GENERAL FEATURE-BASED FRAME REPRESENTATION FOR DESCRIBING MECHANICAL ENGINEERING DESIGN DEVELOPED FROM EMPIRICAL DATA

J. Tikerpuu, Graduate Research Assistant and D. G. Ullman, Associate Professor
Department of Mechanical Engineering
Oregon State University
Corvallis, Oregon

ABSTRACT

To establish the requirements for representing a mechanical engineering design process, practicing engineers were videotaped while performing a practical design. The process was followed from the early conceptual to the final detail design stage. Requirements for representing their progress include the need to describe the design in terms of the state of design objects and the changes that occurred to these objects. A design object's state is structured as a hierarchy of assemblies, subcomponents, and the interfaces between assemblies and subcomponents. These objects are described in terms of context sensitive form and functional features. The changes in a design object's state are described in terms of operations applied to the design object. Current computer aided design representations are primarily parametric object modelers. Very little attention has been given to describing the functions of objects or the interfaces between different objects. These current representations describe the design process as a very inflexible sequence of state changes. This paper presents a representation for a general description of a design object's form and functions based on the empirical data. The hierarchy with which the objects are structured allows for great flexibility in describing objects and their interfaces.

1.0 INTRODUCTION

This paper presents a representation of mechanical design objects based on empirical data. The empirical data was provided by a protocol study (Stauffer et al., 1987; Stauffer, 1987; Ullman et al., 1987; Ullman et al., 1988) aimed at identifying design methodologies. In this study, mechanical engineers where video and audio taped for approximately ten hours each, while performing a mechanical design. During the study, the designers were able to progress from the abstract conceptual to the concrete detail stages of the design process. As a result, a wealth of information was made available to develop a representation of mechanical design information.

The needs of a representation for mechanical design based on the protocol data are presented. To satisfy these needs, a model of a Mechanical Design Database is developed. Using this model as a foundation, a frame-based representation structure (Minsky, 1986) is presented for representing design objects and the changes encountered as they evolve from abstract concepts to concrete forms.

Design objects include assemblies, parts of assemblies, and interfaces between assemblies and parts. All of these objects are described in terms of context sensitive form and function features. The changes the design objects encounter are as a result of design *operators* that are applied to the existing design information within the Mechanical Design Database. The operator types were defined in the protocol study and reported in Stauffer (1987) and Ullman et al. (1988).

The Mechanical Design Database representation provides a flexible representation medium for describing forms and functions of objects. The representation of the changes the objects encounter provides a mechanism for capturing the relationship between constraints, design objects and the design process that describes the evolution of the objects from constraints.

2.0 EMPIRICAL DATA SOURCE

The designers that participated in the protocol study, as subjects, were asked to solve one of two design problems; the "flipper-dipper" problem or the battery contact problem. Although the representation was developed from the study of both problems, the sample representation used in this paper is from the battery contact problem. Thus, only this problem will be discussed. A more detailed description of both problems can be found in Stauffer (1987) and Ullman et al. (1987a and 1987b).

The battery contact problem used in the protocol study required the engineer to design the containment compartment and electrical connections for three small batteries to be used to power the clock/calendar of a portable computer. Detail dimensions of the batteries and battery envelop were provided along with the topological relationship between a PC board and the battery envelope. The batteries were to be connected in series. The entire assembly was to be robot assembled at a rate of 5000 a month for three years. Knowledge in basic D.C. electronics, metal springs, and plastic injection molding practices was required to solve this design problem. A complete specification list with drawings is available in Stauffer (1987) and Ullman et al. (1987a and 1987b).

The protocol method (Newell and Simon, 1972) was used to study design engineers because of the amount of detailed information that could be obtained about the design process. The audio tapes provided a record of the designer's verbalizations. These recordings were transcribed and used to gain an understanding of the flow of information used during the design. The transcripts also provided a written document with which the designers vocabulary could be analyzed. The video tapes provided a record of all of the hand gestures and drawing processes the designer exercised during the design process.

The protocol study describes the design process in terms of a goal hierarchy. The top level goal of this hierarchy is to define some type of form to satisfy a set of constraints. To accomplish this top goal, a set of subgoals is carried out. These subgoals are referred to as *tasks*. In an effort to carry out each *task* subgoal, a sequence of goal oriented episodes is executed. An *episode* may be completed through the implementation of *operators* that are applied to existing information in the Mechanical Design Database to create a change in the state of the information in the database. These database state changes occur until the goal of the episode is attained. The types of tasks, episodes, and operators found in the protocol studies are presented in Figure 1. An explanation of each task, episode and operator can be found in Stauffer (1987) and Ullman et al. (1988).

From the protocol data, eight needs of a Mechanical Design Database representation have been established. These needs can not be satisfied by either existing CAD tools or published representation (Dixon et al., 1985; Kinoglu and Donath, 1985; Lai and Wilson, 1987; and Takase and Nakajima, 1985). Consequently, a new representation is presented which can satisfy all of the needs for mechanical design.

TASKS:	Conceptual Layout Detail
EPISODES:	Assimilate Plan Specify Repair Verify Document
OPERATORS:	Generate Type *Select, Create* Evaluate Type *Simulate, Compare, Calculate* Decide Type *Accept, Reject, Suspend, Refine, Patch* Draw Type *Draw*

Fig. 1 Goals, episodes, and operators from protocol study

3.0 OVERVIEW OF THE MECHANICAL DESIGN DATABASE

The eight needs of a database representation based on the empirical protocol data are:

1. Provide a structure for describing design objects and constraints in terms of functions as well as forms.
2. Record changes that a design object experiences as it is refined throughout a design process.
3. Describe objects in engineering terminology to provide acommon vocabulary between the database representation and designer. This is also important for preservation of the ideas the designer incorporates into the design.
4. Describe design objects in such a way as to capture the hierarchical form relationships between assemblies and parts.
5. Record design strategies for designers to review and modify to better achieve the design goal.
6. Reference context sensitive information in domains as engineers do to provide the information a designer needs to accomplish a domain specific task.
7. Relate design objects to constraints to facilitate the need for checking for constraint violation and satisfaction.
8. Describe design objects in decomposable units to allow for object modification at all levels of the hierarchy described in Need #4.

A design database to facilitate these needs is presented in Figure 2. It can be divided into 2 subdatabases, the Procedural and Design State Subdatabases. The information in the Procedural Subdatabase is the knowledge the designer may use to create changes in the Design State Subdatabase. In other words, the Procedural Subdatabase contains the tools, in the form of information, needed to obtain the top level goal of the design process. The information that results from the application of these tools will be stored in the Design State Subdatabase. This paper focuses on the Design State Subdatabase. Details about the Procedural Subdatabase can be found in Tikerpuu (1988).

The Design State Subdatabase contains information which describes the *design state*. A *design state* is defined as a snapshot of the design process and all the design information that exists as part of the process at the time the snapshot is taken. The description of the design state will also include a record of all of the changes that have occurred to the design information up to the point when the design state is viewed. This subdatabase can be divided into two subsections. One that is related to the Object State Information and one related to the Changes Of State Information.

The Object State Information section contains form and functional descriptions of any physical objects being developed and/or manipulated during the design process. These categories include information describing *assemblies*, *parts* of assemblies, *interfaces* between objects, and *features* of assemblies, parts and interfaces. Since the representation of data in the Object State Information section of the Design State Subdatabase is so important, the next two paragraphs will be dedicated to describing the categories of this subdatabase section.

Empirical protocol data indicated that designers describe objects primarily in terms of assemblies and

MECHANICAL DESIGN DATABASE

- Procedural Subdatabase
- Design State Subdatabase
    - Object State Information
        - OBJECTS
            - *Assembly and Parts*
            - *Interfaces*
        - FEATURES
            - *Function Feature*
            - FORM FEATURES
                - *Manufacturing feature*
                - *Geometric feature*
                    - *GEOMETRIC SHAPES*
                        - *Disk shape*
                        - *Block shape*
    - *Change Of State Information*
        - *CONSTRAINTS*
            - *Function Constraints*
            - *Form Constraints*
        - PROPOSED SOLUTIONS
            - *Proposals*
        - OPERATORS
            - *Select*
            - *Create*
            - *Simulate*
            - *Calculate*
            - *Compare*
            - *Accept/Reject/Pending*
            - *Patch*
            - *Refine*
            - *Draw*

Each element in *italics* has a corresponding Frame representation.

Fig. 2 Mechanical Design Database and corresponding frames.

parts. An assembly can be decomposed into subassemblies and components (referred to as parts). A subassembly is just an assembly that is an integral part of a larger assembly. In general then, all the objects can be labeled *assemblies* or *parts*. The relationships between these assemblies and parts are described in terms of physical *interfaces* between objects. These assemblies, parts, and interfaces are described in terms of form and function features.

A feature is any descriptive attribute of an assembly, part or interface that cannot be included in a bill of materials. These features can be form or function oriented. For example, an open box can be defined as having 5 form features, 4 walls and the bottom. These, in turn, have features such as dimensions and geometric details. A functional feature of the bottom of the box is to prevent something from falling between the walls. Another functional feature is to hold the walls in a square configuration. In essence, a feature is the smallest grain size of information representable with the Mechanical Design Database representation while an assembly is the largest.

The changes that occur to the design objects, termed *delta states*, are included in the Change of State Information section of the Design State Subdatabase. As previously mentioned, these changes are described in terms of operators. Knowledge about how these operators can be applied exists in the operator knowledge category of the Procedural Subdatabase. The representation structures for the operators essentially note the information needed to apply the operator to the design state.

Other information in the Change of State Information section is data describing strategies, proposed solutions, and form and function constraints. Strategies will not be addressed in this paper but constraints and proposed solutions will be described in detail in section 4.2.3 and 4.2.4, respectively. Data about strategies can be found in Stauffer (1987) and Ullman et al. (1988).

4.0 DETAILS ABOUT THE DESIGN STATE SUBDATABASE

In the early stages of the design process, an abstract concept is incorporated into the design to satisfy primarily functional constraints. As the design progresses, the concept quickly evolves into an overall assembly which is the form equivalent to the concept. This overall assembly is then decomposed into subassemblies which themselves can be decomposed into parts and finally into features. A designer needs the flexibility to represent his design in a way that reflects the manner in which he decomposed it. This type of information is contained in the Design State Subdatabase. In the following sections, the representation of Object State Information will be presented before that of the Change Of State Information. This is done since the representation of the Change Of State Information references the information in the Object State Information section of the Design State Subdatabase.

Throughout the discussion of the Design State SubDatabase representation, references will be made to frame numbers. These frame numbers correspond to frames included in the sample representation presented in Section 4.2.6 and in Figures 3 and 4. The sample is that of an episode from the protocol study requiring several types of Design State Subdatabase representation structures to describe the information state of the design during that episode. The boldface characters within the sample frames represent frame attributes and the values associated with these attributes are placed in slots to the right of the attribute written in normal characters. Information is extracted from the frames by referencing the attributes and extracting the corresponding values. The values related to the attributes is provided by the designer.

4.1 Object State Information Representation Structure

As previously mentioned, the Object State Information section of the Design State Subdatabase contains all of the representation structures describing the state of design objects. Therefore, at any one time, a frame representation of every design object incorporated into the final design may be found in this section of the Design State Subdatabase. Even frame representation structures of objects not incorporated into the final design are maintained in the Object State Information section to preserve as much design information as possible. This is crucial in considering the need to represent information referenced by operator frame structures that eliminate those particular objects not incorporated into the final design.

4.1.1 Assemblies and Parts.

Considering assemblies as structures of subassemblies and parts, the substructures of the assembly work together to satisfy the functional constraints of the problem. Each part of the assembly also accomplishes unique subfunctions. The subfunctions of the parts work together to accomplish the function of the assembly.

When developing the frame representation for assemblies and parts, empirical protocol data pointed to the fact that there are many similarities in the way designers described these design objects. There were even several instances where a designer chose to describe an assembly as a part, as in the case when choosing an item from a catalog; for instance a motor assembly. The designer did not need to clutter his mind with the parts of the motor assembly therefore, out of convenience, the motor became a part in his mind.

The description similarities between assemblies and parts led to the creation of a frame capable of providing a representation for either object. Some of the slots to describe assemblies will be filled while those same slots will remain empty when describing parts.

An example of a frame structure representing an assembly can be found in the sample representation, Section 4.2.6, frame #1. This frame can be viewed as the central point which references all frames related to the assembly/part, in this case, the battery. Reference in this case means "points to." In other words, the assembly/part frame is a place to record pointers to other frames describing the assembly/part. In the frame, the **composed of** attribute is used to relate frames describing parts of the assembly. When the object frame describes a part, main features that are used to describe the part, such as the four walls of a box are referenced by the **composed of** attribute. This is because the walls may have features associated with it such as the corners of the wall or either side of the wall.

The **geometric feature** and **manufacture feature** attributes are used to reference the actual frames that describe the context sensitive form and function features. These two types of features are represented because the protocol samples required representation in contexts of geometry and manufacturing processes. Other contexts can be referenced by including additional attributes into the frame structure such as **kinematic feature**, **thermodynamics feature**, or **fluid flow feature**, to name a few.

Other representable types of information not obvious from the attribute names relate to the **origin** and state of the object. The **origin** attribute will be associated with the name of a specific designer, a catalog reference, or stock inventory number. The attribute is associated with a label "problem specific" when the object described by the frame is provided as part of the design specifications specific to the design problem. The **state** attribute can take a value of "active," "inactive," or "pending." These state values depend on whether the object has been incorporated into the final design, omitted from the final design, or might be included in the final design, respectively.

Another attribute, **generic**, is included in the object frame for reasons that are not obvious. In an effort to minimize the duplication of frames describing the same object, generic objects may be designated. The frames representing these generic objects are referenced from the generic attribute slot. For details on these frames see Tikerpuu (1988).

4.1.2 <u>Interfaces</u>. To capture form and function relationships between design objects, a frame structure referred to as an *interface* is introduced. Interfaces between design objects may occur in one of three configurations, between an assembly and a part, between two assemblies, or between two parts.

Frame #13 of the sample representation in Section 4.2.6, is an example of an *interface* frame. It describes the interface between the battery contact and battery as that of pressure contact since the battery contacts exert a force on the battery. From this frame, the geometric feature frame, **contact surface1 geometry** is referenced. Pointers to object frames describing parts of the battery and battery contact are recorded with the **composed of** attribute since the battery and battery contact are interfaced together. The **interface component** attribute is related to the component that is used to join the objects, pointed to by the **composed of** attribute, together. For example, a bolt may act as an interface component between two objects.

The **interface type** attribute is associated with the vocabulary used to describe the interface. An interface type may be described as having a certain number of degrees of freedom, specific clearances, tolerances, fastening characteristics, etc.

4.1.3 <u>Features</u>. Even though "*features*" is defined as an object category type in the <u>Object State Information</u> section in Figure 2, the concept of features is probably the most important aspect of the Mechanical Design Database representation. This is so because all objects (*assemblies*, *parts*, and *interfaces*) are described in terms of form and function features.

In the early stages of the design process, abstract functional data is primarily manipulated. As the abstract functional data evolves into concrete forms, form features are manipulated. Thus, it is important to provide a frame structure for both types of features. The representation for *function features* will be described first, followed by a discussion of *form features*.

From the empirical protocol data it was evident that designers describe objects in terms of *features* related to functions of the object. Although *function features* are related to domain dependent contexts, only one *function feature* frame is needed to describe functions in all contexts. The context is noted as a value related to the **context** attribute in the *function feature* frame.

A *function feature* is described in terms of vocabulary provided by the designer. The representation of function with syntax is not uncommon (Kinoglu et al., 1985; Lai and Wilson, 1987; and Takase and Nakajima, 1985). It is the most commonly exercised method designers use to describe a function of an object. As a result, the feature frame representing function has provisions for storing parts of a sentence the designer uses to describe an object's function. The correlation between a sentence structure and the function feature frame attributes that record selected parts of the sentence to capture the function of an object is shown below.

<u>SENTENCE STRUCTURE</u>:

Sentence	– subject + verb phrase
Verb phrase	– verb + noun phrase
Noun phrase	– determinant + noun + prepositional phrase

<u>RELATIONSHIPS WITH FRAME ATTRIBUTES</u>:

Attribute name :	Associated attribute value
Subject of function:	Subject
Function :	Verb

Object of function	:	Noun of noun phrase
Function modifier	:	Preposition phrase of Noun phrase

By no means does this sentence structure represent all possible function descriptions. It is, however, possible to represent all of the function descriptions from the empirical protocol data. Since an example of a *function feature* frame is not included in the sample representation in Section 4.2.6, one is presented below.

```
[(Name battery compartment function 1)
 (Is a feature)
 (Is type function)
 (Context geometric)
 (Origin catalog)
 (Reference drawing ())
 (State active)
 (Subject of function (battery envelope))
 (Function (enclose))
 (Object of function battery)
 (Function modifier    )
 (Function constraint    )]
```

This sample describes the object battery envelope as having the function of enclosing the battery. Since it is related to geometry it is a geometric context type.

A form type feature frame is used to describe geometric feature information about an *assembly*, *part* or *interface*. There are two types of form feature frames: *geometric feature* frames and *manufacture feature* frames. The *geometric feature* frames reference shape frames defining the dimensions of the geometric shape described by that particular *geometric feature* frame. Two example types of shape frames shown in Figure 2 are the *disk shape* and *block shape* frames. This feature frame arrangement is demonstrated in the sample representation by frames #14 and #15 in Section 4.2.6.

Frame #14 represents a *geometric feature* frame referenced from the object frame #13. Frame #15 represents the corresponding *disk shape* frame referenced from the geometric feature frame #14. This *disk shape* frame actually provides the geometric dimensions of the shape defining the feature. The need for the *geometric feature* frame becomes evident when considering the need to describe different objects with a similar geometric feature.

For example, the battery contact problem specifies three batteries to be included in the battery envelope. Three *geometric feature* frames are created with a different **location** attribute value for each battery. Each of these *geometric feature* frames references the same *disk shape* frame that describe the battery geometry. Even though the *disk shape* frames have **location** attributes in them, many duplicate *disk shape* frames would be required to describe the geometric location of the three batteries using the **location** attribute in this frame. Instead, the *geometric feature* frame describes the three different battery locations of the three different batteries with it's **location** attribute. The **location** attribute in the *disk shape* frame is not used in the sample representation but will be used in a multiple object representation case where a different location tolerance information is needed to be associated with the different locations represented in the *geometric feature* frames. The sample representation does not include either a *block shape* frame or a *manufacture feature* frame. A *block shape* frame has all of the same attributes as the *disk shape* frame except the dimension attributes are related to a block instead of a disk. It is evident that the *geometric feature* frames can point to much more complex representations such as solid models.

A *manufacture feature* frame describes a manufacturing process associated with a part or assembly. Information unique to this frame includes the attributes about the **process rate**, the **process sequence** in terms of part assembly sequence, **process lifetime** and **process type**. A process type can be designated as "injection molding", "robotic assembly," or any other type of process described by designers.

4.2 Change of Object State Information

Along with the Object State Information section of the Design State Subdatabase, exists the Change of State Information section. This section contains the operator representations and the operands that are their focus. These operands include constraints and proposed solutions (proposals). Constraints are represented by *function constraint* and/or *form constraint* frames. A proposed solution is described by a *proposal* frame. These frames will be discussed in the following paragraphs.

4.2.1 Change of State Representation. In general, an *operator* is applied to constraints and proposals. A proposal is described in terms of *assemblies*, *parts*, *interfaces*, and *features* and the constraints affecting these objects and features. A constraint may be linked to design objects by associating the constraint with a label represented somewhere in the design object representation. In essence, all information about design objects is referenced through constraints and proposals that are manipulated or created by the application of an operator. A sample of a protocol section complete with the frame representation of *operators*, *constraints*, *proposals*, as well as *assemblies*, *parts*, and *features* will be presented.

4.2.2 Constraints. Throughout the entire design process, a designer formulates ideas to satisfy a given set of form and function *constraints*. As these constraints guide the designer in developing and incorporating abstract concepts into the Design State Subdatabase, these abstract concepts act as *constraints* for the delta states yet to occur. As these abstract concepts evolve into concrete shapes, the concrete shapes act as constraints for other concrete shape definitions.

During this entire process, the designer must make decisions whether to keep established constraints that he derived and included into the design. He also notes that some constraints may not be relaxable from the Design State Subdatabase such as the constraints provided to him in the initial problem statement. Decisions about constraints can be associated with a type of priority system to indicate the ability to relax, or change constraints. Based on the empirical protocol data, four categories of constraints are needed to indicate a priority system. These four categories also indicate the origin of the constraints.

4.2.3 Constraint Representation. Since constraints may be of form or function type, each requires a separate representation. The frame representation for a *function constraint* is almost exactly the same as that for *function feature*. This is somewhat expected since a *function feature* also acts as a constraint on the design process. It is also expected since empirical protocol data supports the fact that both *function constraints* and *function features* are described in vocabulary that correspond to parts of a sentence structure.

Constraints within the Mechanical Design Database are represented in two ways, explicitly and implicitly. They are labeled so because of the location they occupy within the design database. Explicit constraints are those that are represented in *constraint* frames. Implicit constraints are represented by the object descriptions in the Object State Information section of Figure 2. Therefore, the representation reasoning facility may infer constraint information from either source. To access constraints within a design object description, information representing objects are recorded with a label similar to a variable associated with a numeric value. This label may be referenced as a variable if a number is not associated with it or a value associated with this label can be checked for specification constraint violation.

Frames #4, #6, and #8 in Section 4.2.6, all are examples of *form constraints*. The form constraint values are associated with a label which may also be contained in the object description frames. In this way a correlation is made between the constraint and the actual object the constraint affects.

4.2.4 Proposals. Proposals are possible solutions for satisfying a set of given constraints. A proposal may be described in term of any of the categories of information described in the Object State Information section of the Design State Subdatabase such as *assembly*, *part*, *interface*, *form feature* or *function feature*. A mechanism's description is contained in the sum of all the proposals that are accepted, by the accept operator, in the design process related to that mechanism. As the design process progresses, proposals are refined into new proposals that satisfy or comes closer to satisfying active *constraints*.

Frames #3 and #11 represent proposals frames. Frame #3 represents the proposal to dimension the battery contact surface and label the surface dimension as "contact surface1." Frame #11 represents the proposal that results from the refinement of the proposal represented in frame #3.

Other information in this proposal frame (frame #11) references the constraint frames (sample representation frames #4, #6, and #8) that provided information for the refinement (CONS3, CONS4, CONS5, respectively).

4.2.5 Operators. The *operator* representation structure is basically a bookkeeping data structure. The *operator* frames are used to note what operator was executed during the sequence of operations in the design process. The **task** and **episode** type is noted for each operator as well as the constraints and/or proposals affected by the operator. In the sample representation, frames #5, #7, #9, #10 and #12 represent operators. A complete operator representation is available in Tikerpuu (1988)

The frames representing the *constraints* and *proposals* are referenced from each *operator* frame by an attribute unique to each. In frame #5, a *select* operator, the constraint specified is referenced by the **entity selected** attribute. In the *create* operator frame structure, frame #7, the **entity created** serves the same purpose. Frame #10 describes a *refine* operator. This operator representation is needed to record what was refined and the result. The **entity refined** and **entity refined into** attributes are used to reference this information. In Frame #12, an *accept* operator, what is accepted is referenced from the **decided on** attribute. How these operators are used in the representation is covered in the next section.

4.2.6 Sample Representation Of Protocol Episode. The following sample representation is taken from the battery contact protocol of one subject, approximately 56 minutes into the design process. The task at hand is the *layout design* of a battery contact that will connect two batteries in series. This sample representation is part of a single episode in this *task*. The goal of this episode is to *specify* the dimensions of the contact surface area of the battery contact. Five operators are used during this part of the episode. Before the operator sequence is described, some Object State Information data must be described which is used to accomplish the episode.

The frame representation structure for the protocol sample is provided below and in Figures 3 and 4. In Figure 3 the flow of the episode is shown, operator by operator. Also shown are the pointers from the operators to the proposals and constraints. In Figure 4 the representation of the initial state at the beginning of the episode is shown, along with the representation of the final state at the end. Also, the pointers between frames are shown.

The first frame represents the main object descriptor frame for the battery. Several other frames referenced from the **composed of**, and **geometric feature** slots of the battery object frame completely describe the battery object in detail. The frame after the battery object frame is the battery contact object frame. This describes the battery contact object in the same way the battery object is described. Frames #3 and #4 are the proposal and constraint frames used during the actual episode. They are the information from the Object State Database that is needed to carry out the operation designated by the various operators within the episode. The sample follows with interleaved discussion of the operator frames.

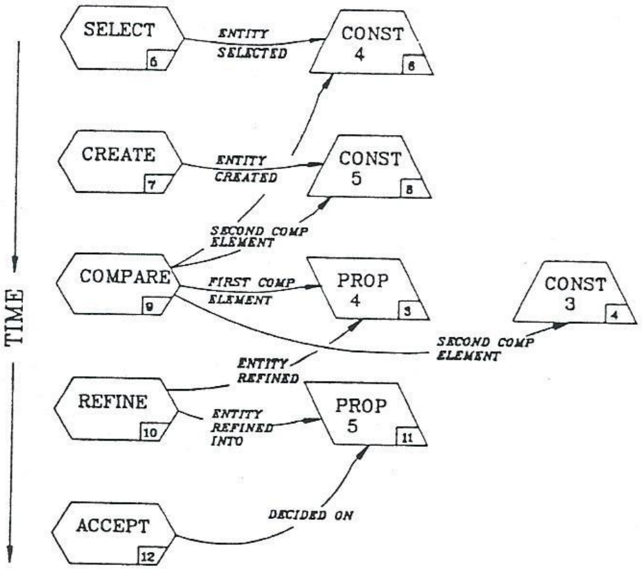

Fig. 3 Delta state frame representation

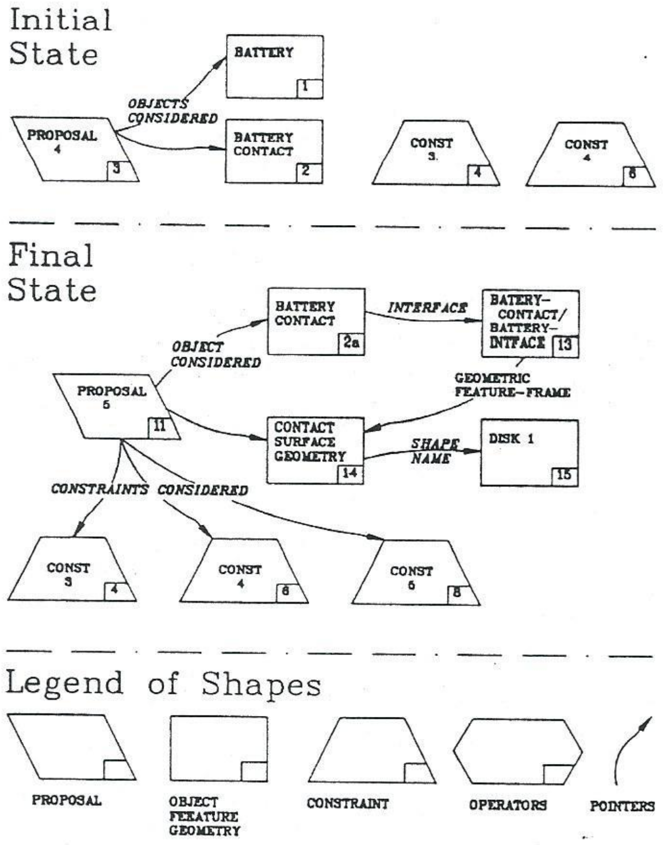

Fig. 4 Object state frame representation

```
1 [(Name Battery)
   (Is a assembly)
   (Type of battery)
   (Quantity 3)
   (Origin catalog)
   (Referenced drawings (...))
   (State active)
   (Generic (generic battery))
   (Reference axis (x,y,z,thetax,thetay,thetaz))
   (Composed of (battery top, battery bottom))
      (Geometric feature (battery anode geometry,
                          battery cathode geometry))
      (Manufacture feature  )
      (Material   )
      (Interfaces (battery to battery envelope
                   bottom))]
```

```
2 [(Name battery contact)
   (Is a part)
   (Quantity 1)
   (Origin subject-s1)
   (Drawing referenced (6b))
   (State active)
   (Generic () )
   (Reference axis (0,0,0,0,0,0))
   (Composed of )
      (Geometric feature contact geometry )
      (Manufacture feature  )
   (Material beryllium copper)
   (Interfaces ())]
```

```
3 [(Name PROP4)
   (Is a proposal)
   (Reference drawing (6b,7a))
   (Description
      (diameter variable contact surfacel for
       battery contact surface from drawing))
   (Constraints considered ())
   (Objects considered (battery contact, battery))
   (Featured considered )
   (Objects created   )
   (Features created )
   (Data label affected contact surfacel)
   (Episodes involved (EPS3))
   (State active)]
```

```
4 [(Name CONS3)
   (Is a constraint)
   (Is type form)
   (Origin Independent derived)
   (Parent constraint )
   (Reference drawing (specl))
   (State active)
   (Constraint label contact surface)
   (Constraint value <.28)]
```

The next several frames represent the *operators*, *constraints* and *proposals* that make up the sample episode. The first operator to be used is the *select* operator, frame #5. A selection of a constraint (CONS4, frame #6) to round off a number is made. The next operator is a *create* operator, frame #7, where a *constraint* (CONS5, frame #8) stating that the clearance between the contact diameter and battery diameter must be 0.040. A *compare* operator, frame #9, is used comparing a proposal (PROP4, frame #3) to three constraints. The first constraint (CONS3, frame #4) indicates the diameter of the contacting surface must be less than .28". The second *constraint* is CONS4 (frame #6), and the third is CONS5 (frame #8). After the comparison a *refine* operator, frame #10, is used to refine PROP4 into a new *proposal* (PROP5, frame #11). This new *proposal* (PROP5) is to make the diameter of the contacting surface equal to 0.2". This *proposal* (PROP5) is incorporated into the final design with an *accept* operator, frame #12.

```
5 [(Name select process P3)
   (Is a operator)
   (Task layout component)
   (Episode type specify)
   (Episode designator EPS3)
   (Operator select)
   (Subepisode    )
   (Notes round off value for ease of numbers)
      (entity selected CONS4)]
```

```
6 [(Name CONS4)
   (Is a constraint)
   (Is type form)
   (Origin independent derived)
   (Parent constraint )
   (Reference drawing (specl))
   (State active)
   (Constraint label contact surface)
   (Constraint value [= 100*integer(100*contact
                        surface)])]
```

```
7 [(Name create process P4)
   (Is a operator)
   (Task layout component)
   (Episode EPS3)
   (Operator create)
```

```
    (Subepisode of  )
    (Notes  establish geometric clearance constraint)
       (Entity created CONS5)]

8  [(Name CONS5)
    (Is a constraint)
    (Is type form)
    (Origin independent derived)
    (Parent constraint )
    (Reference drawing ())
    (State active)]
       (Constraint label (battery diameter wall
                               clearance))
       (Constraint value =.040)]

9  [(Name compare process P5)
    (Is a operator)
    (Task layout component)
    (Episode type specify)
    (Episode designator EPS3)
    (Operator compare)
    (Subepisode of   )
       (First comparison element (dimension contact
                                    surface PROP4))
       (Second comparison element (CONS3, CONS4,
                                    CONS5))]

10  [(Name refine process P6)
     (Is a operator)
     (Task layout component)
     (Episode type specify)
     (Episode designator EPS3)
     (Operator refine)
     (Subepisode of )
        (Entity refined PROP4)
        (Entity refined into PROP5)]

11  [(Name PROP5)
     (Is a proposal)
     (Description
      (define surface area of battery contact/battery
       interface surface))
     (Constraints considered (CONS3,CONS4,CONS5))
     (Objects considered (battery contact, battery))
     (Feature considered (contact surface1 geometry))
     (Data label affected contact surface1)
     (Reference drawing (7a ))
     (Episodes (EPS3,EPS4))
     (State active)]

12  [(Name accept process P7)
     (Is a operator)
     (Task layout component)
     (Episode type specify)
     (Episode designator EPS4)
     (Subepisode of )
     (Operator accept)
        (Decided on PROP5)]
```

After the new proposal, PROP5, is accepted, the <u>Object State Database</u> changes to reflect the effect of the episode. The battery contact component frame (frame #3) will change to reflect the fact that there is a new *interface* labeled "battery contact/battery interface" (frame #2A). The following frame #13 represents the interface. Frame #14 is the *form feature* frame which references the actual *geometry frame*, frame #15, representing the fact that the radius of the feature has been established at 0.1".

```
2A  [(Name battery contact)
     (Is a component)
     (Quantity 1)
     (Origin subject-s1)
     (Drawing referenced (6b))
     (State active)
     (Generic () )
     (Reference axis ())
     (Composed of ()
        (Geometric feature  contact geometry)
        (Manufacture feature  )
     (Material beryllium copper)
     (Interfaces (battery contact/battery interface))]

13  [(Name battery contact/battery interface)
     (Is a interface)
     (Interface type pressure contact)
     (Interface component ()
     (State active)
     (Composed of (battery, battery contact))
        (Geometric feature (contact surface1 geometry)
        (Manufacture feature
          contact surface1 manufacture process1)]

14  [(Name contact surface1 geometry)
     (Is a feature)
     (Is type form)
     (State active)
     (Origin  subject s1)
     (Context geometric)
     (Reference drawing spec2)
     (Reference axes )
     (Binary status on)
     (Part of object  battery contact)
     (Part of feature )
     (Shape type disk)
     (Shape name disk1)
     (Reference axes (0,0,0,0,0,0))
     (Location )]

15  [(Name  disk1)
     (Is a shape)
     (Is type disk)
     (Part of component battery contact)
     (Part of feature contact surface1 geometry)
        (x location value .245)
        (x location label x surface area center)
        (x location tolerance .004)
        (y location )
        (y location label )
        (y location tolerance )
        (z location .285
        (z location label z surface area center)
        (z location tolerance .0015)
        (Radius value  0.1)
        (Radius label contact surface1)
        (Radius tolerance )
        (Start angle     )
        (End angle       )]
```

This concludes the sample from the battery contact protocol. Information regarding all of the attributes and their possible values can be found in Tikerpuu (1988).

5.0 CONCLUSIONS

In the beginning of this paper, a set of needs for a Mechanical Design Database was established. Keeping these needs in mind, empirical data obtained from protocol analysis was used to identify what types of information engineers use during a typical design process. This information is used to describe design objects and the delta states that they encounter as the progress from specifications to the final design.

From the empirical protocol data, a model of a Mechanical Design Database was established. Of the two subdatabases, <u>Procedural Subdatabase</u> and <u>Design State Subdatabase</u>, this paper focused on the representation of information in the <u>Design State Subdatabase</u>. Objects were represented as *assemblies*, *parts*, and *interfaces*, all of which are described in terms of form and function oriented *features*. For each of these objects, interfaces and features, a data structure was developed to describe the objects and their relationships. A data structure to represent constraints, proposed solutions and design object delta states was developed.

Taking a computer science viewpoint, a computer representation structure, Frames, was chosen for representing the protocol design data in a computer program. The capability to represent form and function constraints was tested by representing the given constraints for the battery contact problem. A protocol sections from the battery contact problem was also represented to further refine the delta state representation.

From representing the data of the protocol sample, it has been concluded that the proposed representation structure provides a flexible data structure for representing form and functional descriptions of objects and their relationships. It is felt that it would be possible to make inferences about an object's form and function during all stages of the design process.

The Mechanical Design Database representation efficiently relates design objects, constraints, proposals, and delta states. By representing all of these types of design information relative to each other, it is possible to get a snapshot of the design process to determine to what level of abstraction design objects have been developed. It is also possible to see what design objects have been developed but not incorporated into the final design process for some reason or another. These two qualities of the representation may be capitalized upon by some type of design tool that is able to understand the history of the design process.

By describing objects in terms of context sensitive form and function features, a very well organized computer representation is created. Information organized by contexts provides designers with information indexed by domains that designers are familiar with. This organization also promotes efficient data manipulation due to the narrowing of information requested when a designer may request context sensitive data.

The ability of the Mechanical Design Database representation to completely and efficiently store design information will be tested in a database query program currently being developed. Typical questions, provided by the protocol data, the designers had about the given constraints will be posed to the query program.

In conclusion, the Mechanical Design Database representation structure of information in the Design State Database has the potential for reasoning about form and function features of design objects. The Mechanical Design Database representation of constraints, proposals, and delta states provide enough information to understand what design objects satisfy what design constraints, and what possible solutions the designer is presently pursuing.

6.0 BIBLIOGRAPHY

Dixon, J.R., Libardi, Jr., E.C., Luby, S.C., Vaghul, M., and Simmons, M.S., 1985, "Expert Systems for Mechanical Design: Examples of Symbolic Representation of Design Geometries," *Applications of Knowledge-Based Systems to Engineering Analysis and Design*, a publication of the American Society of Mechanical Engineers.

Kinoglu, F., Riley, D., and Donath, M., 1985, "Knowledge-Based System Model of the Design Process," *Proceedings of the 1985 ASME International Computers in Engineering Conference and Exhibition*, Boston, MA.

Lai, K. and Wilson, W.R.D., 1987, "FDL - A Language for Function Description and Rationalization in Mechanical Design, " *International Computers in Engineering Conference and Exhibition*, New York, NY.

Minsky, M., 1986, "A Framework for Representing Knowledge," In *Reading in Knowledge Representation*.

Newell, A. and Simon, H.A. 1972, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ.

Rich, E., 1983, *Artificial Intelligence*, McGraw Hill Series in Artificial Intelligence.

Stauffer, L.A., 1987, "An Empirical Study on the Process of Mechanical Design," Ph.D. Thesis, Department of Mechanical Engineering, Oregon State University, Corvallis, OR.

Stauffer, L.A., Ullman, D.G., and Dietterich, T.G., 1987, "Protocol Analysis of Mechanical Engineering Design", International Conference on Engineering Design, Boston, MA.

Takase, H. and Nakajima, N., 1985, "A Language for Describing Assembled Machines," *Design and Synthesis*, Elsevier Science Publishers B.V. (North-Holland).

Tikerpuu, J., 1988, "Data Representations for Mechanical Design Based on Empirical Data," Masters Thesis, Department of Mechanical Engineering, Oregon State University, Corvallis, OR.

Ullman, D.G., Stauffer, L.A., Dietterich, T.G., 1987a, "Preliminary Results of an Experimental Study of the Mechanical Design Process," NSF Workshop on Design Theory and Methodology, Oakland, CA.

Ullman, D.G., Stauffer, L.A., Dietterich, T.G., 1987b, "Towards Expert CAD," *ASME Computers in Mechanical Engineering*, Vol. 6, No. 3, Nov-Dec, pp 21-29.

Ullman, D.G., Stauffer, L.A., Dietterich, T.G., 1988, "A Model of the Mechanical Design Process Based on Empirical Data," submitted to *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* and in summary form to the 1988 AI in Engineering Conference.

COMPUTERS IN ENGINEERING 1988

Proceedings of the 1988 ASME International
Computers in Engineering Conference and Exhibition
July 31–August 4, 1988
San Francisco, California

sponsored by

Computers in Engineering Division, ASME

Editors

V. A. Tipnis
E. M. Patton

Associate Editors

D. W. Bennett
A. A. Busnaina
G. L. Kinzel
M. F. Kinoglu
D. R. Riley
K. K. Tamma

VOLUME ONE

- Expert Systems
- Artificial Intelligence
- Knowledge-Based Systems

THE AMERICAN SOCIETY OF MECHANICAL ENGINEERS
United Engineering Center 345 East 47th Street New York, N.Y. 10017