INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 03 STOCKHOLM, AUGUST 19-21, 2003

INVESTIGATION OF THE DESIGN EXEMPLAR AS A CAD QUERY LANGUAGE

AMEYA P. DIVEKAR AND JOSHUA D. SUMMERS

Abstract

Design engineers create CAD models with commercial Computer Aided Design (CAD) solid modeling systems and manage the data files through Product Data Management (PDM) systems. However these systems stop short of retrieving information stored "within" CAD models and it is believed that a query language in CAD may provide the desired access to the information stored in the CAD models. This paper seeks to identify the essential components and tasks of a CAD specific query language and investigate the design exemplar as a basis for the CAD query language. The design exemplar provides a standard representation of engineering design knowledge based upon a canonically derived set of entities and relationships for representing topologic and geometric problems. The paper concludes with an offering of the extensions to the design exemplar, necessary for the development of a true CAD query language.

Keywords: computer-aided design, design representations, geometric modeling, databases.

1 Introduction

As the proliferation of CAD use increases, the amount of knowledge that is generated in the course of design will in turn increase. With this immense collection of knowledge, designers need tools to access the information. This paper seeks to identify the needs of a CAD specific query language and investigate the design exemplar as a basis for a CAD query language. This paper offers necessary extensions to the design exemplar for it to qualify fully as a true CAD query language. Query languages are non-procedural, high-level computer languages and are primarily focused towards retrieving data held in files and databases. They could also be used for retrieval, updates, deletions and additions [1]. We have identified the essential characteristics of a query language and also the tasks that are performed by a complete query language. Existing database technologies have been investigated to support CAD data and their limitations have been discussed in [2]. The design exemplar provides a standard representation of engineering design knowledge based upon a canonically derived set of entities and constraints for representing topologic and geometric problems. It leverages the entities and constraints for representing the "semantic" information that a designer may map with the concept in mind. This mapping allows the designer to express geometric queries in the form of a bi-partite graph consisting of the entities and constraints. The data-structure of the design exemplar has embedded into it the capability to provide for change propagation. This allows for adapting the "similar" concepts to accomplish an exact match between the concept in the designer's mind and the changed characteristic. These capabilities of retrieving similar characteristics and changing them are based upon the generic exemplar algorithm for constructing and submitting constraint problems to a general constraint solving system.

2 Background

This section discusses the various issues associated with a query language like dialogues, query formulation and processing, query expression, components of a query language and the necessary functions performed by a query language.

There are mainly two main types of dialogues [1] user driven and system driven. In CAD/CAM, every record is likely to have varying amounts of data. Moreover, users will be restricted if they are asked to express the concept in their mind, through a predefined dialogue driven by the system. Hence, queries in CAD should be user-driven so that the designers have latitude in expressing their concepts in a complete manner.

A common example of query formulation within the constrained mode of dialogue is the following: (**FIND *target* WHERE *qualification***). The *target* data may be a file name, workspace, record name, or collection of data item names from one or more records. The *qualification* is a set of conditions involving data items in both the *target* and the related data. The conditions may be numerical or character string comparisons [1]. Queries may be processed by processing one record and making it available to the user or by processing all the records at once and displaying the records retrieved. In CAD/CAM, either approach may be appropriate, as it is irrelevant whether a single CAD model is processed at a time and the results shown or a list of the CAD models is displayed, allowing the users to browse through each record.

The query may be expressed in different manners and it is important that the appropriate mode of querying be identified for the particular application domain. Peabody (et. al) [3] classify queries as *textual* (based on key words), *query by example, query by sketch and iconic* queries for 2D matching and image retrieval. Users express spatial concepts when they query the database of CAD models. Egenhofer [4] argues that users prefer to sketch spatial queries, as they more readily support human spatial thinking. Hence, it is clear that a graphical query language will be more appropriate than a query language that requires the user to formulate the query lexically. Another way to query is to use a query already formulated and stored in a library. Such queries can also be combined with other s to formulate more complex queries. In other words, an earlier query is used as an example for retrieving information from the database. Since these queries use pre-formulated queries they may result in saving the time and effort required to reformulate queries.

We have identified the components of the de-facto query language SQL (Structured Query Language), which we believe are essential in making it a "query language". These components include data-types, predicates, and logical connectives. A data type is a set of data with values having predefined characteristics. Variables are instances of one of these data types. Some commonly supported categories of data-types in SQL are numeric, character, boolean, date/time, and objects. The predicate is a condition that can be evaluated to produce a truth-value of true, false, or unknown. The result is achieved by applying the predicate to a given row of a table. Some examples of these predicates are =, <>, >, <, between, IN, LIKE, IS NULL, IS NOT

NULL, or EXISTS. They are also referred to as “comparison operators”. They are critical components of a query language as they enable the construction of conditions that allow the users to access the data of interest to them. SQL provides, at a minimum, for the logical connectives: AND, OR, MINUS. These represent the intersection, union, difference of relational algebra. These operators enable the construction of compound conditions within a single query. These operators may also be used to combine two queries so that the result sets of both queries may be obtained.

A query language will be expected to perform the following tasks on a database: retrieval, updates, deletions, and additions [1]. In addition, Egenhofer[5] has given a set of requirements to be satisfied for a spatial query language, the relevant requirements discussed here. Users must be able to treat spatial data at a level independent from internal coding such as x-y co-ordinates. The results should be displayed in graphical form, as it is the most natural form to analyze geometric data. It should be possible to combine one query result with the results of one or more previous queries giving rise to a dynamic interaction. Graphical presentations may require the display of context in addition to the information sought. An extended dialog allowing selection by pointing and direct selection of a result as a reference to an upcoming query is required. Graphical presentation of query results may require dedicated language tools. Labels are important in understanding drawings so that users are able to select specific instances of objects.

3 GEOMETRIC QUERY LANGUAGES /MECHANISMS

There have been many proposals at developing a geometric query language. Most of them have attempted to extend SQL to spatial data. Egenhofer [5] gives a detailed comparison of the query languages GEOQL, Extended SQL, PSQL, KGIS, and TIGRIS. These are all extensions of SQL for the GIS domain. Chan, et. al, [6] have developed a geometric query language in the GIS domain built on SQL(QL/G) by incorporating relevant geometric data-types and predicates. In GIS applications the geometric data, such as the location of a city, can be uniquely stated whereas in CAD models the locations may change with translation, rotation, and scaling of models.

A query language that offers support for geometric data types is PostgreSQL [7]. Geometric data types like point, line, box, path, and circle can be represented in the language. The geometric predicates allow the comparison of geometric data-types. For example, predicates can check whether an entity is to the right of, left of, above, below another entity. From the list of data-types and predicates, it can be observed that this query language supports 2D geometry.

A geometric query language for process planning has been developed [8]. This approach allows for querying against a single part file. It is assumed that the part exists in a feature-based system and “tag” a feature type to each feature. A list of relations and operations are defined and the models are processed to form a representation of the part in the form of tables. These tables have all the relations and the entities in the part satisfying them. A query is processed against this tabulated information using a list of defined operations. This method requires preprocessing of the CAD models before they can be queried.

Yang, et. al [9]have developed a query approach to retrieve features from part files. They use the Attribute Adjacency Graph (AAG) for representing the features of the parts. They classify the features into a set of primary and secondary features and describe “parent-child” or “same-level” spatial relations between them. This approach makes use of SQL to query preprocessed data of CAD models populated in the database. This approach allows designers to query against a database of CAD models and retrieve the features desired by the users. However, the approach allows limited capability to query the CAD models, as those features and relations defined in the system may only be used. Moreover, there is no provision for quantitative predicates to compare the dimension values of features.

Rosenthal, et al [10] used an analogy between the data-structures used for the data about parts, which they referred to as part hierarchies, and the part-of relationship. The queries retrieve the parts that intersect, lie totally or within a distance from the given volume in an assembly. The queries are suitable for environments in which the parts are arranged hierarchically and may find application in situations where the engineer would want information regarding the parts that are contained in an assembly, as well as the number of those parts available in the inventory. However, this approach may not be useful in situations where the designer wants information about the features of a particular part.

Querying of the data contained in Express modeling format has been studied [11]. This query language is designed to aid the moving of files from one CAD system to another CAD system. This allows proprietary systems to only query data in the STEP files, which may be of relevance to them. Spatial database integration for novel CAD applications into off-the-shelf database systems has been investigated [12]. The system allows users to query spatial regions for the parts that may lie completely, intersect or lie within the specified distance of the selected region. This system is of limited value since it does not allow the design engineer to retrieve the CAD models that carry the semantic information that is sought.

As has been previously explained, designers may be looking for models that are globally similar to an existing model. Much work is going on in retrieving structurally similar models and the research has culminated into many geometric search engines. Some popular 3D geometric search engines include those of [13-20]. CAD models may be retrieved based on global similarity, or by matching characteristics expressed in a query or by using geometric information to retrieve parts from an assembly.

Most of the work in geometric query languages has been done in the field of GIS, with a majority of them being mere extensions of SQL. In GIS systems users deal with maps, and the geometric data in the maps is populated in the database. Thus, the whole database is the information from a map and users usually query against positions of various entities like rivers, cities, towns etc. in the map. It is important to realize that we are dealing with geometry of a single CAD model as well as the entire database of CAD models. This is an important distinction between CAD query systems and GIS query systems.

4 Exemplar Definition

Exemplars provide a standard representation of mechanical engineering design knowledge based upon a canonically derived set of entities and constraints for representing topologic and geometric design problems. Exemplars are bi-partite graphs, where one set is composed of a number of entities and the other set, a number of relationships or constraints, such that every member of the first set is related to at least one member of the other set and no member of the same set. The exemplar is composed of two pairs of orthogonal sub bipartite graphs (Figure 1) of entities and constraints: match/extract (used for retrieval) and alpha/beta (used for modification). The entities and constraints are based on those derived by [21]. A bi-partite graph representation scheme is chosen as it fits well with boundary representation for geometry and equation set modeling for parametric representation. One sub-graph (match) of the exemplar corresponds to the entities and constraints that are explicitly stored in the model. The other sub-graph (extract) represents the information that is not stored explicitly in the model, but may be inferred through reasoning. The extract part represents the relations that must hold true in addition to the matched part, thus facilitating reasoning about the matched part of the exemplar. The transformation axis of the exemplar represents the alpha and beta sub graphs of the exemplar allowing modification of models from alpha state to the beta state.

The step feature, as shown in the Figure 2, consists of the four planes with every two planes sharing an edge with the correct convexity. The information regarding the planes and the curves that bound them is found explicitly in the model and constitutes the match portion of the q_step exemplar. The information about the convexity of the edges may not be necessarily stored explicitly in the CAD model and needs to be reasoned. This constitutes the extract portion of the exemplar and is represented by dashed lines. The convexity of the edges is determined by a sequence of steps. Topologically correct normals to the planes are extracted and a cross product of the vectors is taken. The resultant vector is then compared with the correct direction of the curve to infer the convexity of the edge. While a query to retrieve a simple feature was demonstrated, a range of more complicated features have been retrieved with exemplars [22]. Also, the modification of geometric information has been demonstrated using the exemplar [23].

Figure 3 illustrates the design exemplar for the step feature as found in Figure 2. The design exemplar is represented here in graphical bi-partite form where the circles represent entities and the squares represent constraints. The solid lines are associated with match elements and the dashed lines with extracted elements. Thus, Plane P1 is a matched entity, while the Cross Product is an extracted constraint.

The various aspects of a query language were discussed in the previous section. The qualifications were summarized and the appropriate qualities with respect to a CAD query language were outlined. The design exemplar is discussed here as it relates to these various aspects of a query language. The design exemplar allows queries to be expressed through a graphical interface and the queries are user-driven. The implementation of the design exemplar allows users to sketch the queries and hence allows expression of spatial queries in a graphical manner as suggested by [4].

5 Investigating the Design Exemplar as a CAD Query Language

A possible way of investigating the exemplar as a query language would be compare it with the de-facto query language SQL. We have identified the essential components and tasks of a query language in our background section and will now investigate if the exemplar has all these components and performs all these tasks. Further, we investigate if the exemplar also satisfies the requirements of a spatial query language as given by [5].

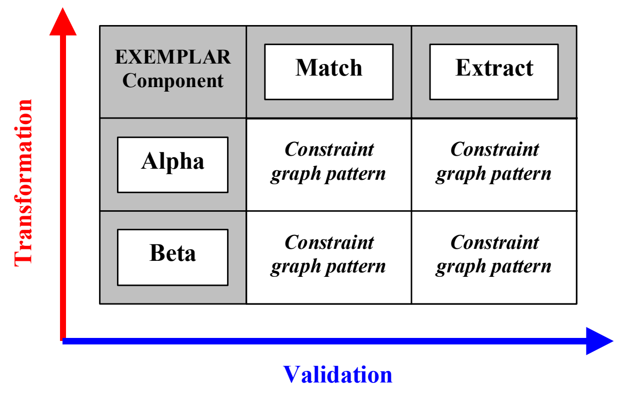

Figure 1 .Components of an Exemplar

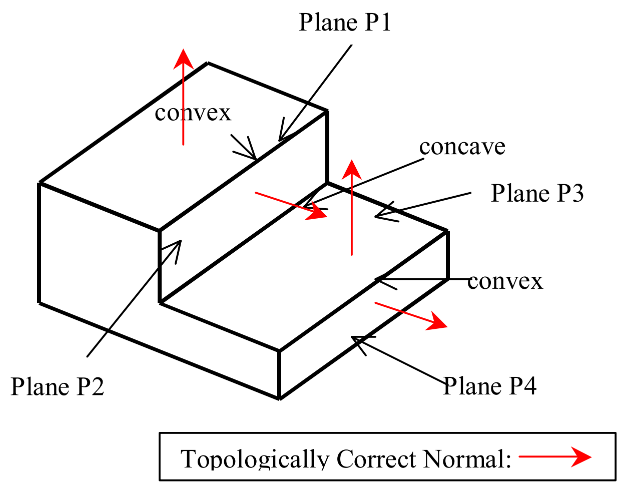

Figure 2. Example of a step feature

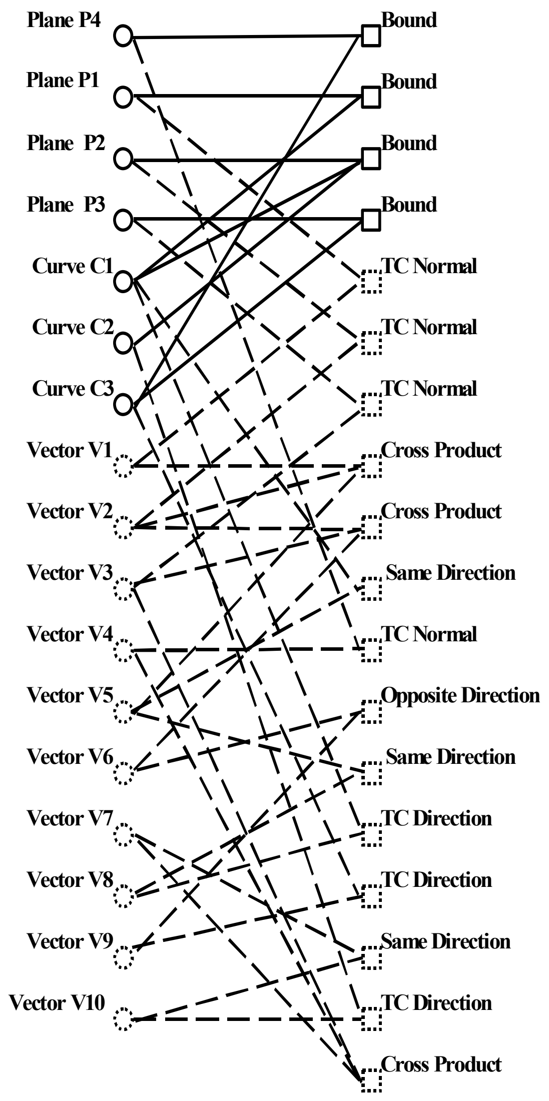

Figure 3. q_step exemplar to retrieve the step feature

SQL incorporates the data-types that a user might encounter when dealing with lexical data. For a query language in CAD, one might expect to have all the data-types that are present in the existing representations of the CAD models. The inclusion of these data-types enables the users to operate at the same abstraction level as that of the data. Table 5 gives a partial list of the entities that may be used for all data used for formulating the query. It states the level of abstraction at which the data-types operate. A complete list is given in [21]. The exemplar, being developed based upon this canonically derived set of entities, has the necessary data-types that would be expected of a CAD query language.

Predicates allow the construction of conditions used to compare the variables, which are instances of data-types, with other variables, constants, literals etc. When expressing spatial concepts variables are instances of the entities and they may be compared with other spatial variables or constants if the entities have numeric attributes. Again, a CAD query language may be expected to allow the formulation of conditions at the same level at which data is stored in CAD models.

Table 6 gives a representative list of the predicates that may be constructed using exemplars. A complete list can be found in [21].

Table 5 – Data-types essential in CAD

Abstraction Level	Entity
Algebraic	Real parameter, Integer parameter, Vector, Rotation Matrix
Geometric	Point, Direction, Line, Plane, Circle, Ellipse, Cylinder, Sphere
Topologic	Solid volume
Semantic	Form Features, Part ,Assembly

Table 6 – Predicates used in exemplars

Abstraction Level	Predicates
Algebraic	Scalar equations, Scalar inequalities, Vector equation, Cross Product
Geometric	Distance Angle Radius, Focal Distance, Distance to resolved geometry, Control points, Knot values, Continuity conditions, In_Set, Map Coincident ,Incident Parallel ,Right Angle
Topologic	Boundary, Length, Area, Volume, Directed_Left_Of, Curve Direction, Curve Direction TC, Surface Normal, Surface Normal TC, Same Direction
Semantic	ID, Part Of, Same

These geometric predicates take the form of constraints (relationships) that are applied between the entities. These constraints could mean placing a condition between two entities to be parallel, perpendicular, tangent etc. It may be noted here that since scalar equations may be used as constraints, the predicates =, <>, <, >, <=, >= can all be embedded into the query. Thus the exemplar includes the predicates from SQL that are relevant for CAD and also includes all the others that are essential for CAD. Other than these predicates, SQL also supports arithmetic operators like Addition (+), Subtraction (-), Multiplication (*), Division (/). The exemplar allows the use of the operators to be incorporated in equations that may be applied as constraints to the entities.

Logical connectives represent the usage of intersection, union, difference of relational algebra and are often referred as "logical operators" in the context of query languages. The exemplar, due to its inherent representation, accommodates for the intersection connective. The union (OR) connective and the difference (MINUS) connective are not yet implemented in the exemplar. A complete query language should include all three Boolean operators. Implementing the union and difference connective is the focus of current research. It was also found that the NOT connective implemented in SQL might be relevant for a CAD query language as it allows users to filter out conditions and express their query so that they may retrieve only the relevant information.

In this section we investigate whether the exemplar performs all the previously identified tasks of a query language. The exemplar modeler provides a user-interface to build queries, then processes them and finally displays the result set, thus allowing the user to browse through it. The essential tasks identified for a query language are: retrieve, update, delete, and add. Comparing these functions with design tasks of Table 4, it can be seen that the retrieve function corresponds to the first three exemplar design tasks (pattern matching, query extraction, and design validation) whereas the other three query language functions of update, delete, and add correspond to the modify task of the exemplar (Figure 2). The exemplar can perform all the tasks that might be expected of a query language.

The design exemplar has been demonstrated useful for retrieving CAD information, both explicit and implicit, from single models [23]. Further, extensions to the system have provided support for query multiple models with a single query [24]. The tasks of adding, modifying and deleting data rest on the concept of transformation in exemplars. In the context of CAD, there may be instances, where the designer wishes to extract information, validate it for some criterion and finally modify it if it does not satisfy the criterion This may be achieved by expressing the chracteristic to be changed in the alpha sub-graph of the exemplar and the desired final form in the beta part of the sub-graph. However the exemplar does not validate the data-integrity or unresolved geometry when it modifies the characteristic. The addition and deletion functions are also based on the same concept as the update and complex exemplars have been developed that can illustrate the task of adding and deleting data from the database of CAD models. Exemplars recognizing certain characteristics and modifying them may be found in [24]. Thus, it may be concluded that the exemplar is able to perform all the tasks expected of a query language.

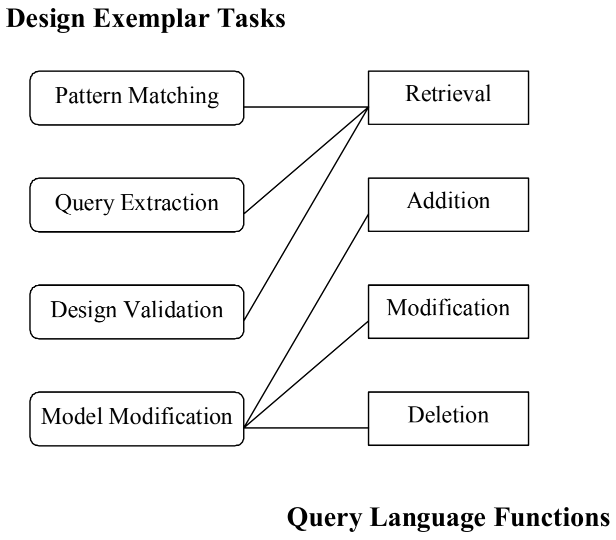

Figure 2 – Task to Function Mapping

Now consider the requirements of a spatial query language by [5]. The exemplar allows users to sketch the query using entities and constraints irrespective of the x-y coordinates. The result of the query is expressed either by highlighting the sought geometry or displaying resulting parameter values in a dialog box. Every instance of an entity has a name attached and it is displayed to the user, thus allowing the user to understand the objects better. The exemplar appears to satisfy the relevant requirements given by Egenhofer for spatial query languages.

6 Discussion

The design exemplar offers the expected functionalities of query language. It effectively performs the functions of retrieving, adding, modifying, removing geometric data from CAD models. It satisfies the general format of a query: (**FIND *target* WHERE *qualification***.). The exemplar finds the CAD files (*target*), which satisfy the match and extract portions of the exemplars (*qualification*). The exemplar relies upon the algebraic, semantic, topologic, and geometric entities and constraints; facilitating multiple levels of domain independent design queries. This is significant, as most previous query systems in CAD were restricted to specialized domains like process planning, manufacturing etc. Moreover, with the exemplar the query is user centric and is not limited to a library of predefined features.

The ability to query the position or retrieving the parts of an assembly that may be enclosed, intersect or within a specified distance from a volumetric region, seem valuable queries that are not currently implemented. However, these provide interesting directions for future work using exemplars. In extending the design exemplar for satisfaction as a complete query language, the logical connectives must be implemented.

References

[1] The British Computer Society, Query Language Group, Query languages: a unified approach, Heyden & Son Ltd., London, UK, 1981.

[2] Liu, M., "On CAD Databases", Proceedings of IEEE Canadian Conference Electrical & Computer Engineering, Edmonton, Canada, May 9-12, 1999.

[3] Peabody, M, Regli, W., Mc Wherter, D., Shokoufandeh, A, "Clustering Solid Models for Database Storage", Technical Report DU-MCS-01-04, 2001.

[4] Egenhofer, M. "Query Processing in Spatial-Query-By-Sketch", Journal Of Visual languages And Computing, 1997, Vol. 8, No. 4, pp.403-424.

[5] Egenhofer, M., "Spatial SQL: A Query and Presentation Language.", IEEE Transactions on Knowledge and Data Engineering, 1994, Vol.6, No.1, pp.86-95.

[6] Chan, E., Zhu, R., "QL/G – A Query Language for Geometric Data Bases", Proceedings of the 1st International Conference on GIS in Urban Regional and Environmental Planning, Samos, Greece, April 1996, pp. 271-86.

[7] "PostgreSQL", http:// techdocs.postgresql.org.,2002.

[8] Silva, R., Wood, K., Beaman, J,"An Algebraic Approach to Geometric Query Processing in CAD/CAM Applications", Symposium on Solid Modeling Foundations and CAD/CAM Applications, Austin, TX, 1991, p. 73.

[9] Ou-Yang, C. and. Liou, P.Y."Applying the Topological Relationships of Form Features to Retrieve Part Models from a CAD System". IIE Transactions,1999,Vol.31, p.323-p.337.

[10] Rosenthal, A., Heiler, S., Manola, F. "An Example of Knowledge Based Query Processing in a CAD/CAM DBMS", Proceedings of the tenth International Conference on Very Large Data Bases, 1984, pp. 363-370.

[11] Koonce, D., Huang, L., Judd, R.. "EQL: An Express Query Language", Elsevier Science ltd, Computers Ind. Engg. 1998, Nos1-2, pp. 271-274.

[12] Kriegel, H, Müller, A, Potke, M, Seidl, T " DIVE: Database Integration for Virtual Engineering", Demonstration Proceedings 17th Int. Conference on Data Engineering (ICDE), Heidelberg, Germany, 2001, pp. 15-18.

[13] Arizona State University, "ASU – Prism", http:// 3dk.asu.edu,2002.

[14] IBM - Tokyo Research Lab "3D Geometric Search", 2002,http://www.trl.ibm.com/projects/3dweb/SimSearch_e.htm, 2002.

[15] Heriot-Watt University, "ShapeSearch.net", http: //www.hw.ac.uk/mecWWW/research/3Dsearch/3Dsearch.htm , 2002.

[16] Sandia Laboratories, "Geometric Search Engine", http:www.sandia.gov/isrc/Working_with_Us/Organization_Chart/IS_Principles/Geometric_Search_Engine/geometric_search_engine.html, 2002.

[17] Min, P., Chen, J., Funkhouser, T."Evaluating Sketch Query Interfaces for a 3D Model Search Engine", Workshop on Shape-Based Retrieval of 3D Models, Princeton, New Jersey, October 28-30, 2001

[18] Brown University, "3D Object Recognition" http://www.lems.brown.edu/~cmc/3DRecog/overview.html , 2002.

[19] Universiteit Utrecht, "3D Shape Retrieval Engine" http://www.cs.uu.nl/centers/give/imaging/3Drecog/3Dmatching.html , 2002.

[20] Berchtold, S., Keim, D.: "Section Coding: A Similarity Search Technique for the Car Manufacturing Industry". IADT 1998: pp.256-263.

[21] Bettig B., Shah, J., "Derivation of a standard set of geometric constraints for parametric modeling and data exchange", Computer aided Design, 2000, V33 (1), pp 17-33

[22] Venkatamaram, S., Summers, J., Shah, J. "An Investigation of Integration of Design by Features and Feature Recognition", Feature Modeling and Advanced Design for the Life Cycle Systems, IFIP, Valenciennes, France. , 2001.

[23] Bettig, B., Shah, J., Summers, J., "Domain Independent Characterization of Parametric and Geometric Problems in Embodiment Design", ASME Design Engineering Technical Conference-, DAC-14259, 2000, Baltimore, MD.

[24] Summers, J., Lacroix, Z., Shah, J., "Case-Based Design Facilitated by the Design Exemplar", 7th International Conference on Artificial Intelligence in Design, ed. J. Gero, Kluwer Academic Press, Netherlands, 2002, pp. 453-76.

For more information, please contact:

Joshua D. Summers Clemson University, Mechanical Engineering, Clemson, SC 29634-0921 USA
Tel: Int 1 864 656 3295 Fax: Int + 1 864 656 4435 E-mail: joshua.summers@ces.clemson.edu
URL: http://www.vr.clemson.edu/credo/AID/