updated with information about the intersection. For overly complicated parts this procedure may become prohibitively expensive. In that case a more efficient algorithm will have to be devised.

Individual Steps of Algorithm

Step 1

Construct a list containing all geometries that belong to a form feature which represents an existing surface. For example, only one surface will be included for a through hole, since the top and bottom surfaces don't really exist. By the same token, three surfaces will be included for a through slot (i.e., the bottom and side surfaces). This list is referred to as `form-geom-list`. (In reality, all of the lists discussed here will be stored as part of a class instance.) This list is generated by performing the following substeps:

Step 1a

Obtain form feature instances of the part.

Step 1b

For each form feature instance, separate the associated three-dimensional geometry into its two-dimensional subgeometries. A method (or methods) is employed to determine which subgeometries to use based on the feature type.

Step 1c

For each subgeometry, create a pair consisting of the subgeometry and the name of the form feature to which it belongs.

Hence, the `form-geom-list` will consist of a series of geometry/feature-name pairs.

Step 2

Construct a list of all geometries that are toleranced. This list is called *`tol-geom-list`*. This is actually a list of pairs. Each pair consists of a geometry and the name of the form feature that a geometry belongs to. In order to avoid redundancy, this list is constructed based on the measurement requests generated by the IPEM. As each measurement request is generated, a geometry/feature-name pair is added to *`tol-geom-list`*. The geom number is simply obtained from the measurement request instance. The form feature name is obtained by tracing back through the data structure. The IPF-Link of a measurement request instance is used to obtain the associated IPF. Then the tolerance feature can be obtained using the Tolerance-Link of the IPF. Once the D4 instance is obtained, the form feature name is easily accessible.

Step 3

Check each geometry in `tol-geom-list` for intersection with all geometries in `form-geom-list` except for those that come from the same form feature. The form feature name in the `tol-geom-list` pair is checked against the form feature name in each `form-geom-list` pair. The pairs in `form-geom-list` that have the same feature name are ignored. The intersection check is done using a Concept Modeler function. For each pair of geometries that is found to intersect, the output list is updated. For each intersection found, the output list contains a four-element list made up of the geometry and form feature names for each intersection found. The algorithm is pictorially represented in

Figure 28.

Placement Within Inspection Planner Algorithm

The exact placement of the intersecting feature check within the Inspection Planner algorithm has not been determined and will be finalized as the code and algorithm discussed above are finalized. It is tentatively planned as a pre-processor to Step 3 (Determine Inspection Surfaces of Design Features) of the inspection planning algorithm.

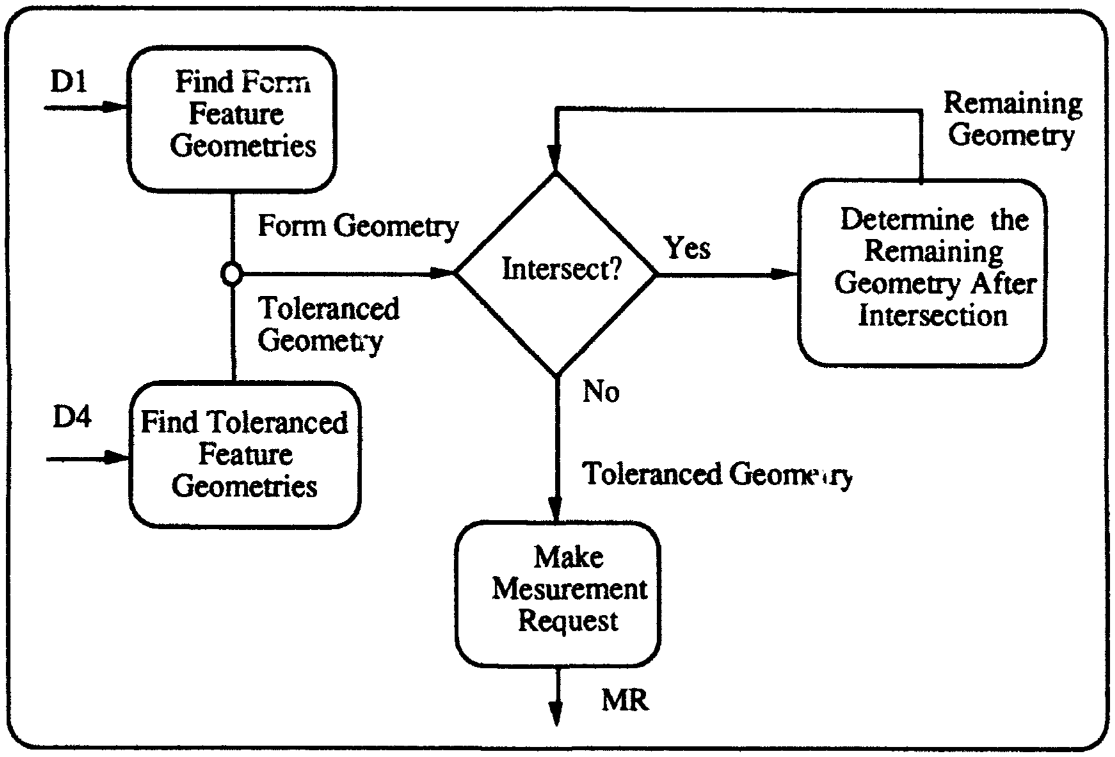

D1
Find Form Feature Geometries
Remaining Geometry
Form Geometry
Intersect?
Yes
Determine the Remaining Geometry After Intersection
Toleranced Geometry
No
D4
Find Toleranced Feature Geometries
Toleranced Geometry
Make Mesurement Request
MR

Figure 28. Intersecting detection algorithm

Discussion and Conclusions

The algorithms described in this section have been used to automatically generate inspection plans and CMES for simple RDS part designs such as the part shown in Figure 17.

The IPEM has demonstrated significant technical and manufacturing achievements. From the technical viewpoint the IPEM implements an inspection planning algorithm that treats toleranced features as three-dimensional objects (called geoms) existing in a three-dimensional workspace. All IPEM algorithms are based upon determining general three-dimensional attributes (such as orientation) of these geoms and manipulating (such as rotating the part to present a surface for inspection) in three dimensions. Because of this general approach to inspection process planning, the IPEM is fundamentally capable of being extended to complex three-dimensional tolerances such as profiles of sculpted surfaces.

A representation of inspection procedures as inspection plan fragments, i.e., inspection features, has been developed. This representation incorporates the geometry of the toleranced surface (a three-dimensional geom), a method of measuring the toleranced geometry, a method of evaluating the toleranced geometry, and a procedure for comparing the evaluated measurements with the toleranced nominal part dimensions. Because of the large number of possible combinations of these representations, procedures, and methods a generic inspection feature class has been defined, i.e., an inspection plan fragment, rather than defining a specific feature class for each combination. The toleranced geometries and the different inspection procedures and methods are defined as attributes of the more general inspection feature class.

These inspection plan fragments are organized by the part geometry and the tolerance specifications into part orientations called setups. A setup corresponds to a stable positioning of the part on a CMM inspection stage. Inspection plan fragments are assigned to setups and redundant setups are eliminated. The planning algorithm successively re-assigns inspection plan fragments to setups with the goal of maximizing the number of tolerances to be inspected in each setup. Since each setup represents a significant amount of human operator time, this approach minimizes the most time consuming aspect of the actual inspection procedure. Once the number of setups has been minimized, the inspection plan consisting of setups and associated inspection plan fragments is translated into CMES code which can be executed on the LK CMM located at the DMMF. Each setup generates instructions to the inspector as to how to position the part for inspection and the CMES code to automatically inspect the tolerances.

From a practical manufacturing viewpoint the IPEM is producing useful, working code that would be useful to the inspectors at the DMMF. At present they are not actually using the IPEM because it is still under development; however, feedback from DMMF personnel indicates a great deal of enthusiasm and interest for using it. Specifically, the IPEM reasons about the geometry and tolerances of the designed part-model (just as an inspector does) and creates detailed CMES code to properly inspect the part. Just by itself this is a significant contribution since this is the process that is the most tedious and error-prone part of CMM inspection — generating CMES code. The inspector then can interact with this code, modifying it as appropriate to his own goals, *resulting in significant* labor savings.

The major limitations of the present implementation are that the IPEM uses a small set of part geometries and tolerances and does not handle interacting features. Remedying the first limitation is merely a matter of implementation. Since all IPEM algorithms use three-dimensional tolerance and geometry descriptions, there is no reason not to develop the code that would handle these more complex features. The second limitation is more serious; most "real world" part designs incorporate significant numbers of intersecting features. Little work has been published about this particular problem. A methodology and an algorithm for handling intersecting features have been developed which appear to have potential. Rather than use "fragile" rules to describe feature interactions, we propose to use pattern recognition techniques and an episodal associative memory of similar situations which have been encountered and solved in previous part designs. The inspector would then be able to retrieve solutions for similar situations and modify them accordingly. This then be able to retrieve solutions for similar situations and modify them accordingly. This is a very powerful and robust technique, because the system can learn previous inspection solutions. Furthermore, the interaction with the inspector allows the inspector to control the process and apply his own experience to the problem. Although developed and manually tested on sample parts, the algorithm has not been implemented as of the date of this report.

The IPEM is presently capable of producing useful CMES code for the DMMF inspectors. It is not done in a rigid, hard coded manner for a specific part design but uses a general algorithm which incorporates geometric reasoning about the part geometry and tolerances to produce an inspection process plan and CMES code. Many of the rules and optimization criteria used in the IPEM for inspection process planning are heuristics determined from interviews with the DMMF inspectors. The inspection planning subsystem of the RDS is based upon input from the DMMF QC group and will be tested by them in the near future.

Practical goals include extending the design and tolerance features to more complex geometries, incorporating an extended set of evaluation functions (all LK CMM evaluation functions are based upon least squared error fits) that will include tolerance modifiers such as MMC, incorporating many more inspection procedures (including manual procedures), and simulating the CMM probe path on the computer screen.

References

1. LeClair, Steve, "Rapid Design System: Feature-Based Design", Proceedings of 1991 IEEE International Systems Conference, Dayton, Ohio, August 1991.
2. Dimensioning and Tolerancing, ANSI Y14.5M-1982, ASME, 345 East 47th Street, New York 10017.
3. "Dimensional Metrology and Geometric Conformance" in The Tool and Manufacturing Engineers Handbook, 4th Edition, Edited by Charles Wick and Raymond Veilleux, McGraw-Hill, 1987.
4. Jacobsohn, J., Radack, G. and Merat, F., "Integrating Knowledge of Geometric Dimensioning and Tolerancing into a Feature-Based Design System," *Proceedings of Rensselaer's Second International Conference on CIM*, Troy NY, May 1990, 151-159.

5. Foster, Lowell W., Geo-Metrics II (Revised 1986 Edition), Addison-Wesley,Tempe AZ, 1985.

6. DMIS 2.0 Specification, ASME 3.0, 345 East 47th Street, New York NY 10017.

7. CMES Programming Manual, LK Instruments.

8. F.L.Merat and Gerald M. Radack, "Automatic Inspection Planning within a Feature-Based CAD System," J. Robotics and Computer-Integrated Manufacturing, Vol. 9, No.1, 1992.

9. Galm, James, "Functional Surface Estimation of Machined Surfaces," Ph.D. Dissertation, Case Western Reserve University, January 1991.

10. Preparata, F.P. and S.J.Hong, "Convex Hulls of Finite Sets of Points in Two and Three Dimensions," *Communications of the ACM*, Vol. 20, No.2, February 1977, p.87-93.

11. Jeon, O-Keon, "Efficient Inspection Path Planning Algorithm," M.S. Thesis, Case Western Reserve University, Cleveland, Ohio, August 1990.