# Design knowledge representation, retrieval and delivery for co-operative knowledge processing

*C.-J. Su and M. Tseng*
*Department of Industrial Engineering and Engineering Management, Hong Kong University of Science and Technology*
*Clear Water Bay, Kowloon, Hong Kong*
*Telephone: (852)2358-7091 - 2358-7881; Fax: (852)2358-0062*
*E-mail: TSENG@usthk.ust.hk*

**Abstract**

The design, engineering, manufacture, and maintenance of high performance and high complexity systems have become one of the most important issues in the industrial world today. Facilitating these engineering activities depends on our ability to assemble and effectively deliver the life-cycle engineering knowledge. This paper deals with different problems of design using concurrent engineering, and with the application of information and knowledge management, with acquisition, representation, retrieval and delivery of design knowledge to the designers to reach co-operative knowledge processing. Car body shapes will be given as examples to shape based and feature based knowledge. Finally the support for interacting / integrated designer systems will be discussed.

**Keywords**

**Design knowledge, knowledge representation, co-operation, life-cycle, concurrent engineering**

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35357-9_22

A. B. Baskin et al. (eds.), *Cooperative Knowledge Processing for Engineering Design*

## 1 INTRODUCTION

The essential components of a system that can provide design knowledge assistance in co-operative knowledge processing include:

1) Application of previously acquired knowledge in the engineering area to the task at hand. The product designer is able to gain access to the knowledge of product design accumulated by his predecessors in an efficient manner. Similarly, the manufacturing or process engineer must have access to the knowledge of previous experts.
2) Application of background knowledge from outside the engineering area to the task at hand. The experiences and lessons learned in manufacturing and maintenance should be accessible to the product engineer so that products are designed with optimum manufacturability and maintainability.
3) Access to the decision rationale and the trace that led up to that most current state of the product or process definition. The product engineer must be able to quickly uncover the rationale of the conceptual design, and the manufacturing engineer must be able to lay open the decision rationale and assumptions of both the product designers and engineers.
4) Access to the most current state of the product or process design which includes information such as configuration description.
5) Man-machine interfaces to support in-site capture of product definition data. The system should be able to characterise and record the decisions relating to product definition data that are being made after the fact. Otherwise, the delivery of concurrent engineering support is virtually impossible. In addition, man-machine interfaces are crucial to the accumulation of the rationale and experience for establishing knowledge bases to support concurrent engineering

Management of the assimilation and flow of knowledge is complicated by both long project life spans and short project life spans. For example, at NASA, project managers used to be able to see from start to finish several major initiatives within a single assignment. However, the space station and other technologically-advanced proposed projects have development and operation life spans that exceed the normal career spans of design engineers or project managers. Excessively long development and operation cycles mean that the use of a "human" as the knowledge base manager has become a major barrier to success. In addition, there is a problem with the accumulation of knowledge based on experiences of similar situations. In contrast to this, a 60-month (or longer) development process has for decades been the norm in the automotive industry, whereas now the goal is to reduce this rate of development to less than 30 months. Dramatically reduced development cycles impair the formation of knowledge, denying the opportunity for reflection and experimentation. The compression of the engineering cycle time results in narrow focus of engineering efforts in critical points rather than the

development of a complete understanding of the features and characteristics of either the product or problem environment.

To give the engineer an appreciation of the life cycle implications of a design decision requires access to historical information. That is, there are many unquantifiable factors in the design process. The designer often makes judgements without an awareness of their implications which manifest themselves much later in the product life cycle. In large, long-life, complex products such as modern weapons systems the original design team is long since gone when the product exhibits design problems. The only way to provide the new design team with insight into the correlation between the design decisions and their implications is to provide a means of capturing of the design rationale and correlating this rationale and the product definition information to the effects (lessons learned or experience of the manufacturing or maintenance engineering activities). Both the original designers and the sustaining engineering staff need access to manufacturing, maintenance, and logistics experience.

## 2 THE FLOW OF KNOWLEDGE REQUIRED TO SUPPORT CONCURRENT ENGINEERING

There is a need for flow forward and flow in of knowledge relative to the product development cycle. Design rationale, alternatives considered, and lessons learned in the process of life-cycle design/engineering must flow forward to the next set of individuals in the development chain as shown in the upper portion of Figure 1. Similarly, experiences gained, lessons learned, and limitations discovered must flow from the downstream engineering activities of previous projects back to the upstream engineering activities of the current project as illustrated in the lower half of Figure 1.

The flow of knowledge that needs to be supported in Concurrent Engineering appears to be somewhat (though not radically) different from the flow of information studied for many years in support of information integration and information resource management efforts.

Relative to a particular product development effort, the knowledge flow tends to be unidirectional whereas the information flow is bi-directional as shown in Figure 2. For example, one of the primary flows of information into product design activities is feed-back information from product engineering and manufacturing engineering activities related to the current product definition. On the other hand, the flow into product design of concurrent engineering knowledge assistance is most often the accumulation of experience from many previous product histories.

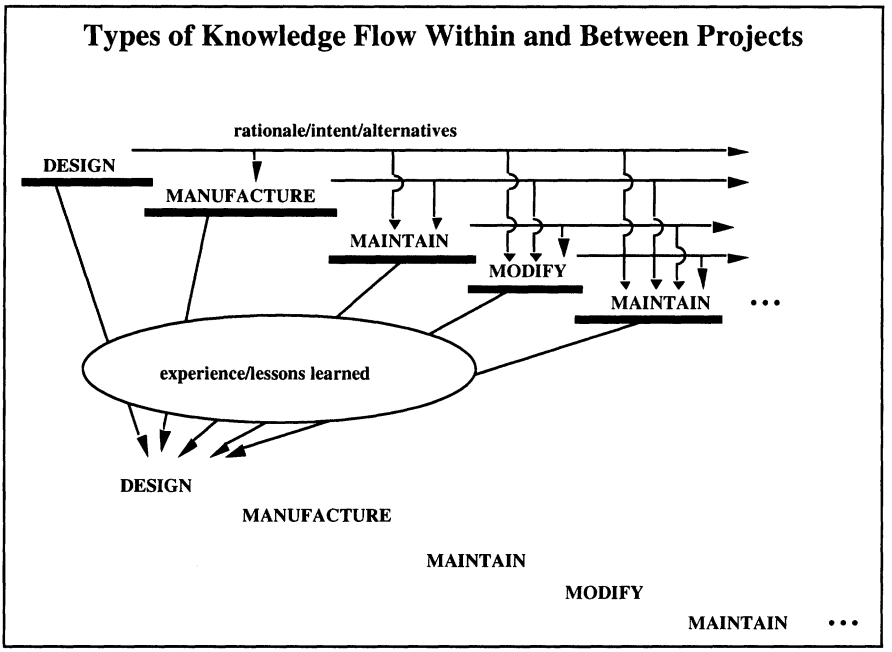



**Figure 1:** Typical types of knowledge flow within and between projects.

The loss of the current product definition as a frame of reference for the knowledge flow makes the management of, access to, and application of the knowledge resource much more difficult (as though management of the flow of information was not difficult enough).

In summary, there is a need for capturing of design knowledge

- To support passing on of that knowledge to future engineers.
- To support down stream decision making by other product designers.
- To support down-stream decision making by manufacturing engineers.
- To support down-stream decision making by field support & maintenance engineers.
- To support down-stream decision making by sustaining engineering efforts.

How can these knowledge flows be assisted and effected? First, we must be able to capture such knowledge efficiently . Second, we must be able to represent such knowledge. Third, we must be able to deliver the necessary advice or recommendations implicit in this knowledge to whatever task requires those recommendations. Fourth, we must be able to manage the evolution of such knowledge because we do not want to deliver out of date recommendations, nor do

we want to accumulate lessons learned on out of date product definitions, nor can we afford to deliver recommendations on out of date product definitions. Fifth, we need more efficient means for generation of accurate and complete specifications as a part of the engineering conception / analysis / decision making process and not as an after thought. Finally, we must improve the use and availability of engineering performance models.

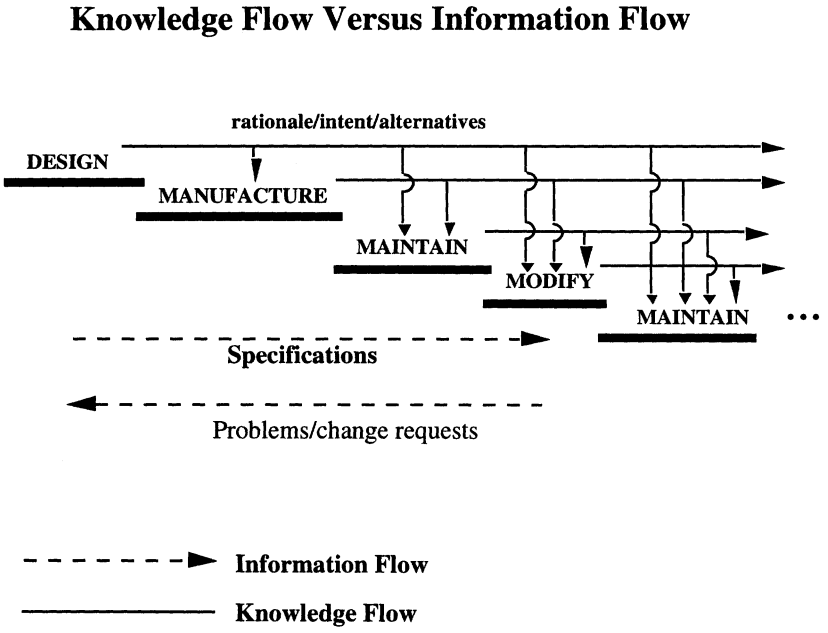



**Figure 2:** Predominant flows for knowledge and information.

The product design/engineer needs to have faster access to product design evaluation mechanisms. The manufacturing planner/engineer needs to have faster access to process / plant design evaluation mechanisms. The maintenance engineer needs to have faster access to product design / process / plant design modification evaluation mechanism. These types of performance prediction models speed up the acquisition of knowledge by these various engineering groups.

In this paper, we describe some of the general characteristics of how multiple forms of knowledge can be treated, and then we describe two specific forms of shape indexed knowledge and container object knowledge representation methods that we have developed.

The system requirements for developing a knowledge based design assistants - product designer system that can assist in producing more nearly optimal designs in less time are also addressed.

## 3 DESIGN KNOWLEDGE ORGANIZATION AND RETRIEVAL

The issue of knowledge capture, storage, and retrieval is primarily one of organisation and retrieval as distinct from representation and matching. Organisation differs from representation in that it is focused on structuring knowledge and information together to support use by multiple processes. Previous efforts in this area have traditionally focused on attempting to add flexibility to a base representational scheme and interfaces to traditional data base systems. Generally the result is a representational scheme that is inefficient or ineffective for all but the class of problems addressed by the original base representation.

Retrieval is distinguished from matching since matching is the dominant process required for support of the reasoning methods described above. Retrieval, on the other hand, has as its domain the management of memory use and object indexing. Retrieval is concerned with locating and retrieving collections of knowledge units that possibly contain the structures needed by the reasoning process. In the retrieval process prototypes, matchers, and candidate objects are all treated equally. In other words, the retrieval resources can be used by a reasoning process to support its matching of prototypes against candidate objects, in which case the role of the retrieval mechanism is to produce as small as possible a set of candidate objects that potentially fit the matching specification of the reasoning mechanism. Similarly, the retrieval mechanism in support of a design decision support process could return a reasoning mechanism (i.e., a collection of patterns and a matcher) to support a particular phase of the decision support activity.

The general form of knowledge organisation proposed is independent of the underlying knowledge representation. It is based on a treatment of each element of knowledge as an independent object with associated descriptions and procedures. The remainder of this section will focus on three new knowledge representation and reasoning methods developed to fill critical voids in the capture and delivery of design knowledge and product experiences within a concurrent engineering environment.

### 3.1 Shape-based knowledge representation and reasoning

Shape Matching is an important issue in Unified Life Cycle Engineering (ULCE). The research results addressed in this section relate to two important aspects of this problem: (1) finding potential problem regions, and (2) matching the geometrical model of a potential problem region with problem shapes that were described by

manufacturing experts and stored in a product life cycle experience knowledge base. We propose a method using object descriptions in terms of their surfaces. The surface of an object is described by segmentation into surface patches and the complete description consists of each patch's normal and their interrelationships. The partition and description of the surface is based on measured curvature properties of the surface. We believe that our chosen representation has many advantages for potential problem shape finding and shape based associative matching for retrieval of life cycle experience from the knowledge base. It also enables us to construct alternative shapes on the part design as advice (or work-around) to potential problems (Ting-jung, 1989).

As previously mentioned, we have chosen partitioned surface descriptions for representing objects in the potential problem matching work. The automatic identification of potential problem shapes has not been addressed in the research community to date. The difficulty of performing this task automatically is similar to the difficulty of performing it manually. That is, how can you tell a designer to avoid a shape that he knows nothing about. Since these shapes can result from the interaction of more than one part, they tend to evolve as the design evolves. We are proposing an approach that is a variation on the theme "common things occur commonly". More specifically, we are proposing to break up complex shapes into component shapes by examination of "unexpected" changes in the shape definition. The discontinuities (breaking surfaces), creases (radical change in curvature), and limbs (connection curve between surfaces) are chosen to be the factors for partitioning surfaces as they are explicit shape determiners of the surfaces of 3-D objects (see Figure 3). These factors are also being used for localising the region for potential problem shape checking.

In the first stage, the object represented with surfaces is partitioned at discontinuities which can be detected by examining (1) the zero-crossings, (2) limbs which are formed by the boundary of two connected surfaces, and (3) creases which can be found by evaluating the extreme values of surface curvature. As shown in Figure 4, these detected discontinuities, limbs, and creases are then used to partition a complex surface into patches. These patches can be subsequently approximated by planar facets. The region which hold the property of discontinuity, limb, or crease would be localised for problem shape checking.

After the patch generation stage, the system would query the designer about the discontinuity situation. If it was a design error, the designer could request the system to reconnect the broken surface to its neighbouring surfaces using a surface smoothing algorithm. Along the boundary curve between two surfaces, the vector field formed by the normal vectors of patches would be checked against the vector field of the problem shapes in the database (see Figures 5 and 6). If indexing the isolated shape into the knowledge base, the system determines a potential problem shape has been detected and will then report the problem to the designer and provide a suggested modification.

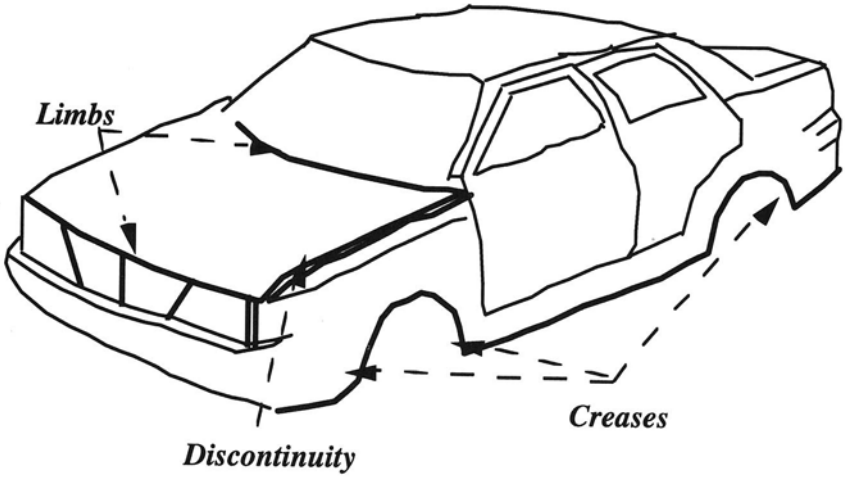



**Figure 3:** Characteristics of problem shapes.

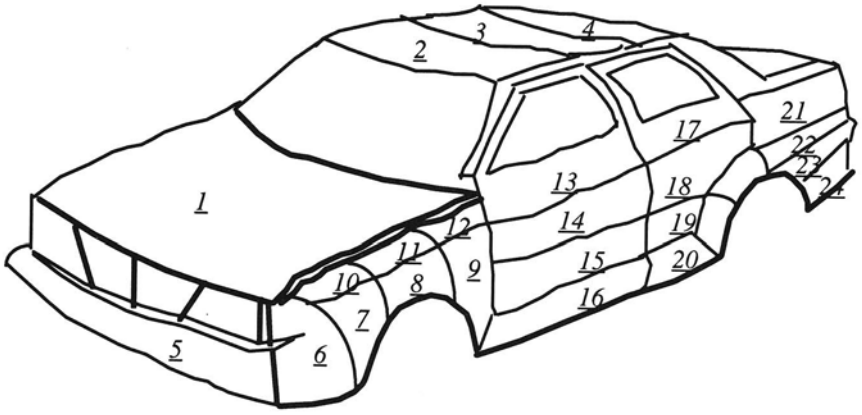



**Figure 4:** Potential problem shape analysis - surface partitioning.

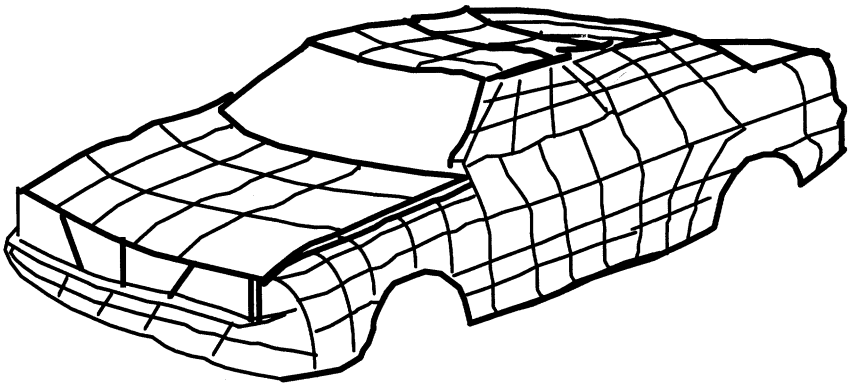
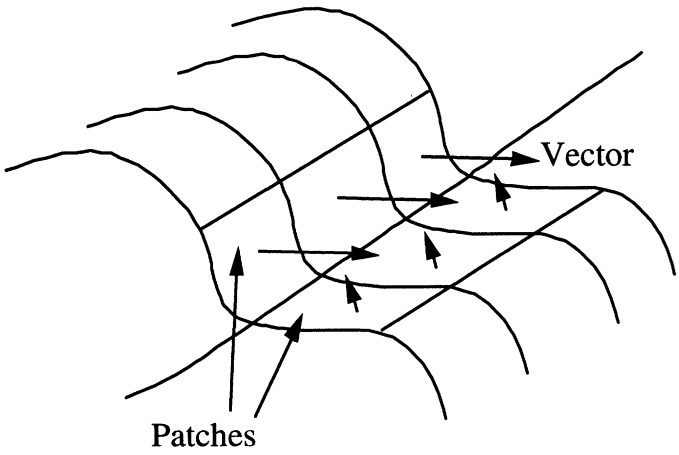


**Figure 5:** Surface patch generation for potential problems isolation.




**The vector field formation along the boundary curve between two patches**

**Figure 6:** Boundary curve traversal for shape recovery.

This potential problem shape detection method can be described algorithmically as follows:

### *3.1.1 Algorithm for shape extraction*

Given a set of surfaces (planar or sculptured) which represent an object:

1. Compute curvature for each sculptured surface and find the extremum of the curvature.

Note for a general space curve **r** = **r(t)**, the curvature **k** is obtained by differentiating r(t) twice. Thus

$$\dot{\mathbf{r}} = \dot{s}\mathbf{T} \text{ and } \ddot{\mathbf{r}} = \ddot{s}\mathbf{T} + \dot{s}^2\mathbf{k}\,\mathbf{N},$$

where **s** is the arc length and **T** is the unit tangent vector.

For a curve **u** = **u**(t) on the surface **r** = **r**(u,v) the curvature can be obtained from the following equation :

$$\ddot{\mathbf{r}} = \ddot{s}\mathbf{T} + \dot{s}^2\mathbf{k}\,\mathbf{N} = \frac{\partial^2 \mathbf{r}}{\partial u^2}\dot{u}^2 + 2\frac{\partial^2 \mathbf{r}}{\partial u \partial v}\dot{u}\dot{v} + \frac{\partial^2 \mathbf{r}}{\partial v^2}\dot{v}^2 + \frac{\partial \mathbf{r}}{\partial u}\ddot{u} + \frac{\partial \mathbf{r}}{\partial v}\ddot{v}$$

If the extremum of the curvature is greater than a predefined tolerance, the surface is then partitioned along the extreme curve.

For each subsurface, recursively subdivide the surface until the extremum of curvatures of all sub-surfaces are smaller than the predefined value.

2. Check the discontinuities among surfaces using the crossing zero method. If discontinuity is detected, query the designer for the reconnection.

3. Generate patches for each subsurface and finer (more) patches are generated along the surface boundaries.

4. Collect normal vectors of the patches to form vector fields along the boundary among surfaces.

5. Check the vector field against the vector field of potential problem features stored in the knowledge base. If a similar vector field is detected, report the potential problem to the designer, and suggest a modification.

## 3.2 Feature-based associative retrieval

The understanding of named shapes (features) is an important link in the unification of the design, manufacturing, and maintenance components of any Unified Life Cycle Engineering (ULCE) support environment. Global understanding of shape and form comes from the ability to characterise a part in terms of its constituent features. The constituent features may be specific to design or manufacturing processes. Because of the necessity of being able to associatively

retrieve manufacturing and design information based on form features, it has become imperative that the major thrust in research in this area should be in feature recognition, and in feature oriented CAD systems.

Research in the feature recognition area to date has been pushed by research in process planning. The most promising techniques that have been tried to date include:

a) Volume Decomposition (Woo, 1982);
b) Syntactic Pattern Recognition (Kypriano, 1980);
c) Logic Programming (Henderson, 1985);
d) Graph Algorithm (Joshi et al, 1988, Sakurai et al, 1990, Marefat et al, 1990), (De Floriani, 1989);
e) Convexity and concavity of faces - (Nnaji and Liu, 1990, Gadh et al, 1992).

In this discussion, it is important to distinguish between Geometric and Topological modeling:

i) Topological modeling describes the relationships between primitive geometric elements.
ii) Geometric modeling defines the shape, orientation, and location of each topological element.

For a feature representation/extraction system to be useful for providing keys for database retrieval of parts, it should have the following properties:

a) The representation must be unique.
b) The representation should not be brittle, i.e., subtle variations in parts should not produce drastically different feature representations.
c) The representation must allow the definition of its own limits. The limits of the representation should be precisely defined, i.e., the system should be able to warn engineers about topologies that it cannot classify.

Henderson uses a depth first search strategy on the boundary representation of the stock minus the part to pattern match features defined in terms of rules. On matching a feature, it is subtracted from the boundary representation of the material to be removed and asserted as a fact. The problem with this approach is that the subtracting of features from the material to be removed may render other features unrecognisable. Furthermore, this approach requires the definition of raw stock.

Syntactic pattern recognition was used for group technology coding by Kypriano and Jared at Cambridge University. It relies heavily on local information within the boundary representation, such as loops of convex or concave edges. It does not handle interfering features very well.

Volume decomposition involves searching for patterns in an object's Constructive Solid Geometry (CSG) representation, by first subtracting the part from the convex hull of the part, and then recursively subtracting the difference

from the convex hull of the difference until only convex differences remain. It should be pointed out that this restricted CSG representation is unique, but in general CSG representations are not unique. While this approach can provide a unique representation of the features of an instance of a part, the representation is brittle. Because slight variations of the same part could produce radically different CSG trees, the representation is not useful as a key for database retrieval.

De Floriani uses a generalised edge-face graph to represent the topological information about the part. The feature recognition algorithm partitions the graph into bi-connected and tri-connected components that it uses for recognising form features such as protrusions, depressions, through holes, handles, and bridges. The decomposition of the edge-face graph into bi-connected and tri-connected components is a directed non-cyclic graph, called an object decomposition graph. The object decomposition graph can be used for associative database retrieval of parts based on shape and form, because it provides genus, shape and feature keys in the patterns of bi-connected and tri-connected components. Parts with similar arrangements of features can be retrieved by resolving sub-graph isomorphism between the key decomposition sub-graph and the decomposition graphs of parts in the database. It is possible, however, to define patterns for which no recogniser exists; for example, De Floriani's system cannot classify features that cross more than 3 faces.

As an example of this procedure, Figure 7b shows the decomposition of the part shown in Figure 7a into three tri-connected components T1', T2', and T3', all having Face 1 and Face 2 {f1, f2} as a separation pair. Figure 7c shows the object decomposition of the edge face graph shown in Figure 7b, where it can clearly be seen that T1, T2, and T3 are adjacent components. T2 is a closed component as l1 and l2 are closed loop nodes, as shown in Figure 7a. T3 is a trivial component, and T1 is an open nontrivial component since the edge described by T3 does not belong to T1. A topologically-based form feature-based representation, is better suited for feature indexed knowledge base retrieval than a CSG sub-graph isomorphism.

## 3.3 Shape-based design knowledge retrieval

We developed our shape-based design knowledge by classifying the approaches taken by prominent researchers in the field of feature recognition. In our survey we found that no consensus exists on such fundamental issues as: (1) the definition of a form feature, (2) the appropriate mathematical definition of a form feature, and (3) whether or not there exists a canonical set of form features. To illustrate these points, the following definitions for form features have been proposed in the literature:

a) A region that alters the convexity of a part. (Woo, 1975)
b) A region to which has topological, geometric, or functional significance in a particular domain. (Requicha, 1989)

c) Recurring patterns of information related to a part's description, relating nominal size and shape, precision, and material. (Shah, 1988)
d) A higher level semantic description of a part in terms of a specific domain. (Marefat, 1990)
e) A single face or a set of contiguous faces called a face set, and a set of characteristic patterns in topology and geometry defines each shape feature. (Sakurai, 1990, Gossard, 1988).
f) Regions of a part having some manufacturing significance in the context of machining. (Joshi, 1988)
g) A specific geometric configuration formed on a surface, edge or corner of a work piece, intended to modify outward appearances or to assist in achieving a given function. (Shah, 1988)
h) A geometric form or entity that is used in reasoning in one or more design or manufacturing activities. (Duffey, 1988)

After reading these definitions, the reader might justifiably be perplexed by: (1) the apparent lack of consensus on the definition of form features, (2) the fact that each of the definition is true in one way or another, and (3) the fact that they are somewhat contradictory. The solution to this conundrum is that the researchers were defining features for different purposes. Figure 8 shows a shape acquisition framework that we propose, which places these definitions in perspective. In Figure 8, the arrows to the left, top, right, and bottom, of each box represent inputs, controls, outputs, and mechanisms, respectively, used by the IDEF0 function modeller (ICAM, 1985). Some of the definitions deal with the recognition of geometric and topological patterns, others deal with matching these patterns to prototypical shapes, while a third group deals with the interpretation of feature shapes using AI techniques. In our approach to this shape acquisition problem, the manner in which the acquisition is performed in each stage, is controlled by a user selected frame of reference. The geometric algorithms we have developed for shape acquisition consist of (1) predicates such as coplanar, intersect, and convex, (2) derived properties such as area, perimeter, and volume, and (3) constructions such as convex hull, minimum enclosing box, and Boolean operations. Examples of the graph search algorithms that are used include articulation, bi-connected and tri-connected components, and depth first search (De Floriani, 1989). The AI inference algorithms that have been modified, for use with these geometric algorithms, are: abduction (Poole, 1990), truth maintenance, and constraint propagation.

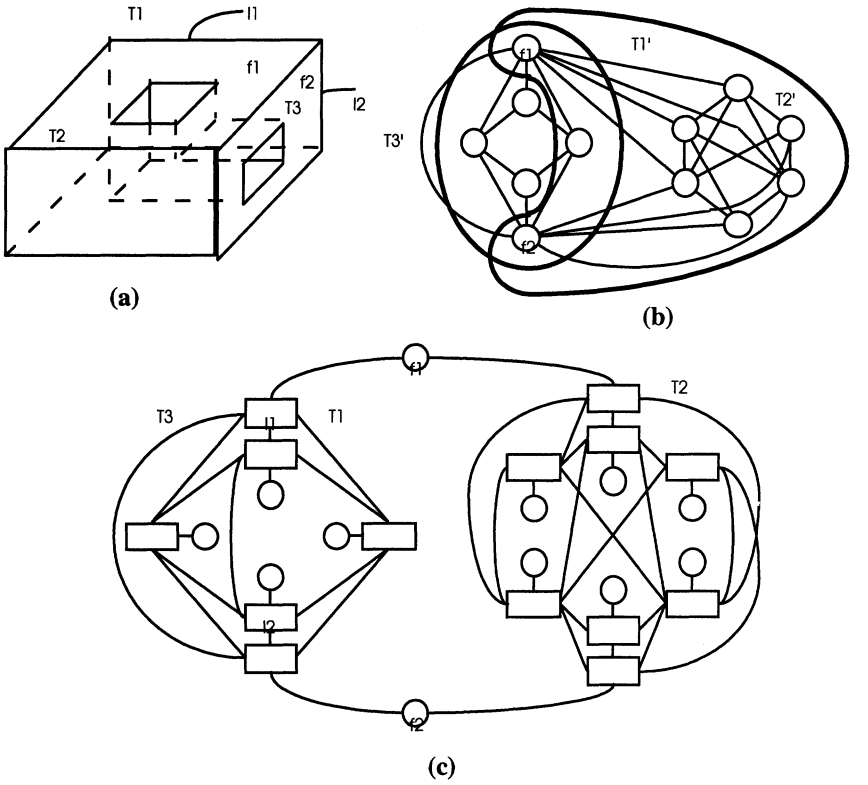



**Figure 7:** Example of graph networks for topological form feature representation.

## 4 DESIGN INFORMATION DELIVERY - PRODUCT DESIGNER SYSTEMS

Design information sources typically used by a human designer of mechanical parts include geometric data, engineering domain principles, material properties, cost data, analytic models, and product life cycle histories, among others. However, current automated design aid systems are unable to access all of these information types in any integrated way, and these systems typically have no understanding of the information usage patterns and design development rationale employed by the human designer. One might say by rough analogy that current automated design aids are at the "calculator plus pen and paper" stage, not at the spreadsheet stage. We have built / are building design support systems using a variety of these components. Such design support systems are individually useful,

however, the greatest long-term benefit to industry will be gained from developing a tool box of utilities for constructing specific design support systems.

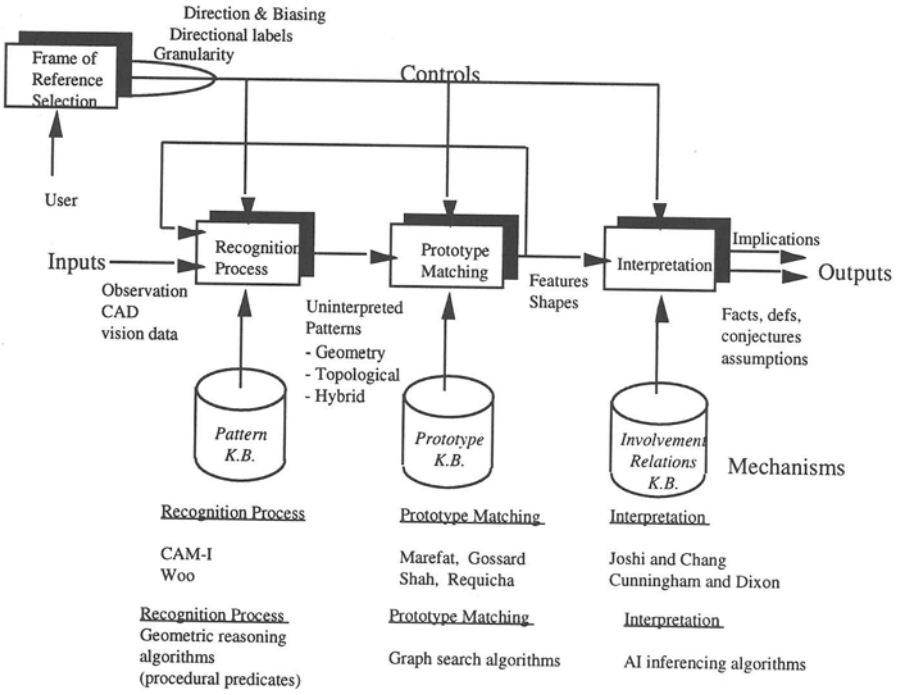



**Figure 8:** A framework in which the feature definitions found in the literature are compatible.

## 4.1 Characteristics of designer systems

One important characteristic to recognise about "designer" systems is that they are rarely autonomous. Rather they tend to provide more of a "design assistant" capability that improves the efficiency and productivity of the human designer. These systems usually provide a "manual" as well as "automatic" mode of operation. In the manual mode they provide design history management and high productivity interfaces to the traditional performance analysis tools. In the automatic mode they provide services ranging from qualitative assessment (e.g., manufacturability, cost, reliability etc.) of proposed designs to computer generated and analysed system designs.

The designer system models both the domain reasoning and the design process reasoning of an expert engineer as he generates initial design specifications. Inputs to such a system typically comprise a description of the environment in which the

system will operate from the perspective of the system, i.e., the environment parameters that constrain the performance requirements on the system, a specification of test conditions (e.g., horsepower, speed, ambient temperature), and certain technical or administratively directed constraints on the system. In the process of producing acceptable design specifications, the system iteratively proposes specifications, tests these using an analytical performance prediction program, evaluates the test outputs, and revises the design specifications. The analytical program models state defining characteristics of the system in the context of the related environment. The Design Assistant does not search exhaustively, but uses heuristics gleaned from the expert designer during the redesign stage. Based on these concepts, we derived a set of requirements for developing the proposed designer system:

1) The use of existing engineering analysis models as evaluators on a design. An important issue in the use of such models involves the acquisition of the portion of the designer knowledge base that defines how the system being used will understand the capabilities, resulting outputs, and particularly the limitations and underlying assumptions of the analysis program.
2) The use of curve based reasoning as part of the reasoning model. Engineers frequently think in terms of curves (or sometimes surfaces) when relating design parameters to performance expectations.
3) The use of truth maintenance techniques. Mechanical design engineers typically think in terms of alternative design proposals, that is, they will freely make and retract assumptions as the design process proceeds. Truth maintenance techniques are useful in managing the complexity generated by this situation in a complex design history.
4) Highly flexible user presentation. Designers rarely take one prescribed path to a solution, but may change viewpoints readily. To support this process, an automated design support system must allow the user to choose to look at various aspects of the problem freely, and at varying levels of detail.
5) The support for integration with product definition databases including both geometry and non-geometry based systems.
6) "Special study" design, analysis, and presentation. The fact that a design satisfies requirements and constraints may not be enough for the design to gain acceptance in an organisation; rather the design engineer will be expected to demonstrate the design rationale, show why the design is better than other satisfying designs, and answer "why not" questions about alternatives. This implies the ability to maintain the design rationale, to do comparative analysis on test results, to do sensitivity analysis on designs, and to produce a suitable presentation of the results.
7) The use at varying levels of detail of engineering models used for performance analysis. A design support system should accommodate system specifications at varying levels of detail, e.g. a component may in one case be modelled as

simply an output, in another as a set of parameters that are manipulated by a routine simulating performance.

## 4.2 Components of designer systems

The principle components of a designer system are represented in Figure 9. These components are not shown in any particular structure primarily because the structure of interconnections is generally left up to the session user. The cognitive process of the life-cycle engineering activities is a rationalised exploratory, learning process generating its own internal structure as it proceeds. Any attempt to impose a defined pattern on such a process by a tool may work well for one problem instance and then fail miserably for another (even similar) instance. Therefore, our approach to this problem is to build on systems that have been implemented to develop more of an "object" based approach to the development shell components. Under this approach these components become programmable services in the respect that a "designer" system developer would be able to tune the generic services to accommodate the needs of a particular domain. He could even provide a structured interaction for new users. However, as soon as the users have progressed to the point of understanding the protocols of the interactions between the objects they can quickly develop their own specialised interfaces. One important note on these components is that in any one application there are almost always multiple domains whose knowledge is a part of the services provided by the "Engineering Domain Knowledge Representation" component. This is even more the case in Concurrent Engineering Applications.

## 4.3 Knowledge acquisition approaches for designer systems

Knowledge acquisition in the life-cycle engineering areas described above can be a time consuming and tedious task. Of all the traditionally studied problems of knowledge acquisition (internment of the process as well as the domain knowledge etc.) the most serious problem with knowledge application in the life-cycle engineering domains is the lack of a written corpus of cases that can be analysed. Part of this problem is due to the previously noted use of "languages of thought" which are often semi-formal symbol systems. These symbol systems carry a rich semantics and generally (from a computer point of view) a complex syntax.

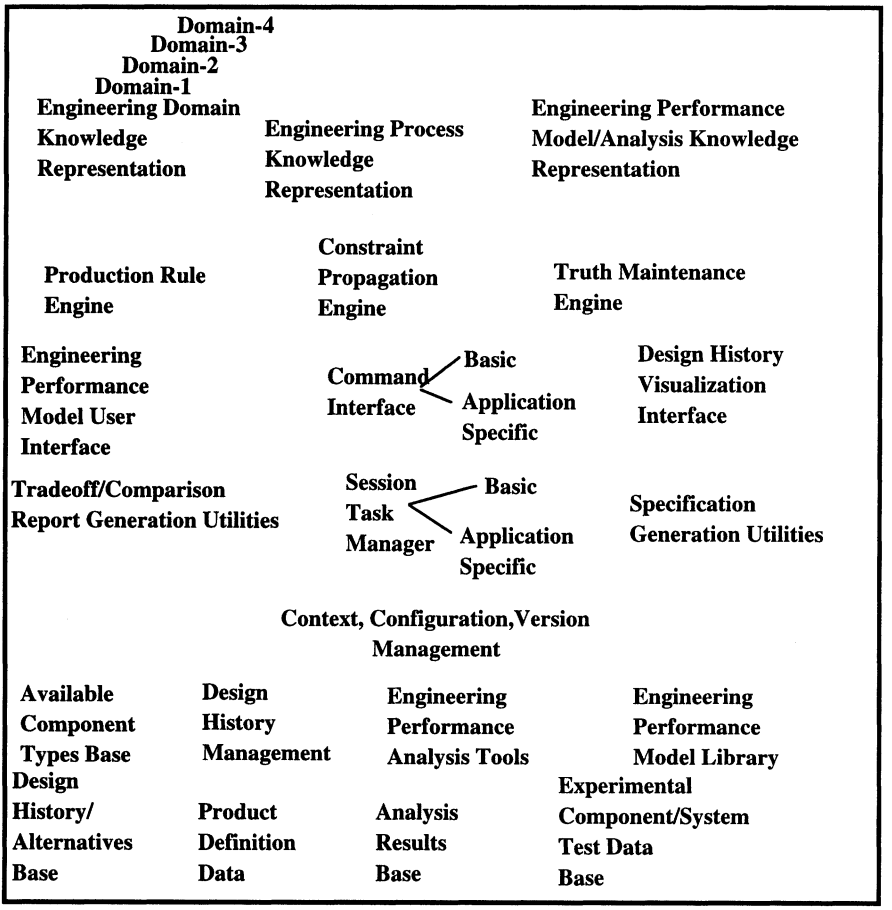



**Figure 9:** Generic component objects in designer systems.

Complexities range from the definition of an entire sub-language to the use of complex two dimensional (graphics/iconic based) languages. Part of the problem of knowledge acquisition in these domains then is the problem of assisting the domain experts in a formalisation of their languages of thought. Another is the creation of note-making environments that can be mixed into early prototypes of a designer system that the human expert can utilise to capture his comments as he uses the system. We have also experimented with the delivery of sophisticated user interfaces to existing engineering performance modelling analysis programs as a first step. These systems offer the human user a direct benefit and hence an incentive for use. Into these systems then we can integrate session monitors (flight recorders of a sort) as well as note-making capabilities described above. Once in

place, the information collected by these systems can be periodically recovered and studied by the knowledge engineers.

To expect to produce anything other than "crude" automated knowledge acquisition aids is believed to be highly presumptuous . However, there are utilities that can greatly assist human knowledge engineers in their acquisition, organisation and analysis of the domain knowledge in life-cycle engineering knowledge bases.

## 4.4 Knowledge application approaches in designer systems

There are four basic approaches to the application of acquired engineering knowledge in designer systems:

1. Heuristic guided search using generate and test methods.
2. Design option generation through back-solving of the performance models followed by heuristic based option ranking methods.
3. Structured selection using constraint tree representation of the design knowledge.
4. Constraint satisfaction supporting a combination of structured selection and heuristic guided search.

The first approach is illustrated in Figure 10, each node in Figure 10 represents a possible rule generated design configuration. Those nodes that are connected with links represent possible "follows from" relationships. That is, one node is generated based on the analysis performed on its parent node. It is also important not to misinterpret the illustration as though each generation is based strictly on the results of an analysis of its direct parent. In fact, the "next step" decision is generally based on every node that has been generated to that point. As can be seen from Figure 10, there are nodes in the design space that may not be generated at all by a particular execution of the designer system. This can be the result of the "learning" process that the system undergoes as it solves a particular instance of the design problem (program learning can be as faulty as the human counterpart). The thicker links represent a level of heuristic based "belief" on the part of the "designer system" that the pursuit of that particular link is more advantageous (such heuristic-based beliefs can also be erroneous).

The second common knowledge application strategy uses analytic methods and quantified constraints to generate the entire set of acceptable solutions. The design knowledge in these cases usually takes the form of the constraint set or heuristics for rank ordering the resulting set of acceptable solutions. The application of the configuration analysis techniques is reserved for the more detailed evaluation of only the top ranked techniques.

## 4.5 Generic utilities required to develop designer systems

A set of utilities is required to develop designer systems:

1) User interface / composite part representation construction utilities. The automated design of mechanical systems frequently involves a large representation structure for the objects being designed, and a sophisticated user interface for intelligent management of the input of object descriptions. A utility being developed facilitates the rapid prototyping of these interfaces and automatically generates the representation structures (which are accessible either to a rule based reasoning system or to analytic programs) as the input formats are defined.
2) Knowledge base structures for supporting reasoning based on the identification of situation types in the design process. Frequently a designer will make design decisions based not only on the current state of the designed object and domain principles, but also on the current "situation" as determined by the history of design changes and evaluations to date.
3) Utilities for defining control structures to manage the interplay of numeric components, heuristic components, display and communication components, and so on.

### *4.5.1 Knowledge engineering utilities (representation and reasoning methods)*

By far the most common knowledge representation scheme used to date in designer systems is the production rule representation with an associated set of logical consistency constraints managed by an assumption based truth maintenance system. As described in the previous section, there are serious shortfalls in this area when confronting knowledge domains which have extensive geometry conception or recognition components. In these situations, the only options in the past have been to:

1) Develop linear language encoding (names or descriptors) for the "20%" of commonly occurring or most important shapes and use the traditional symbolic processing facilities.
2) Hard code sophisticated (but generally limited) geometry processing and reasoning mechanisms.
3) Avoid the problem as it is too difficult for current methods.

With the addition of the shape representation / reasoning capabilities as well as the container objects, the option will exist for the direct representation of shape indexed knowledge representation. The development of production rule facilities which we have found essential to the construction of designer class systems was based on a Rete net processor with additional language facilities:

1) Seamless integration with the host language environment.
2) Rule language facilities for both formulation of relational and object oriented database queries.
3) Rule language facilities for the convenient processing of the results of such database queries.
4) Rule language and processing facilities for matching against objects in the execution domain without the direct incorporation of those objects into the working memory of the rule processing system. This capability avoids the problems of data redundancy and currency control.

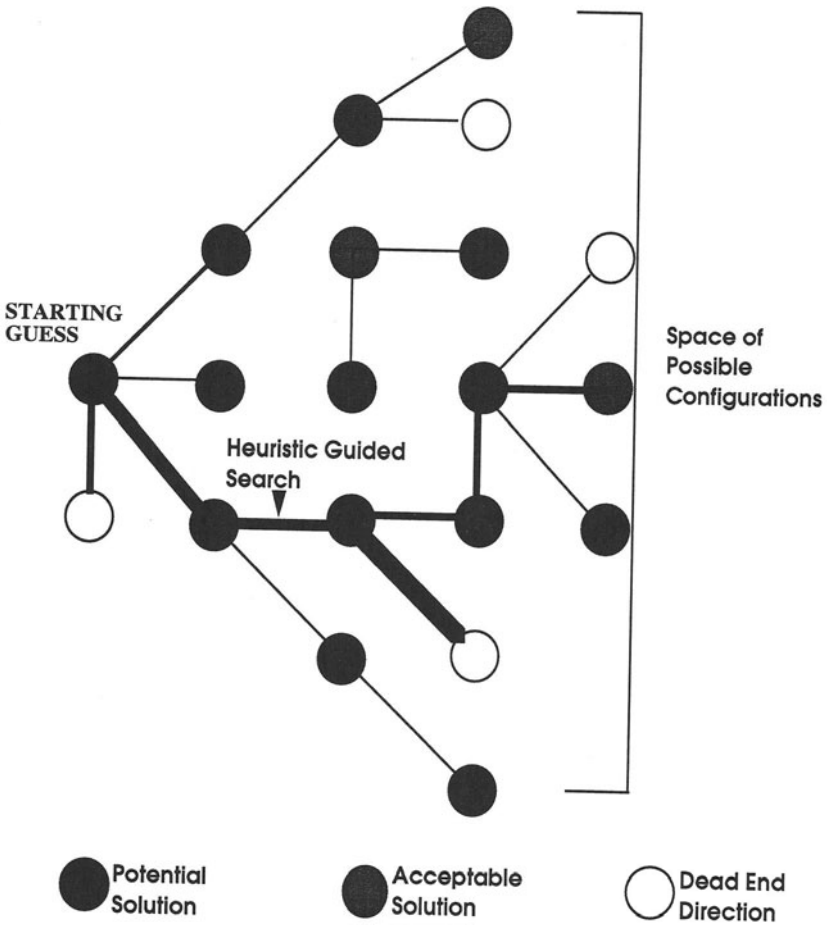



**Figure 10:** Heuristic guided generate and test.

### *4.5.2 User interface construction utilities (visualisation of the design history)*

Based on our experience in the development of designer systems the sophistication, intelligence and flexibility of the user interface is most often the determining factor of the acceptability of the system to the target end user. Besides the familiar desktop, windowing, menu, mouse and presentation manager capabilities, the developed designer system provides for the development of specialised visualisation support for:

1) Browsing a visual representation of the design history.
2) Multiple simultaneous display of the text/tabular results of performance analysis runs.
3) Ad hoc specification and simultaneous display of multiple graphs and charts for comparative visualisation of the results of performance analysis runs.
4) Visualisation of the base of components, design sessions, rule sets and other information, knowledge or processing resources to support end user management of these resources.

Our current approach to the provision of these facilities is based on the extensive use of the New-Flavors objects and the Symbolics Presentation and window manager systems. We are also continuing to monitor the progress of AFHRL IMIS model-based user interface strategies as a design philosophy for the underlying architecture of user interface construction utilities (Gunning, 1989).

## 4.6 Support for interacting / integrated designer systems

One of the implications of the establishment of an effective product design capability in an enterprise is the inevitability of the desire to construct not only individual designer systems but also federations of interacting co-operating designer systems. Several specialised utilities are needed to address problems of the nature of modelling the negotiations and compromising associated with team engineering of complex systems . The designer shell must provide support in the following areas in order to enable the construction of this class of truly simultaneous engineering applications:

1) Facilitator construction.
2) Decision scenario specification and implementation.
3) Blackboard construction.

Facilitators are knowledge-based components that support the interaction of a primary designer with the other primary designers in a co-operative design scenario. The role of the facilitator is to relieve the designer of the need to understand the complexity of the co-operative design process, thus allowing the reuse of the primary designer in many different co-operative sessions. The facilitators' knowledge base includes knowledge of:

1) The decision process.
2) The other agents involved in that process.
3) A model of its primary designers' activities.
4) Negotiation procedures for the critical state variables that define the coupling between the components being designed.
5) Error / failure mode diagnosis and recovery procedure knowledge.

The facilitator concept has been successfully implemented in complex co-operating systems involving many knowledge based agents with critical fault tolerant capabilities. However, the design of such agents can often be as complex as the primary knowledge sources they serve. Hence special support is required for decision scenario specification, blackboard construction and knowledge source analysis. The concepts discussed in this section can be incorporated into a "design support system" construction shell, i.e., a tool box designed to facilitate the construction of specific design support systems. Such a tool box will ease the development of design support systems able to use coherently a diversity of design information sources. The benefits of design support systems built with this tool box include reduction in the development and modification time for products and systems for the customer. The generic nature of the envisioned tool box will enhance commercial product design support systems that will improve the international competitive stance of industries.

## 5 REFERENCES

De Floriani, L., (1989), Feature Extraction From Boundary Models of Three Dimensional Objects, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, no. 8, August 1989.

Duffey, M.R. and Dixon, J. R., (1988) Automating extrusion design: a case study in geometric and topological reasoning for mechanical design, *Computer Aided Design*, Vol. 20, No 10, December 1988.

Gadh, R. and Prinz, F. B., (1992), Recognition of geometric forms using the differential depth filter, *Computer Aided Design*, **24** (11), 583-598.

Gossard, D. C., et al, (1988), Representing Dimensions, Tolerances, and Features in MCAE Systems, *IEEE Computer Graphics & Applications*, March 1988.

Gunning, D., (1989), A general Framework for Describing Human-Computer Information Systems Technical Report, AFHRL Wright Patterson Air Force Base, OH 45433, 1989.

Henderson, M. R., (1985), Extraction and organization of form features, In *Proceedings of Prolamat* 1985, 131-141.

ICAM Project 1701, (1985), Systems Engineering Methodologies, Vol 1-7, AFWAL/MLTC WPAFB OH December 1985,

Joshi, S. and Chang, T. C., (1988), Graph-based heuristics for recognition of machined features from a 3D solid model. *Computer Aided Design*, **20**, 58-66.

Kypriano, L. K., (1980) Shape Classification in Computer Aided Design, Ph.D. Dissertation, University of Cambridge, United Kingdom.

Marefat, M., and Kashyap, R. L., (1990) Geometric Reasoning for recognition of Three Dimensional Object Features, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 12, No 10, October 1990.

Nnaji, B. O. and Liu, H. C., (1990), Feature reasoning for automatic robotic assembly and machining in polyhedral representation. *International Journal of Production Research*, **28**(3), 517-540.

Poole, D., (1990), A Methodology for Using a Default and Abductive Reasoning System, *International Journal of Intelligent Systems*, Vol. 5, 521-548.

Requicha, A. A. G., and Vandenbrade, J. H., (1989), Spatial Reasoning for Automatic Recognition of Interacting Form Features, *ASME International Computers in Engineering Conference,* MA, August 1989.

Sakurai, H., and Gossard, D., (1990), Recognizing Shape Features in Solid Models, *IEEE Computer Graphics and Applications*, September 1990, 22-32.

Shah, J. J. and Rogers, M. T. , (1988a), Expert form feature modelling, Computer-Aided Design, Vol. 20, No. 9, November 1988.

Shah, J. J., (1988b), Current Status of Features Technology, *CAM-I report R-88-GM-04.1.*

Fan, Ting-Jung, Medioni, G., and Nevatia, R., (1989), Recognizing 3-D Objects Using Surface Descriptions, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 1989, Vol. 11, Number 11.

Woo, T. C., (1975), Computer Understanding of Designs, *Ph.D. thesis*, University of Illinois, Urbana-Champaign, Illinois.

Woo, T. C., (1982), Feature Extraction by Volume Decomposition, *Proc. Conf. CAD/CAM Technol.* Mechanical Eng., M.I.T.

## 6 BIOGRAPHIES

Chuan-Jun **Su** received the B.S. degree in applied mathematics from National Tsing Hua University, Taiwan in 1978. He received the Ph.D. degree in industrial engineering from Texas A&M University, College Station, Texas, in 1989. From 1989 to 1994, he was a visiting assistant professor in the Industrial Engineering Department, Texas A&M University. He is currently an assistant professor at industrial engineering department, Hong Kong University of Science and Technology. His research and teaching interests include CAD/CAM, Virtual Reality, Process Modelling, CAPP, and Information systems.

Professor Mitchell **Tseng** holds a B.S. degree in Nuclear Engineering from the National Tsing Hua University in Taiwan, a M.Sc. degree in Industrial Engineering

and a Ph.D. from Purdue University. He joined the HKUST faculty as the founding department head in 1993 after working in industry for almost two decades. He started his career in industry as a Manufacturing Engineer and progressed through several senior technical and management positions; including product development, information technology and system integration services. He previously also held faculty positions at University of Illinois and Massachusetts Institute of Technology. His research interests include process re-engineering, systems design, systems integration and product development.