Geometric Exemplars

A Bridge between AI and CAD

Bernie Bettig, Joshua D. Summers, and Jami J. Shah

Design Automation Laboratory, Arizona State University, Tempe, AZ 85287, USA

Abstract: In embodiment and detailed design, one is often concerned with sizing and geometric arrangement, i.e. determining the values of key parameters, dimensions, and locations of design elements. Feasible designs need to satisfy certain relationships between these parameters and entities. “Design Exemplars” are introduced as a concept for describing complex situational patterns and extracting information of interest from enhanced CAD models. Such a canonical representation of parametric problems is needed for bridging the gap between CAD and geometric reasoning used in AI. This paper will show that there are a large number of generic parametric and geometric problems that transcend domain boundaries. In design and manufacturing automation systems, these problems have typically been addressed in domain specific terms; lacking on the one hand, a uniform characterization, and on the other, a unified solution methodology.

Key words: Embodiment Design, Case Based Reasoning, Exemplars

1. INTRODUCTION

In embodiment design and detailed design, one is often concerned with parametric and geometric design. Parametric design is the determining the numerical values of the attributes identified at configuration stage. Parameters may be geometric (dimensions, volume, weight) or non-geometric (horsepower, material density, strength). Feasible designs need to satisfy certain relationships between essential parameters. In mechanical design, topology and geometry are often important factors; form follows function and functions are embodied in form. In every phase and aspect of design and manufacturing, geometric and topological parameters are usually

present in exploring options, reasoning about the performance, or specifying the solution. It is, therefore, important to recognise geometric problems as a special subset of parametric design concerned with sizing of dimension and location parameters, shape parameters, and geometric arrangement.

Traditionally, parametric design problems have been formulated as sets of equations that may be solved symbolically or numerically to yield satisfying or feasible designs with respect to the explicitly specified parametric relations. Parametric design tasks involve many types of problem solving tasks: initial design requires constraint solving, subsequent iterations require constraint propagation, evaluation requires pattern matching. All of these may be represented as generic mathematical problems.

A wide range of tools is available for algebraic equations solving, optimization, symbolic math, kinematic analysis, parametric CAD, and geometric constraint solving. Individually, these packages are not sufficient for design. There is a mismatch in problem abstraction and vocabulary; none encompass the entire range of problem types. Further, there is no framework for integrating different solvers, making it difficult to mix and match different approaches to suit different types of problems. Another complicating factor is our inability sometimes to recognize and extract generic geometric problems from domain specific ones.

Our research is based on the premise that there exists a common core set of parametric, topological and geometric problems across a broad range of design and manufacturing applications. In some areas, problems appear directly and explicitly as such, while in others some detachment and re-formulation is necessary. The question remains as to how to leverage this repetition of simple design tasks in routine embodiment design problems.

2. LITERATURE REVIEW

2.1 Parametric Design

A parametric problem can be described in terms of a list of variable entities, parameters describing the entities, relationships between the entities and/or parameters, and the allowable range for each parameter. Such a formulation is *declarative* because the solution procedure is not specified, explicitly or implicitly. Several types of analysis or design tasks are undertaken: initial design, redesign, change propagation, validation, etc., and these must be included in the problem formulation. However, such a

formulation does not incorporate any decision criterion, its objective is to simply produce the data needed for decision making.

Serrano and Gossard (1987) propose a declarative representation for parametric design in the form of a bipartite graph. The nodes of the graph are either design variables or algebraic equations. Rockwell's Design Sheet is also based on bipartite graphs, but uses more robust decomposition and solution procedures to be able to handle very large sets of equations (Reddy, et al., 1996). These formulations seek only satisficing solutions though the latter has incorporated some rather limited optimization capability.

Dixon's DOMINIC system is a well-known example of domain independent formulation of embodiment design problems (Dixon, 1995). DOMINIC, however, goes beyond analysis; it has incorporated measures of goodness so the system seeks to improve a given initial design by changing one parameter at a time. Parameter selection and step size are governed by heuristics embedded in the system. Other similar systems include DPMED and VEXED (Ramachandran, et al., 1992).

Ward and Seering (1989) discussed the need for a "Mechanical Design Compiler" with knowledge of mechanical components and capable of reasoning about mechanical systems to reduce the programming effort in developing design automation systems. Our research has a similar objective: the development of a design representation and language to reduce the development effort of automation systems. We prefer to do this via the more general "exemplars" for describing design conditions rather than the characterization of specific mechanical components.

2.2 Geometric Design

The systems cited in the previous section treat all parameters and relations as algebraic and/or logical. Geometric entities and relations are not considered separately and, consequently, the special properties of geometric problems are not exploited. In parametric CAD and geometric constraint solving, representations have been created to consider such properties. Two types of problems are solved in most parametric CAD systems: parametric sketching and assembly arrangement. In sketching, one can define a 2D topology and a set of geometric relations from which the geometry is determined. In assembly design, the relative location of components is derived from specified mating condition. As well, some systems now allow specifying geometric relations between topological entities in 3D. Each of these problems involves satisfaction of a set of geometric constraints.

Several researchers have proposed standard sets of relationships between geometric entities. Kramer (1992) proposed "markers" (a point and 2 orthogonal axes) as primitives that could be fixed to bodies in order to relate

them to each other by means of 6 basic binary or ternary constraints. This approach is particularly suited to the representation of kinematic linkages. Shah and Rogers (1993) proposed five basic structures for declarative modelling of parts, features, and assemblies. Clement, et al., (1991) has formalized geometric relations between six basic types of surfaces for use in tolerance analysis.

The ISO/STEP community is developing a model for parametric CAD for exchanging data between commercial systems. A procedural model (Pierra, et al., 1996) and a more comprehensive model, ENGEN (SCRA, 1997), have been proposed toward this goal. The latter defines not only geometric constraints but also algebraic equations and construction history. However, 3D geometric constraints are not included. Bettig (1999) presents models that include 3D geometric constraints. None of the above approaches address topological constraints, a formalization of the problem goal, or methods for transformations between design problems.

2.3 Case Based Reasoning

One observation of engineering designers is that they generally tend to get better with experience (Maher, et al., 1995). Designers gain experience about what solutions are effective and about which scenarios satisfy different problems. When a design problem is provided to an experienced designer, they tend to first try to find a similar past design. This proves beneficial for two reasons: redesign generally is less time consuming than novel design and the performance of the base design is known. Case based reasoning is based upon this premise of learning from past design (Wang and Howard, 1991; Hua, et al., 1992; Goel and Chandrasekaran, 1989; Sycara and Navinchandra, 1989). The case based approach to design is to find past designs that satisfy or nearly satisfy the problem definition. Based upon these cases, a new design is developed that specifically satisfies the design problem. Case based systems have three basic components: a case base, search engine, and adaptation system.

Some issues arise in the development of case based system. The main issues for creating a case base include deciding what is to be stored as a case, how the information in a case is internally organized, how the cases are to be cataloged in the case base, and how are cases maintained and updated. Indexing has been addressed by generating a design signature (Elinson, et al., 1997), memory based automated indexing (Lee, et al., 1997), an object oriented approach (Perera and Watson, 1996), or feature based (Krampe and Lusti, 1997). Finally, the adaptation issues include how will mismatches be removed and how will the design be completed to satisfy the design problem.

3. REPRESENTATION

Development of a representation for parametric design problems is the first step in developing parametric design systems. The representation must necessarily rest on a taxonomy, or categorization, of problem types. The characterization sought should: allow representation of mechanical design and manufacturing problems related to the specifying, manipulating, and querying of design models and their properties, must consider appropriate levels of abstraction with mapping between them, allow matching of solution techniques applicable for each category, and be concise and unambiguous with minimal sub-divisions and clear distinctions.

We characterized tasks by identifying the “given” (input) and “find” (output) parts of the problems and then determining a sequence of sub-problems as a “solution” procedure. The sub-problem attributes were generalized and the generic sub-problems considered as “building blocks” in describing other application problems. The attributes we considered were, the types of used entities, types of relationships between these entities, and the types of tasks that are performed. These entities, relationships and tasks were then categorized by their level of abstraction into four levels: (i) the numeric level, including vector algebra (parametric design), (ii) the geometric level, made up of points, directions and different types of curves and surfaces, as well as transformations of these (geometric design), (iii) the topological level, which is based on the idea of bounded geometry (geometric design), and (iv) the semantic level in which entities have meaning to the user but not mathematically (mechanical design). The resulting representation was made declarative to make it independent of any specific implementation method. Each stage of the embodiment design problem is described by the state of the design and the computational task to be performed at that stage. Design models in this study consist of entities and relations or constraints.

Representing designs using the above entities and constraints is done conveniently using a bipartite graph, with entities comprising one set of nodes and constraints the other. Arcs of the graph connect entity nodes to the constraint nodes by which they are constrained. The graph is drawn using the conventions shown in Figure 1. To indicate the type of entity or constraint a descriptive icon or text label is associated with the nodes.

Figure 1. Conventions for Graph Representations

Using this representation, it is possible to combine parametric, geometric, topological and semantic information. For example, the parameterized part and its representation shown in Figure 2.

Figure 2. Parameterized Part

A Design Task (S) operates on the Design Model (D) to either advance or evaluate some aspect of the design. Tasks are divided here into basic level and composite tasks. Six types of generic tasks were identified as shown in Figure 3.

Figure 3. Inputs and Outputs for Low Level Geometric Functions

From a number of case studies of design and manufacturing tasks we extracted application specific tasks (Bettig, 1999). We see that the reasoning required in the pursuit of such application tasks often hinges on describing a situation or circumstance, drawing inferences from the situation, and the execution of some actions based on the situation and inferences. In order to describe such situations and actions in domain independent terms, we introduce the concept of exemplars.

4. EXEMPLARS

In order to provide a useful mechanism, or tool, for a designer in manipulating the proposed representation of design models, we have proposed the “design exemplar”. Exemplars derive from the following observation: all queries about parametric relations or geometry, which arise during the design process deal with some characterization of the design. In design evaluation, we try to determine if the characterization is true. In initial design or redesign we try to make the characterization conditions true by modifying the design, while in design re-use we try to determine whether alternative designs are similar in some aspect. Exemplars allow a formal representation of such problems. Both exemplars and features describe some part or aspect of design in terms of parametric, topological, and geometrical relationships. However, exemplars provide a richer characterization than features and can themselves represent features.

We define an exemplar as a pattern of topologic, geometric, and algebraic relationships with high level engineering significance. Part of the pattern of the exemplar corresponds to actual geometry in the model (matched portion) and part of the pattern is inferred from the model (extracted portion) in order to perform reasoning about the matched geometry. In addition, an exemplar includes an optional second pattern to describe the false condition, when the characterization is not true. This enables the possibility of using exemplars in re-write rules, to change the design from the true condition to the false condition or vice versa. The concept of an exemplar is graphically illustrated in Figure 4.

Figure 4. The Components of an Exemplar

Most of the design tasks fall into four generic types: (I) identifying a region of design having a certain characteristic, (II) comparing two designs in terms of some characteristic, (III) finding some property of a design, and (IV) modifying a design so that it has some characteristic.

Type I task, identifying a region of design having some characteristic, is common in many DFX applications. One example of a Type I task in DFX is searching for regions of a part which cannot be die cast with a given parting plane and draft angle. The search involves finding surfaces whose normal deviates from the parting plane normal by more than 90 degrees less

the draft angle. The criteria may therefore be described and characterised in terms of an *exemplar* shown in Figure 5.

Figure 5. Characterization of DFM Rule

High level Task II involves comparing two designs according to some characteristic. It takes two design models as input, as well as an *exemplar*. The generic mapping of Task II to low level tasks is also provided in Bettig (1999).

High level Task III finds a property or design attribute. It takes a design model and an *exemplar* as input. This task is demonstrated with the help of an example: suppose we need to find the length of a belt or chain in a pulley setup. The belt can be characterized as a closed loop of repeating straight-line segments and circular arcs, with each being tangent to the next at intermediate boundary points. In order to find the total length of the belt it is necessary to find and sum the lengths of the individual segments. This is handled by the exemplar in Figure 6.

Figure 6. Four Pulley Graph Model

Finally, let us consider Task IV, the modify design task, in an application domain. The proposed approach is to take as input the design model in state

1 and an *exemplar* with a *true* and *false* condition and produce the output of the design model in state 2. As an example, we consider the collapsing of a machining volume in two feed directions in order to extract a tool-workpiece interference profile so as to match the volume to feasible machining operations (Shirur, 1994). Following the collapse of the volume into a profile, the profile and feed vectors can be compared available machining processes. Figure 7 shows the *exemplar* with both *true* (“before”) and *false* (“after”) conditions. The idea of this *exemplar* is that a group of entities at one end of the volume are separated from the group of entities at the other end by surfaces and straight edges of equal length. These lines and faces need to be found and eliminated to accomplish the collapse, as performed by the exemplar.

Figure 7. Example for Collapsing

5. SYSTEM ARCHITECTURE

It is clear that exemplars may be useful in developing parametric and geometric design automation tools useful in routine design. In order to develop exemplars into a useful design paradigm, it is necessary to develop an exemplar modeller. This modeller allows the user to specify the entities and constraints of the exemplar. In complicated situations, the number of entities and constraints, their possible internal patterns and relations, and the multiple special functionalities of exemplars may become too complex for a designer to easily manage. For this reason, a second layer of abstraction is provided to allow designers to create exemplar networks by combining established exemplars into complex exemplars.

The current vision of using *exemplars* in engineering embodiment design is focussed at three different levels for different types of end-users. The lowest level provides users the ability to create *exemplars* from the basic entities and constraints. The second level allows designers to create *exemplar* networks for complete design problems. Finally, at the highest level, the design engineers will be able to select an *exemplar* network for the specific design problem at hand, such as belt-design, cooling fin design, or castability analysis. Table 1 illustrates the different users and capabilities of a design by exemplar system at the different levels of abstraction.

Table 1. Levels of Design by Exemplar

Level	Capability	User
I	Model exemplars with entities and constraints	Exemplar Modeller
II	Model design problems by networking exemplars together	Network Modeller
III	Apply networks to solve engineering design problems	Design Engineer

At the first level, simple engineering tasks may be modelled, such as feature recognition, parameter extraction, or geometric modification. The first level may be thought of as the stage at which the tools are created. Just as feature based systems generally provide for the ability to create new features, an *exemplar* based system, ought to provide some capability to develop new *exemplars*. Additionally, the *exemplars* may be merged with other *exemplars* to satisfy new design tasks (Bettig, 1999). The newly created *exemplars* may be filed in a searchable design catalog for future use at this level of *exemplar* design or higher levels. Finally, the *exemplar* modeller provides designers the ability to model the *exemplar* as a “black box” and specify the inputs and outputs. The second level is designed for those wishing to construct a network of *exemplars* to solve routine complex design problems. These *exemplar* networks may be used as a complete design method, providing designers at the next level with a clean interface for engineering design. At the highest level, the design engineer need only to select the correct network for their specific design problem and provide appropriate fixed values for the specific problem.

In order to exploit the tasks performed by exemplars, query, modify, and validate, a comparison is drawn with tasks of case based reasoning (search, adapt, and evaluate). The parallel is striking. For this reason, a CBR system, based upon the exemplar, is proposed to facilitate routine embodiment design (Figure 8).

Figure 8. Case Based Exemplar System

The cases to be stored in the case base are proposed to be the design models of the final solutions represented in the entity/constraint graph form. Additionally, any exemplars that were used in the creation of these design solutions need to be enumerated for the new user. This will provide new user with some general design intent of the original designer. Further, these exemplars may be used as a base line in creating an adaptation exemplar network. The case base manager manages the case base and serves as the interface of the other modules into the case base.

The retrieval module consists of two main modules: the search engine and the case selector. The search engine interfaces with the design problem and the case base index to find all relevant cases. These cases are returned through the case base manager to the case selector. Decisions about which subset of the returned cases are to be passed to the adaptation module are handled by the case selector. The selector uses the information found in the design problem model and the retrieved cases to filter out any cases not directly relevant to the design problem. The exemplars are used as the queries against the case base.

Realizing the true purpose of the adaptation system to be a generalized parametric problem solving system, it is proposed that the exemplar

modelling system proposed above be integrated as the foundation for the adaptation system. While traditional case based system generally have the adaptation rules hard coded into the system, the proposed system with the integration of the exemplar modeller will provide the user with the ability to configure the adaptation process as desired. In this manner, the user may adapt the same retrieved solutions with different requirements in mind based upon the same general design problem.

6. DISCUSSION

The paper presents several examples to show the use of the concepts of the exemplar in embodiment design. It has been shown that exemplars provide a mechanism for converting application specific problems to domain independent problems. This has several advantages. First, it can lead to the development of general-purpose design automation systems. Second, it allows selection of appropriate solvers already in existence and creation of cleaner APIs between such solvers and applications. Third, canonical characterization can form the basis for data exchange standards between future CAD systems. The idea of exemplar networks, combining of established exemplars to form complicated exemplars, was introduced. The necessity for developing exemplar algebra was presented.

A case based system, which uses a retrieval module to find cases that most nearly satisfy the defined design problem, is described here. These retrieved cases are adapted to a single solution for the design problem. The adaptation module uses the first system as a method for modelling the different types of necessary information used in adapting the cases. Exemplars are found in the case base and may be used as initial skeletons on which to build the adapting exemplars.

7. ACKNOWLEDGEMENTS

This work was supported by NSF through grant and DMI-9812977.

8. REFERENCES

- Bettig, B., (1999), "A Graph-Based Geometric Problem Solving System for Mechanical Design and Manufacturing", *Ph.D. Dissertation*, Arizona State University.
- Clement, A., Desroches, A., Riviere A., (1991), "Theory and Practice of 3D Tolerancing for Assembly", *Proc. of CIRP Seminar on Computer-Aided Tolerancing*, Penn State University, USA.

- Dixon J., (1995), "Knowledge Based Systems for Design", *Journal Mechanical Design*, Vol. 117, pp 11 – 16.
- Elinson, A., Herrmann, J., Minis, Nau, D., Singh, G., (1997), "Toward Hybrid Variant/Generative Process Planning", DETC97/DFM-4333, *Proceedings of DETC'97: Design for Manufacturing*, Sacramento, CA.
- Goel, A., Chandrasekaran, B., (1989), "Use of Device Models in Adaptation of Design Cases", *Proceedings of the DARPA Workshops on Case-Based Reasoning*, San Mateo, CA, Morgan Kaufmann, pp. 307-16.
- Hua, K., Schmitt, G., Faltings, B., (1992), "Adaptation of Spatial Design Cases", *Artificial Intelligence in Design '92*, ed. J. Gero, Dordrecht, Kluwer Academic, pp. 559-75.
- Kramer, G., (1992), *Solving Geometric Constraint Systems*, The MIT Press, Cambridge, MA.
- Krampe, D., Lusti, M., (1997), "Case-Based Reasoning for Information System Design", *ICCBR-97*.
- Lee, D., Kang, J., Ryu, K., Lee, K., (1997), "Applying Memory-Based Learning to Indexing of Reference Ships for Case-Based Conceptual Ship Design", *ICCBR-97*.
- Maher, M., Balachandran, M., Zhang, D., (1995), *Case-Based Reasoning in Design*, Lawrence Erlbaum Associates, Publishers, Mahwah, NJ.
- Perera, S., Watson, I., (1996), "NIRMANI: An Integrated Case-Based System for Strategic Design and Estimating", *UKCBR-96*.
- Pierra, G., Ait-Ameur, Y., Besnard, F., Girard, P., Potier, J., (1996), "A General Framework for Parametric Product Model within STEP and Parts Library", *Proc. European Conf. Product Data Tech.*, London.
- Reddy, S., Fertig, K., Smith, D., (1996), "Constraint Management Methodology for Conceptual Design Tradeoff Studies", *Proc. 1996 ASME Design Eng. Conferences Conference*, Aug. 18-22, Irvine, CA
- Ramachandran, N., Langrana, N., Stienberg, L., Jamalabad, V., (1992), "Initial Design Strategies for Iterative Design", *Res. in Eng. Design*.
- SCRA, (1997), "Phase 4 Data Model for ENGEN", *Technical Report Version 4.2 (draft)*, South Carolina Research Authority, Charleston, SC, July, 1997.
- Serrano, D., Gossard, D., 1987, "Constraint Management in Conceptual Design", *Knowledge Based Expert Systems in Engineering: Planning and Design*, D. Sriram and R.A. Adey (ed.'s), Computational Mechanics, Southampton
- Shah, J., Rogers, M., (1993), "Assembly Modelling as an Extension of Design by Features", *Special Issue on Advances in CAD, Research in Engineering Design*, Vol. 5, pp. 218-237.
- Shirur, A., (1994), "Automatic Generation of Machining Alternatives for Machining Volumes", *M. S. Thesis*, Arizona State University.
- Sycara, K., Navinchandra, D., (1989), "Integrated Case-Based Reasoning and Qualitative Reasoning in Engineering Design", *Artificial Intelligence in Engineering Design*, ed. J. Gero, Springer-Verlag, New York, NY, pp. 231-50.
- Wang, W., Howard, H., (1989), "Design-Dependent Knowledge for Structural Engineering Design", *Artificial Intelligence in Engineering Design*, ed. J. Gero, Dordrecht, Kluwer Academic, pp. 267-77.
- Ward A., Seering W., (1989), "Quantitative Inference in Mechanical Design Compiler", *Proc. 1st DTM Conf.*, Montreal, Canada.