FUNCTION DRIVEN DESIGN SELECTION OF PLASTIC INJECTION MOLDING FEATURES

Stephen L. Wood, Ph.D.,
Mechanical Engineering Department
Oregon State University
Corvallis, OR

ABSTRACT
Capturing the basic reasoning of why and how a design is developed is important in fully preserving and documenting the design of industrial products. To maintain captured design information and reuse the information during solution retrieval requires different types of information transferral and preservation than those traditionally used in CAD systems. This paper develops the basic database structure for preserving and transferring the design information along with using the preserved reasoning (functional properties) in the search for solutions. The basis for this paper was the development of a generic architecture for a function driven mechanical design solution library capable of being implemented in any form (i.e., object-oriented, relational, etc.) on a computer system. The solution library was initially targeted for the plastic injection molding domain, but applications from other design areas (e.g., sheet metal, casting, etc.) can also be implemented. The solution library uses a specific architecture to maintain and transfer information related to the interface between features. This architecture allows the designer to retrieve saved solutions through its functional properties, transfer the retrieved solution's information related to the developing design, and to capture the design reasoning or intent of the design during development. Consequently, 1) preserves the information of interfacing features within the product's database, 2) maintains a database of features with their fundamental properties and corresponding functions used by experienced design engineers, and 3) transfers the information within the solution's database to the design under development.

INTRODUCTION
Capturing the basic reasoning of why and how a design is developed is important in fully preserving and documenting the design of industrial products. To maintain the captured design information and reuse the information in solution retrieval requires different types of information transfer and preservation than those traditionally used in CAD systems. This paper develops the basic database structure for preserving and transferring the design information along with using the preserved reasoning (functional properties) in the search for solutions. The basis for this paper was the development of a generic architecture for a function driven mechanical design solution library capable of being implemented in any form (i.e., object-oriented, relational, etc.) on a computer system. The solution library was initially targeted for the plastic injection molding domain, but applications from other design areas (e.g., sheet metal, casting, etc.) can also be implemented. The solution library uses a specific architecture to maintain and transfer information related to the interface between features. This architecture allows the designer to retrieve saved solutions through its functional properties, transfer the retrieved solution's information related to the developing design, and to capture the design reasoning or intent of the design during development. Consequently, 1) preserves the information of interfacing features within the product's database, 2) maintains a database of features with their fundamental properties and corresponding functions used by experienced design engineers, and 3) transfers the information within the solution's database to the design under development.
the product's development (Form follows Function). The paper concludes with the extension of the architecture for components and assemblies followed by specific recommendations for integration with future CAD systems.

BACKGROUND

To implement a generic architecture or structure, for a function driven mechanical design solution library (Wood, 1995), the designer's desired functionality is used to obtain solutions, from the most primitive design structure - the feature, to the most complicated design structure - the assembly. Each potential solution is accessed through the functionality associated with it. The solution library, prototyped at the feature level using the object-based relational PC database "Paradox", uses the function information derived from the analysis of thousands of features on plastic injection molded parts developed in a previous study, (Wood, 1996). By using a feature's functionality for the search criteria during the design of mechanical components, the designer has access to a wide variety of design solutions related "functionally" to the problem.

The ability to present a wide selection of design solutions allows the designer to make use of technologies that he/she may be unfamiliar with. It is literally impossible for a design engineer to be an expert in all domain fields such as plastics, casting, machining, sheet-metal, etc., along with the usage of a variety of components which are found within the multitude of catalogs that exist in the design world. Consequently, a solution library can bring that knowledge and information to the designer when needed. A knowledge of the features, components and assemblies is contained within the solution library, so the designer need not be an expert in the domain field to use the solutions.

For this study, features, also known as "form" features, were the primary focus to test the architecture of the solution library. Each of the features satisfies a function (e.g., such as a hole to position an interfacing object). The term "feature" is an expression used in describing the characteristics of an object, which according to McGinnis (1990) comprises "any particular or specific characteristic of a design object that contains or relates information about that object," or according to Aasland (1993), "primarily 'chunks of geometry' distinguished by their ability to perform a function with one or more other features." These views, along with those recently presented by Shah (1991) in a comprehensive review of many current ideas on feature technology and terminology, all share the concept that features are central to design object modeling. Dixon (1988) proposed the method of designing-with-features, where the designer models the object with primary features that are converted into secondary features with additional information. In his model, primary features are formed from concepts the designer desires to express and manipulate, while secondary features contain information about applications (e.g., manufacturing features are secondary features to manufacturing methods and tool shapes). Designing-with-features "is a synonym for all design work involving predefined details, commonly details of shape, form features" (Sigurjónsson, 1992). Although features are used for many aspects of a design, the retrieval and application using the basic reasoning behind their usage has not been extensively investigated. For this study, features are considered the primary building blocks of a structure, i.e. the specific geometrical forms that satisfy the functional needs pertaining to a component.

Features have many functions associated with them (Wood, 1996); consequently, only the functions that are used predominantly by the engineers to satisfy design problems are used in the search for solutions. It is through the functionality of the individual features that a basic capability of capturing some of a design's functional intent is possible. Using the function information of each feature is necessary for implementing design reasoning into the next generation CAD systems.

How the term "function" is used in this study also needs clarification due to the many different interpretations of the word. A "function" is defined here as the behavior or action that the feature must satisfy in order for the product to achieve its overall purpose. In general, a function in its simplest form can be expressed as a verb (e.g., 'position', 'support', 'align', etc.) (Hundal, 1991). These functions are used by the designer to describe what the feature should do and are associated with the features according to their use.

The use of functions for the search for solutions is not new, prominent design theory researchers (Pahl & Beitz 1984, Andreasen 1991, Anderson 1991, Dighe 1992, Horváth 1990, Iyengar 1992, Bardasz 1990) have suggested solution library contexts revolving around complete design solutions. Other researchers (Cunningham 1988, Joshi 1990, Kimpel 1991) have investigated designing-with-features by using feature-based solution libraries, similar to the one developed here. These investigations are relevant because they are the first step towards designing with features, but they either have not made a complete use of the functional attributes or have not modeled the entire solution in a functional way.

Library systems based on complete product solutions are useful in their specific application but fail in several contexts. First, these library systems do not maintain the
development history of those solutions; consequently, the reasoning and detailed functional development of the design’s solution are unavailable. Secondly, the solutions accessed from a complete product library neither present solutions which are incomplete, nor present solutions from a completely different design domain, or from a different level of development (such as a feature to replace a component or a component to replace an assembly). This ends up reducing or limiting the development of new design variations and innovations. Additionally, the selection of a solution by the satisfaction of a function and the preserving of functional information from that selection is not a standard design practice.

Feature based solution libraries also have the deficiencies associated with library systems previously mentioned, but also possess the drawback of requiring the engineer to have an idea of the solution and its capabilities before actually using the library. This limits the design to solutions that are known to the designer. Consequently, the novice engineer will, in general, design with a limited number of features, and the expert with many. Other feature based design systems, such as ProEngineer (Parametric Technology, 1994), are prime examples of this deficiency; they require the feature and its data to be known beforehand. Additionally, these systems do not possess the reasoning behind the selection of a solution. To overcome these deficiencies, a "function driven" solution library has been developed.

The implemented solution library serves as a computer design assistant to aid engineers in the development of a design by using the functionality of the desired solution as the primary index to search for design solutions. This approach to search for solutions is accomplished by incorporating and using a solution’s known functionality (obtained from previous studies) into the library. Consequently, only solutions that satisfy the function required to be fulfilled by the designer are retrieved. The number and variety of retrieved solutions are related to the function or functions being searched for, thereby producing solutions that help the designer in developing a concept or during a redesign.

The solution library is accessed during the decision process where issues are decided upon through arguments that support a solution. In the design cycle, solutions are accessed from the library when alternatives are requested. The function which is at issue (i.e., to be satisfied), which was developed during the early stages of the design’s analysis, is passed to the solution library along with relating search criteria. This criteria may consist of any of the following: reasoning or usage of the feature, type of application, boundary conditions, initial constraints on the system, introduced constraints due to the relationships to adjacent objects, or the actual functional problem under investigation. An example of an issue in a design context could be to mount a roller bearing. The function "to mount" and the subsequent subfunctions (e.g., such as to support in axial and radial directions) are then used in the search for solutions.

ARCHITECTURE MODEL

To design a computer data structure or architecture model that maintains the search information, several main issues need to be considered. These include: (a) the definition of those elements that are used to store and retrieve the information from the search, (b) the selection of features, components and assemblies that satisfy the required function, and (c) the mechanism(s) that elaborate on both the found solution and the main search contexts for storage or retrieval. This section discusses these issues in the development of a search data structure or search mechanism which I term a "function-object."

The first issue is to develop the definition of the elements that are used to store and retrieve the information from the search. Many authors (Pahl & Beitz 1984, Tikerpuu 1988, Ullman 1992, Hubka 1984) have put forth useful and necessary database elements for feature and component storage. Some of these ideas have been combined into a comprehensive data structure specifically developed for the solution library. Specifically, the database contains information about what the solution is best suited for and why (the reasoning behind the solution concept).

To develop the what and why attributes (i.e., the functions and uses) within the "function-object," the relationships and attributes between features, components and assemblies must be determined. This was accomplished, for the initial solution library implementation, at the feature level in the plastic injection molding domain (Wood, 1996) by an in depth analysis of known features in that domain. With the knowledge derived from the study, the necessary database structure for the function-object was obtained. The information representing the relationships between features comprised of the attributes between interfacing features (i.e., the functionality and what occurred at the interface). The attributes were identified and incorporated in the search mechanism for the solution library. Some of the attributes contained are transferring of energy, materials and/or information between objects, a function/verb and noun used to represent the base relationship in addition to location, and other relating information.
The function-object is a database representation containing information from the engineering specifications, functional decomposition information and other relating data between features, components and assemblies. The function-object maintains the information about the relationships between the interfacing physical/implicit objects and interfaces with other function-objects. The function-object maintains four of the five items Ullman (1993) states in his OREO design representation model. In his model, one object's relationship to another is considered to contain: which objects are related, type of connection, type of transmission, and relation action. The fifth item, relative positioning, is not maintained in the function-object but within the part's database itself.

The function-object also contains information about the relationships between the objects and the functions involved. One of these, the degree of resolution or the complexity of the system or object has great importance. The degree of resolution is the degree that a function has been broken down or resolved. Where the function can not be resolved further is the lowest degree a design can be to be satisfied. Examples of a function include verbs like: represented. The degree of resolution maintains the level of abstraction within the developing function structure. Additionally, the degree of resolution aids in keeping track of each function level as each is satisfied with a solution. Other information contained within the function-object, which contain information about the relationships between objects, is the transmission or conversion of energy, information, and/or material data (Pahl & Beitz 1984, Hundal 1991) that is acted on by the "verb" between the objects.

The type of physical connection that is possible or directly specified between the two interfacing objects is also included within the function-object. This information provides connection information between components with a specific type and degree of freedom (DOF). According to Agogino (1988) this is necessary, "since the basic effect on a body of being connected to another is to reduce its freedom of motion, a classification of virtual bodies by degrees of freedom and type of motion is appropriate." Specifying the type of DOF presents a constraint which narrows the number of solution possibilities. The physical connection information is maintained in the function-object as the type-of-interface. This category includes the degrees of freedom, type of connection (e.g., [flexible, fixed, temporary, permanent],[surface, line, point],[rigid, planar pin, slider, gear contact, prism, helix, cylinder sphere, plane sliding]) and a part-of pointer to another object. (A receiving object is the object that receives the effects of the function. Or in other words, the receiving-object is the object that has been acted upon by the performing-object.)
**TYPE OF FUNCTION:**
(Functional, manufacturing, mating, assembly. These categories divided specify the basic domain the function is concerned with, i.e., functional - used in the product's application, manufacturing - used to aid in the manufacturing process, mating - used during the connection with other components, assembly - used in DFA (Design for assembly).

**PRECEDING FUNCTION:**
(The function that directly proceeds it in the function breakdown of the design that was developed earlier in the design process. The preceding function may be the parent or a brother/sister function. The preceding function is obtained when the current function (issue) under investigation is first passed to the function-objects.)

**Degree of Resolution for the preceding function:**
(The depth or complexity level of the function in the function breakdown. This depth, which may be represented in a tree structure format, shows the design's level of abstraction.)

**SUCCEDING FUNCTION:**
(The function that directly succeeds it in the breakdown - This may be a parent or a brother/sister function)

**Degree of Resolution of the succeeding function:**
(The depth of the current function/primary function under investigation)

**PARENT FUNCTION(s):**
(Parent functions from the function breakdown)

**CHILD FUNCTION(s):**
(Child functions from the function breakdown)

**Transfer at the interface making the function possible:**
The transfer at the interface of two objects is commonly agreed to be the transformation between an input and an output with respect to material, energy and information (signals) (Pahl & Beitz 1984, Roth 1987, Hundal 1991). This transfer at the interface is represented in the function-object by the following:

**MATERIAL FLOW TO:**
(Location/object material is transmitted to)

**ENERGY FLOW TO:**
(Location/object energy is transmitted to)

**SIGNAL/INFORMATION FLOW TO:**
(Location/object information is transmitted to)

**MATERIAL FLOW FROM:**
(Location/object material is transmitted from)

**ENERGY FLOW FROM:**
(Location/object energy is transmitted from)

**SIGNAL/INFORMATION FLOW FROM:**
(Location/object information is transmitted from)

**ENERGY FORM:**
(Type of energy)

**SIGNAL/INFORMATION FORM:**
(Type of information)

**ASSUMPTIONS:**
(Any known assumptions with respect to any or each of the above material, energy, or signal/information)

**Operational Information**
Operation information is the data that specifies or describes the interface between objects (e.g., features, components or assemblies) and is comprised of the following information:

**TYPE OF INTERFACE:**
degrees of freedom: (0 to 6),
type of connection:
(The type of connection represents the information at the interface junction: [flexible, fixed, temporary, permanent], [surface, line, point],[rigid, planar pin, slider, gear contact, prism, helix, cylinder, sphere, plane sliding])

part-of:
(the component or assembly the object is part of).

**Interfacing features, components, assemblies:**
(objects that interface with the feature)

**CONSTRAINTS:**
(Any constraints which are known to act on the system.)

**OPERATIONAL RELATION:**
(The relationship between objects in the transformed design object.)

**OPERATIONAL FUNCTION:**
(Function that represents the transformed design object - may differ from the above function)

It must be noted that other elements can be added to this function-object template as long as the above parameters are also included. The basic set above is necessary for the solution library to access the functional information for a search.

**AN EXAMPLE using functions and the function-object for searching for solutions**
The following example describes the use of functions to search for the necessary features required in the development of a design, and the mechanism(s) that
elaborate on the found solution and the search conditions for storage or retrieval. Note: The function-object information has been abbreviated for viewing the pertinent information.

This example describes the development of a bearing assembly for commonly used roller bearings using the required functionality to find the features. The assembly in its final consists of the bearing housing, mount/shaft and the bearing. For simplicity we will assume that the roller bearing mount/shaft has previously been developed and now desire to develop the bearing housing. The functions required to satisfy the design are used to obtain the solutions. Of the feature solutions presented by the solution library the designer selects only those that satisfy the functions and meet his/her specifications. Upon completion of the design example, an object diagram is presented showing the breakdown of the design assembly.

Before beginning the roller-bearing housing design, the basic knowledge of the roller-bearing is obtained such as inner and outer race diameters and thickness. This information is obtained to develop the initial constraints on the housing design. To start the design an initial blank-stock-form is used to represent the roller bearing housing. The initial parameters for the housing is selected by the designer to satisfy the boundary conditions of the outer and inner races of the roller-bearing and mount. Using this information the functions that are required to be satisfied by features on the roller-bearing housing design are determined:
* Position, mount and limit the roller bearing in the radial direction.
* Limit, position and support the roller bearing in the axial direction (right).
* Limit, attach, position and assist in the axial direction (left). The search for a component to position and limit the roller bearing in the opposite radial direction.
* Assist and guide the roller bearing onto the assembly.

To initiate a solution search for the first function parameters the function-object that is used to start the search receives the function information and any known constraint parameters. So the function object would appear as in figure 1. (Note: only pertinent data in the function-object for the example will be shown.)

Using this function-object and similar ones containing the functions and information of the second, third, and fourth function parameters (above) solutions are searched for and presented to the designer to. The designer then selects a "best" solution from the list presented and the function-object above is passed to the bearing-housing database attached to the solution that satisfies the function/s. Consequently, the information about the basic functionality and relationships between the objects is preserved. By linking the function-object to the design-object maintains the integrity of the design from the search. The search function-object shows the information that is required at the interface before the functions are used to find a solution form; this information is then passed to the object to maintain the information at the interface.

To begin the search, the first combination of functions (position, mount and limit, Figure 2) are used to obtain

Figure 2. First function combination

<table>
<thead>
<tr>
<th>Function/Verb</th>
<th>position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auxiliary Function 1:</td>
<td>mount</td>
</tr>
<tr>
<td>Auxiliary Function 2:</td>
<td>limit</td>
</tr>
<tr>
<td>Auxiliary Function 3:</td>
<td></td>
</tr>
<tr>
<td>Auxiliary Function 4:</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1 Function Object

| FUNCTION: Support |
| AUX. FUNCTION [1]: Mount |
| AUX. FUNCTION [2]: Limit |
| NOUN: Load |
| PERFORMING-OBJECT: Blank-stock part (housing) |
| INITIAL STATE of PERFORMER: Free radial movement |
| FINAL STATE of PERFORMER: Zero radial movement +/- tolerances |
| RECEIVING-OBJECT: Roller Bearing |
| PRECEDING FUNCTION: Unknown (for this example) |
| SUCCEEDING FUNCTION: Support (axial forces) |
| ENERGY FLOW TO: Blank-stock (housing) |
| ENERGY FLOW FROM: Roller bearing |
| ENERGY FORM: Mechanical |
| ASSUMPTIONS: Using Roller Bearing constraints |
solutions (Figures 3, 4, 5, 6). From the solutions presented by the designer to fulfill the initial roller-bearing housing functions.

Name: **Roller bearing housing hole** (Figure 7)
Design functions: Position roller bearing and support tangential forces and related movements.
Assembly functions: Stop motion during the assembly operation of the roller-bearing.

The function object within the housing database associated with the feature "hole" would be as follows:

The designer determines which solution would be most applicable to the design. Unfortunately, saving the designer's rational behind the decision is only possible through notes attached to the feature. This was accomplished by including a note pad for the designer's use. For the function combination case the hole can be selected
FUNCTION-OBJECT
FUNCTION: Support
AUX. FUNCTION [1]: Mount
AUX. FUNCTION [2]: Limit
NOUN: Load
PERFORMING-OBJECT: Blank-stock part (housing)
INITIAL STATE of PERFORMER: Free radial movement
FINAL STATE of PERFORMER: zero radial movement
+/- tolerances
RECEIVING-OBJECT: Roller Bearing
PRECEDEDING FUNCTION: Unknown (for this example)

SUCCEEDING FUNCTION: Support (axial forces)

PARALLEL FUNCTION: Null
DEGREE of RESOLUTION: 1/1
PARENT FUNCTION: Null
CHILD FUNCTIONS: Null

TRANSFER at the interface making the function possible:
MATERIAL FLOW TO: Null
ENERGY FLOW TO: Blank-stock (housing)
SIGNAL FLOW TO: Null
MATERIAL FLOW FROM: Null
ENERGY FLOW FROM: Roller bearing
SIGNAL FLOW FROM: Mechanical
SIGNAL FORM: Null
ASSUMPTIONS: Using Roller Bearing constraints

Operational Information
TYPE OF INTERFACE:

degrees of freedom: 0

- type of connection: temporary, surface, rigid
- part-of: unknown (for current example)
- Interfacing features, components, assemblies: Null

CONSTRAINTS: Cylindrical hole in blank-stock (housing)

due to Roller Bearing constraints

OPERATIONAL RELATION: unknown
OPERATIONAL FUNCTION: unknown

The selected feature is then saved with its relating function-object and checks are made to determine interfacing components and known connection types. The "part-of" is initially classified as unknown so the designer is requested a name: roller bearing housing. Next checks are made to decide what assembly the roller bearing housing is part of, i.e., roller-bearing assembly.

With the first feature selected the next function combination are used to obtain feature solutions; these functions are limit, position and support axial forces (right) and their related movements. A new function-object is created for this solution search containing the information of restraining and mounting roller-bearings in the housing, and the limit-movement, position and support-axial forces functions are used to as the search keys. The solutions found within the database to limit or restrain the movement in one axial direction consist of a boss, depression, groove, gusset, hole, protrusion, snap, and wall. Solutions for limiting and positioning are boss, depression, groove, hole, protrusion, snap, wall and solutions for limiting and supporting are boss, gusset, peg, protrusion, undercut, wall. Solutions that possess all three are the boss, peg, protrusion, wall. From these, the designer evaluates each solution according to the current design. For this case, four of the solutions fulfill the functional requirements, each with different means of solving the problem. From the found solutions the wall is selected.

Name: Roller bearing housing wall
Design functions: Support of axial forces and related movements.

Assembly functions: Stop motion during the assembly operation of the roller-bearing.

Once the wall solution is combined with the previous solution (Figure 8), the functions, limit, attach, position and assist are used to find a solution for the axial direction (left). A search to restrain the bearing in the opposite axial direction would come up with various solutions as presented before. Of these solutions, only the snap (see Figure 2) would satisfy the functional requirements. The other solutions, such as wall and boss, are not removable. Therefore mounting the roller bearing would be impossible.

For the prototype solution library, component solutions
have not been entered into the solution library. Once component solutions are entered in the library, a search for a component to restrain, attach, position and assist the roller bearing would produce a solution such as a retaining ring and corresponding groove. To accomplish this with features the designer must evaluate the solution accordingly. Consequently, the retaining ring is represented as a snap, since it locks in similar to a snap.

Searching for a solution solving the functions limit, attach, position and assist produces the solutions: depression, groove, and rib. The groove is selected to attach the retaining ring (Figure 9).

**Figure 9. Groove solution**

Name: Roller bearing housing retaining ring groove
Design Functions: To seat a retaining ring to support axial forces and related movements.
Assembly Functions: Realization of a detachable connection with the roller bearing (with the help of a retaining ring)

To finish the design of the roller-bearing housing, the functions assist and guide are searched for to obtain a feature to aid in the assembly of the roller-bearing on the housing. From the search solutions of chamfers, tapers and corner radii are found. The chamfer solution is selected to complete the design (Figure 10).

Name: Roller bearing housing chamfer
Assembly Functions: Aid in mounting the roller bearing during the assembly process.

Manufacturing Functions: Realization of a definite form of the edge.

All of the above features combine together to form the roller bearing housing (Figure 11). When the roller bearing housing is combined with the roller bearing mount and the roller bearing into an assembly the complete object structure would be as in Figure 12.

**Figure 10. Chamfer solution**

**Figure 11. Roller Bearing Housing with bearing and roller bearing mount.**

Name: Roller bearing housing
Design Functions: Position, support
Description: Bearing-seat housing: roller-bearing mount/housing.
Input: Load upon housing from roller bearing.
Output: Stresses on housing
Participants: Children: hole feature, wall feature, groove feature, chamfer feature
Type of interface: degrees of freedom: 0
type of connection: temporary, surface, rigid
part-of: unknown (for current example)
Connection requirement features, components, assemblies: Retaining ring, Roller-bearing
Figure 12. Object structure

This example demonstrates the search and retrieval of solutions using the function requirements that make up a design. The example also demonstrates how features are obtained from the functions derived from a function breakdown. With this technique features are combined into components which are in turn combined into assemblies.

The information residing in the function-object is used by the solution library to search for solutions and for representing the basic information about the interface between objects. The function information and the function-objects discussed in this paper were used for obtaining feature solutions, but adapting this concept to components and assemblies is also possible. The search for solutions of complicated structures requires delimiting factors to be used in order to reduce the potential search area. The number of component and assembly solutions is extremely high and requires the search area to be reduced to a manageable level. Solutions that are not directly applicable to the design problem inhibit the designer's ability to make sound design selections. To incorporate delimiting factors the same function-object is used as the primary search mechanism, but additional delimiting elements are implemented. The following delimiting component and assembly concepts were obtained from the research of Bardasz and Zied (1992). These concepts were modified and incorporated into the function-object to reduce the search space for component and assembly searches.

Type: (e.g., undefined, feature, component, assembly)
Phenomena: The physical phenomena used to satisfy the function/function-combination (e.g., friction, transverse stress, heat transfer).
Domain: The manufacturing domain (e.g., plastic injection-molded, sheet-metal, cast, composites, ceramics).
Class: A description of the design (e.g., bearing, gear, electric motor)
Description: A descriptive list of terms used to describe the situations under which the object best serves its purpose (e.g., helical gear: high load, high number of revolutions per minute, parallel shafts etc.).
Assessment: Limiting constraints (e.g., minimum weight, minimum cost, maximum speed, maximum efficiency, etc.).
Shape: The rough shape of the component or assembly (e.g., cylindrical, rectangular).
Type classification: Topology & shape form.

The retrieval of components and assemblies has been investigated by many researchers (Pahl & Beitz 1984, 1988; Hubka 1984, 1988, 1992; Bardasz 1990, 1992; Andreasen 1991; Sigurjónsson 1992). Of these, the research of Bardasz and Zied (1992) using an Analogical Problem Solving (APS) scheme, based on case-based reasoning using semantic network, is one of the most promising. Using this scheme, but only by focusing primarily on the functionality of the product as the primary index for the search, and by using the other criteria as delimiting factors, allows for the reasonable selection of solutions within the entire population of solutions. To accomplish this, the primary index is fixed and delimiting factors and indices are developed to be used during the design process.

Conclusion
Capturing the basic reasoning of why and how a design is developed is important in fully preserving and documenting the design of industrial products. Therefore, the maintenance of captured design information and it's reuse during solution retrieval requires different types of information transferral and preservation than those traditionally used in CAD/Design systems. This paper develops the basic database structure for preserving and transferring design information along with using the reasoning (functional properties) in the search for solutions. The designer retrieves saved solutions through functional properties, transfers the retrieved solution's information, and captures the design rationale or intent of the design during development. To accomplish these actions several main issues are considered: (a) the definition of the elements that are used to store and retrieve the information from the search, (b) the selection of the features, components and assemblies that satisfy the
function, and (c) the mechanism(s) that elaborate on both the found solution and the main search contexts for storage or retrieval.

The database structure, termed a "function-object," maintains the functional, constraint, and information parameters that occur at the interface between two objects. This function-object is used in the search for solutions by using the contained function information for the solution search keys. The function-object is additionally combined with the object's (feature, component, or assembly) database to maintain the connection information with other components. It is through the function-object that the design's purpose or "reasoning" is preserved.

The information presented in this paper targets some of the aspects that need to be developed and applied to future CAD/Design systems. First, a new generation CAD system should include feature based design that is based on functional reasoning. This system should focus on features and the functions they fulfill. This would allow designers to develop a product through the fulfillment of its functions (function generates form). Second, a system using features and functions could capture some design rationale, but only with features and functionality embedded in the system. Third, once the features and functionality are contained within the system, the product under development could be understood more completely by evaluating the multifunction aspects of certain features. Fourth, the features and corresponding functionality could help designers from one discipline understand features specified by designers from another, and thus avoid design conflicts. Fifth, since the interaction between components in a product is at the interface between features, it is necessary to understand the interface or the functionality that is represented at the interface. Only by introducing functionality into CAD algorithms can feature-based design be effectively used.

REFERENCES

Aasland, K., 1993, Conversations with Knut Aasland (Norwegian Institute of Technology, University of Trondheim, Norway) regarding design theories, Oregon State University, 1992 through 1993.


Parametric Technology Corp, February, 1994, ProEngineer demonstration, shown at Oregon State University, Michael Lawrence.


Salomons, O., 1991, "Features, link between design and manufacturing?" Dept. Production engineering, University of Twente.


