Towards Establishing the Design Exemplar as a CAD Query Language

Joshua D. Summers[1], Ameya Divekar[2] and Srinivasan Anandan[3]

[1]Clemson University, joshua.summers@ces.clemson.edu
[2]PTC Inc., adivekar@ptc.com
[3]Clemson University, sananda@clemson.edu

ABSTRACT

Computer aided design solid models are frequently interrogated by design engineers and other design stakeholders for reasons such as design reuse, standardization, manufacturability analysis, compatibility, feature extraction, or requirement validation. Currently, two approaches are employed in interrogating these solid models: interactive user interrogation and predefined feature recognition. A new interrogation approach is proposed here based upon the design exemplar. Using this new engineering knowledge representation schema, engineers may interactively define queries against their design models incorporating geometric, topologic, semantic, and algebraic entities and relations; specifying both the explicit and implicit desired characteristics. This paper presents the requirements for a CAD query language and positions the design exemplar as an appropriate solution for this problem.

Keywords: design exemplar, CAD query language, retrieval, interrogation

1. INTRODUCTION

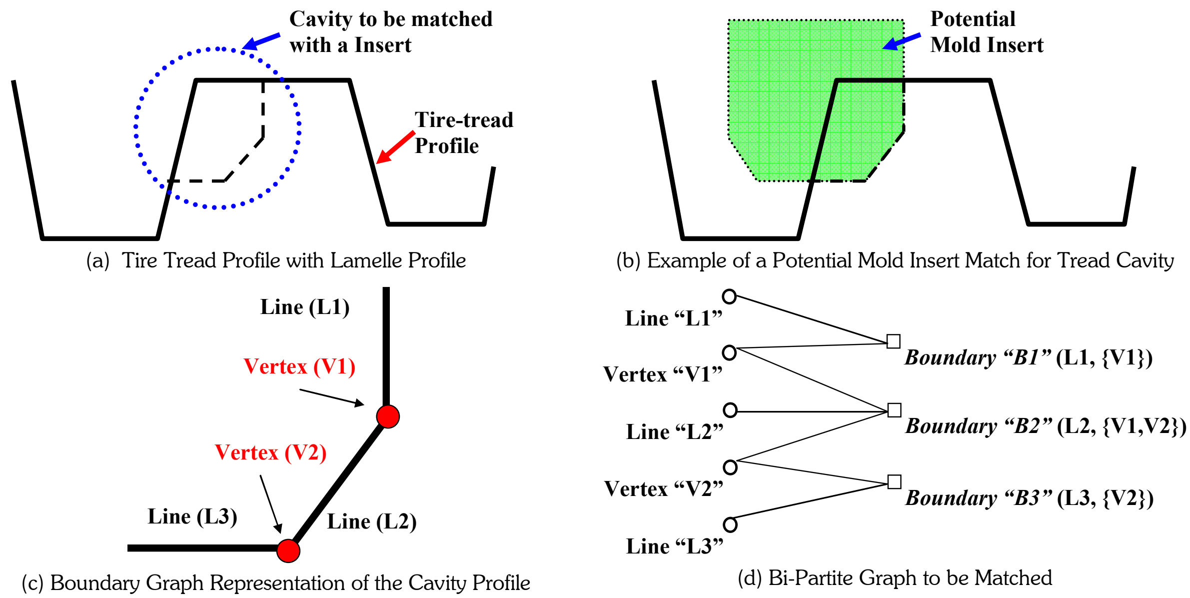
Just as the Internet and the vast amount of associated stored lexical data has propelled the need for developing new search and retrieval technologies, the proliferation of computer aided design (CAD) and product data management (PDM) technologies and the voluminous generation of CAD models necessitate the development of search and retrieval tools that are designed for CAD data: geometric, topologic, numeric, and semantic. This CAD search and retrieval based on user-defined queries is important to engineering design. The design exemplar was offered as a tool for CAD interrogation that combines the flexibility of direct interaction with the efficiency of custom feature recognition systems [24]. Recently, the design exemplar has been investigated as a CAD query language with logical connectives identified as the required extensions to fully realize a CAD query language [9].

A simple example of a possible situation where a CAD query language could prove quite useful is in tire mold design (Fig. 1). Consider, a tire designer has created a tire tread profile that requires a mold insert. The mold designer is tasked with either finding an existing insert from previous tire designs or creating a new tire insert to be used in the new design. In a global tire manufacturing company, the number of inserts that have already been developed may be extremely high with no systematic or uniform method for indexing the previously designed and fabricated mold inserts. The mold designer would like to be able to find inserts from this large database that are suitable to the specific needs, conformance to the tire profile design. A query language that allows the mold designer to define the desired characteristics, such as angles, depths, thicknesses, and number of cross holes, would facilitate this process, eliminating the creation of duplicate tire inserts, saving the company significant money and development time. The problem is defined as:

> *"a cavity (defined by planes, cylinders, lines, circles, points) is to be matched to a model (insert) that corresponds to the geometry and topology within specified angular and distance tolerances."*

As the shapes of the tire cavities change across multiple models and product lines and as the local geometric regions, not global shapes, of the inserts are what are of interest to the designers, traditional feature recognition and shape recognition systems would not provide adequate support. The designers need the flexibility to interactively define the queries, refining them with additional parametric bounds and relations. Thus, the designer needs a true CAD query language to satisfy their requirements.

The design exemplar was first introduced as a declarative representation for geometric problems. Design exemplars are based upon the basic observation that parametric and geometric related queries about engineering designs deal with some characteristic of interest for the designer. These characteristics derive from a pattern of entities and relations



(a) Tire Tread Profile with Lamelle Profile

(b) Example of a Potential Mold Insert Match for Tread Cavity

(c) Boundary Graph Representation of the Cavity Profile

(d) Bi-Partite Graph to be Matched

Fig. 1. Example Mold Insert Scenario Demonstrating Need for CAD Query Language Capabilities.

found explicitly or implicitly in the design model. Design exemplars represent these characteristics as patterns of topologic, geometric, algebraic, and semantic relationships. Design exemplars have been used in feature based design and feature recognition systems, used to model standard design procedures, used for rule validation and querying, and proposed for use as view transformation mechanisms (e.g. [24, 25]). The design exemplar has been investigated as a CAD query language by comparing the components of the de-facto query language, SQL, with those of the exemplar. Logical connectives (NOT, OR, and AND) have been identified as required extensions.

The potential users of such a query language may include anyone retrieving geometric information for various purposes. Specifically, two groups of people are anticipated to benefit the most from such a query language. First, designers may search for CAD models that match a specific characteristic. The query may be made specific by including more relations in the extract and may allow a modification to be applied to the retrieved characteristics. The second group of people anticipated to benefit from this query language is researchers developing various design automation systems. The CAD query language may eliminate the need to develop special retrieval algorithms and enable the researchers to focus on the processing of the geometric information retrieved by the query language. We have shown in previous work that the design exemplar overcomes limitations of earlier query languages, operating in a domain independent environment by using either a pre-defined library of queries or user defined queries [9]. The design exemplar allows users to build their own queries (user-centric) while enabling them to operate with the CAD vocabulary (data-centric). This query language goes beyond features by enabling the users to encapsulate the semantics of the geometric data and provides for comparison of CAD models based on key parameters and dimensions [20]. It is believed that as the exemplar based query language evolves, it will play a vital role, not only in retrieving geometric information during the various design stages, but also in automation of the design process.

This paper seeks to identify the needs of a CAD-specific query language based upon an analysis of the essential characteristics and the tasks performed by traditional query languages. Query languages are non-procedural, high-level computer languages that are primarily focused towards retrieving data held in files and databases. They are also used for updates, deletions, and additions in databases. Geometric query mechanisms and search engines are evaluated to determine the extent to which the design exemplar can support similar functionality. Ultimately, the design exemplar is offered as a suitable foundation for a true, complete CAD query language.

2. QUERY LANGUAGE REQUIREMENTS

Query languages are non-procedural computer languages, where the user specifies what is to be done and not how it is to be done. They are primarily focused on retrieving data held in files and databases. The user has control over the desired functional result. While the principle behind a query language is not limited to any domain specialization; practicality suggests that specific domains may require a customized language, with extended vocabulary and syntactic rules. The power of a query language to select the data relevant to the user and keep out the irrelevant data could be assessed in terms of the ability to materialize the target data from the underlying data structure and also the variety of ways in which the conditions can be constructed and connected. This section discusses query dialogue, formulation and processing, expression, components, and functions.

2.1 Dialogue

There are two main types of query dialogues: user driven and system driven. In the system driven dialogue, the user responds to messages initiated in the query system. In systems where all records of data have the same structure and quantity of data, a system driven approach may be useful. User driven dialogue implies that the user employs a language to define his query. Most of the so-called "English-like" query languages today fall into this class. Commands have a defined syntax. Further, only predicates from a specified list may be applied. In computer aided design and computer aided manufacturing (CAD/CAM), every record (model) is likely to have varying amounts and types of data. Moreover, users would be restricted if they are asked to express the desired concept to be queried, 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.

2.2 Query Formulation and Processing

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 numeric or character string comparisons. Complex queries can be built by performing simple comparisons either by using logical connectives (AND, OR, MINUS) in a single query or by nesting.

Query processing generally operates in one of the following two basic ways:

- One record that satisfies the conditions is made available to the user. To retrieve all the records satisfying the conditions, the query statement is repeated in a procedural loop.
- All records that satisfy the conditions are made available to the user. This is implemented either by printing the selected records as they are retrieved or by using a workspace to store the selected records (or their references, pointers, etc.). In this case the display generally shows a count of the number of records selected. Such a workspace helps a user refine queries and provide more flexibility.

In CAD/CAM, either approach may be appropriate, as it is irrelevant whether a single CAD model is processed and the results shown or a list of the CAD models is displayed, allowing the users to browse through each record.

2.3 Query Expression

Queries may be expressed in different manners and it is important that the appropriate mode of querying be identified for the particular application domain. Four ways in which queries could be expressed for 2D matching and image retrieval include [16]:

- *Textual* query is based on keywords.
- *Example* query uses similarity measures derived off a set of query images provided as input.
- *Sketch* query looks for image segments matching the sketched profile.
- *Iconic* query uses templates of aspects of the desired image to identify images with similar features.

Users need to express spatial concepts when they query the database of CAD models. This implies that lexical descriptions of the query in these spatial situations will be ambiguous and may easily lead to misinterpretations. Dialogue boxes and menu driven queries provide little help as they use the same syntax and grammar as lexically expressed languages, only providing support of external memory for the user [11] Since CAD data is spatial and graphical, it is logical to express it in terms of explicit spatial concepts. Egenhofer argues that users prefer to sketch spatial queries, as they more readily support human spatial thinking. 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. Users of a CAD query language may also want to store already formulated queries that may be combined with others to formulate more complex and compound queries. This may also result in savings of time and effort while formulating new queries and hence such a facility of combining pre-existing queries may be expected of a CAD query language.

2.4 Components

The components of the de-facto query language SQL (Structured Query Language) [12] that are essential in making it a "query language" include data-types, predicates, and logical connectives. A data type is a set of data with values having predefined characteristics. Some commonly supported categories of data-types in SQL are numeric, character, Boolean, date/time, and objects. Variables are instances of one of these data types. A predicate is a condition that can be evaluated to produce a truth-value response. Some examples of "comparison operator" predicates are =, <>, >, <, between, IN, LIKE, IS NULL, IS NOT NULL, or EXISTS. SQL supports, at a minimum, the logical connectives of AND, OR, and MINUS. These represent the intersection, union, and 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 resulting retrieved sets of both queries may be obtained.

2.5 Tasks

A query language is expected to perform the following tasks on a database: retrieval, updates, deletions, and additions. Consider a table in a relational database that contains information about the names of employees and the department to which they belong. Tab. 1 lists the tasks that are typically performed by a query language and provides an example for each task and expresses the query in SQL.

Task	*Example*	*Query with SQL*
Data Retrieval	Retrieving names of employees from 'CS' Dept.	SELECT Employee_name FROM Employee_table WHERE Dept_name = 'CS'
Data Addition	Adding a record of employee Joe.	INSERT into Employee_table VALUES ('Joe', 'CS')
Data Modification	Changing the Department of Joe from 'CS' to 'ME'.	UPDATE Employee_table SET Dept_name = 'ME' WHERE Employee_name = 'Joe'
Data Deletion	Deleting all employees from 'CS' Department.	DELETE FROM Employee_table WHERE Dept_name = 'CS'

Tab. 1. Tasks Performed by a Query Language.

In addition to the qualifications of a query language a CAD query language may be expected to meet the requirements of a spatial query language as given by [11]. 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 MECHANISMS

3.1 Geometric Query Languages

Much of the effort in developing a geometric query language has been through extension of SQL to spatial data. For example, a geometric query language (QL/G) in the geographic information systems (GIS) domain built on SQL has been developed [7]. The language uses geometric data types (e.g. REGION, LINE, or POINT) to serve the needs in 2D applications like GIS. In CAD where geometric data is defined in three dimensions, these data types may not be sufficient, but could be extended, theoretically, to accommodate data types found in CAD. Further, 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.

Another query language supporting geometric data types is PostgreSQL where geometric data types (e.g. point, line, box, path, polygon, or circle) can be represented [4]. 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, or below another entity. This query language only supports 2D geometry.

Another approach, for process planning, allows for querying against a single part file where it is assumed that the part exists in a feature-based model with "tags" on each feature [21]. A list of relations and operations are defined to process the CAD models 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.

Another feature retrieval from part files approach proposed is based on the Attribute Adjacency Graph (AAG) for representing the features [17]. This language relies heavily upon preprocessing the CAD data into a new representative format. The transformation module in their system transforms the B-Rep structure of the part files to a feature database where the features are classified into a set of primary and secondary features with “parent-child” or “same-level” spatial relations between them. A relation table contains the relations between the features. This approach uses SQL to query preprocessed data of a database of CAD models, retrieving models with features desired by users. The approach has limited capability to query CAD models because only those features and relations defined *a priori* in the system may be used. Moreover, there is no provision for quantitative predicates to compare the dimension values of features.

To formulate intersection, proximity, and containment queries, bounding boxes may be used as geometric predicates [19]. The queries retrieve the parts that intersect, lie totally, or within a distance from the given volume in an assembly. The predicates are classified as either downward monotonic (if a part satisfies the predicate, then all descendents are said to satisfy the predicate) or upward monotonic (if a part satisfies a predicate, then all predecessors are said to satisfy the predicate). Further, attributes are classified as order preserving or inverse order preserving depending on whether the attributes have lower or higher values for the descendants. The queries are suitable for environments in which the parts are arranged hierarchically. However, this approach may not be useful in situations where the designer wants information about the features of a particular part.

A textual query language, based on SQL and used to interrogate the Express modeling format, has been offered to aid the moving of files from one CAD system to another [14]. This allows proprietary systems to only query attribute data in the STEP files that may be of relevance to them.

Finally, voxelized geometries of VRML (virtual reality modeling language) models have been used to integrate CAD applications in a common database system [15]. Enabling the generated voxel set, in order to use it as a spatial key, it is transformed to an interval sequence on a space-filling curve and stored in a tree. This approach relies upon an approximation of the VRML models, which are approximations of original CAD models. The system allows users to query spatial regions for the parts that may lie completely within a selected region, intersect the selected region, or lie within the specified distance of the selected region. The queries offered by this system are of limited value, as they do not encapsulate the semantics of the data.

In comparing these different approaches, a set of comparisons is offered (Tab. 2). These comparisons include the dimension of the geometric information (2D vs. 3D), the scope of querying (single file vs. multiple files), the processing of data required (pre-processing vs. no processing), the application domain, and the form of querying. To the best of our knowledge, this is a first attempt to derive the essential components of a query language and develop a CAD query language based on the requirements derived from a standard query language.

Query Approaches	*2D/3D*	*Single (S)/ Multiple (M)*	*Pre-processing Required (Y/N)*	*Application Domain*	*Form of Querying*
[7]	2D	S	N	GIS	Mixed Lexical/Graphical
[14]	3D	S	?	CAD	Lexical
[21]	3D	S	Y	CAD	Lexical
[17]	3D	M	Y	CAD	Lexical
[19]	3D	M	N	CAD	Lexical and Guided
[4]	2D	S	N/A	General	Lexical
[15]	3D	M	Y	CAD	Unknown

Tab. 2. Comparison of Query Approaches.

3.2 Geometric Search Engines

Designers may look for models that are globally similar to an existing model. There are many researchers working on retrieving globally similar models and the research has culminated into many geometric search engines. There are conceptually three different types of queries in the context of CAD models:

- Retrieving CAD models that globally match the query model [1-3, 5, 8, 13, 16, 18]
- Retrieving CAD models that locally match the geometric characteristics expressed in the query[17, 21]

- Retrieving regions within a CAD model that match the geometric characteristic expressed in the query [15, 19]

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 where the geometric data is populated in an associated database. The database is the information from a map and users usually query against positions of various entities (e.g. rivers, cities, towns). It is important to realize that CAD-users are dealing with geometry of a single CAD model as well as the entire database of CAD models.

4. EXEMPLAR DEFINITION

The design exemplar is a data structure that, when combined with a generic constraint solving algorithm, is a construct representing geometric and parametric design problems. Designers, using exemplars, can find geometric properties (e.g. walls with a specific thickness) in CAD models and then change these properties as needed (e.g. the wall thickness from 0.1" to 0.5"). For example, a designer searching for a circle with a radius between 1" and 3" may create the exemplar illustrated in Fig. 2 and shown in text format in Fig. 3.

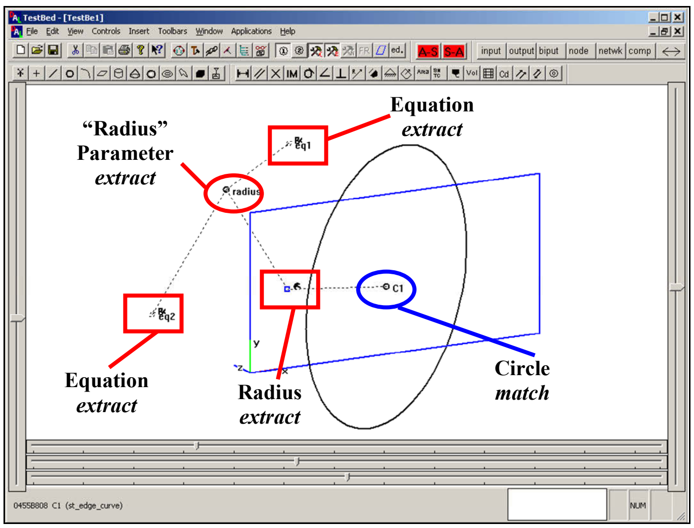


Fig. 2. Circle Radius Checking Exemplar.

Match:
Circle "C1";
Extract:
Parameter "radius";
***Radius* (radius, C1);**
***Equation* "eq_1" (radius);**
***Equation* "eq_2" (radius);**

eq_1: radius < 3
eq_2: radius > 1

Fig. 3. Circle Exemplar.

The circle entity is designated *match*, meaning that the design exemplar entity must be matched in a model. The radius constraint relating the radius parameter and the circle entity is used to extract the radius value from the matched circle in the model. This radius parameter is then evaluated using two equations (radius > 1; radius < 3). The radius parameter, radius constraint, and two equations are designated as *extract* as these are not actually found in the model, but are used by the exemplar algorithm to determine the geometric/parametric aspects of the matched circle. At this point, this design exemplar can be applied against various CAD models composed of circles, lines, planes, or any other geometric entities to determine if any of them have a circle with a radius between 1 and 3. The exemplar is composed of two pairs of orthogonal bipartite sub-graphs (Fig. 4) of entities and relations: *match/extract* (used for retrieval) and *alpha/beta* (used for modification). For a more complete discussion on the design exemplar see [24]. One sub-graph (*match*) of the exemplar corresponds to the entities and relations that are explicitly stored in the model. In a B-Rep model, this may often consist of the entities related by the boundary relations and other relations that the designer may have explicitly imposed on 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 and allow for modification of models from the alpha state to the beta state.

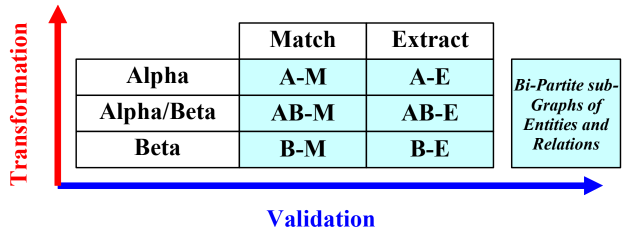


Fig. 4. Components of the Design Exemplar (based upon [24]).

A more sophisticated illustration of an exemplar showing more utility represents a design guideline stating that tapers from a parting line should be used to improve the quality of cast parts. Fig. 5 shows a simple part model, with a parting line (plane) without tapers. Based on the guideline, this part would be improved as seen in Fig. 7 where two of the four possible surfaces that intersect the split line are tapered. To do so, the first step is to determine the possible faces that should be tapered. Fig. 6 shows a design exemplar in text format written to find the surfaces that should be tapered based on the design guideline. These surfaces are identified as intersecting the parting line (plane) as the exemplar matches in the CAD model an identified parting line, a plane, a line bounding that plane, and two points bounding that line. This information is then used to determine:

(1) the angle between the matched split-line plane and the plane bounding the solid model,
(2) a point incident on both the split-line plane and the matched line, and
(3) the distance from the extracted point to each of the two vertices.

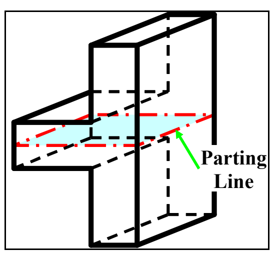


Fig. 5. Model Without Tapers.

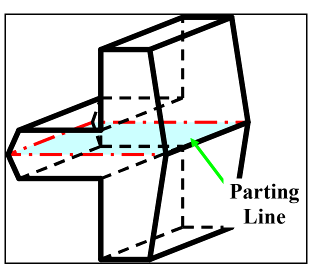


Fig. 7. Model With Tapers.

Match:
Solid "Body";
Plane "Splitting Surface";
Plane "Surface";
Line "Curve 1";
Point "Point 1";
Point "Point 2";
***ID* "Split-Line Plane" (Splitting Surface);**
***Boundary* (Body, {Surface});**
***Boundary* (Surface, {Curve});**
***Boundary* (Curve, {Point 1, Point 2});**
Extract:
Parameter "angle";
Parameter "dist_1";
Parameter "dist_2";
Parameter "dist_3";
Point "Point 3";
***Incident* (Point 3, Curve 1);**
***Incident* (Point 3, Splitting Surface);**
***Angle* (angle, Splitting Surface, Surface);**
***Distance* (dist_1, Point 1, Point 3);**
***Distance* (dist_2, Point 2, Point 3);**
***Distance* (dist_3, Point 1, Point 2);**
***Equation* "eq_1" (angle);**
***Equation* "eq_2" (angle);**
***Equation* "eq_3" (dist_1, dist_2, dist_3);**

eq_1: angle < 91 degrees
eq_2: angle > 89 degrees
eq_3: dist_1 + dist_2 = dist_3

Fig. 6. "Casting Taper Rule" Exemplar.

As these examples illustrate, the design exemplar was first introduced as a declarative representation for geometric problems based on the basic observation that all parametric and geometry related queries about engineering designs seem to deal with characteristics of engineering significance [6]. These characteristics, derived from a pattern of entities and constraints found explicitly or implicitly in the design model, are represented in design exemplars as patterns of topologic, geometric, and algebraic relationships. Exemplars have been used in feature recognition systems, to model standard design procedures, for manufacturing rule validation, and for view transformation. An open question for investigation in this area is the scalability of the concept to handle more complex problems than illustrated here. This question is addressed by creating networks of design exemplars as discussed in [23].

5. EVALUATING THE DESIGN EXEMPLAR

While querying, designers might be looking merely for pattern matches to their specified queries or models that satisfy the conditions specified in the extract part of the exemplar, in addition to the pattern match. They might also want to modify/add/delete information to an existing model. Tab. 3 illustrates the pairing between the different tasks expected of a query language with the sub graphs of the design exemplar. For the retrieval task, only the pattern match may be sufficient and more sophisticated queries may be formed with the alpha extract sub graph along with the alpha match. The modification, addition and deletion tasks essentially center on locating the characteristic to be modified and then transforming to the final desired state. These require the alpha and beta sub graphs, with the alpha sub graph locating the characteristic to be modified and the beta sub graph representing the final state desired.

Query language task	*Alpha*		*Beta*	
	Match	*Extract*	*Match*	*Extract*
Retrieval	Required	Optional		
Modification	Required	Optional	Required	Optional
Addition	Required	Optional	Required	Optional
Deletion	Required	Optional	Required	Optional

Tab. 3. Query Language Tasks vs. Exemplar Sub-Graphs.

The various aspects of a query language were discussed in Section 2. 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 [11].

The exemplar is investigated against the de-facto query SQL. Tab. 4 shows the components and tasks performed by the exemplar as they relate to the qualifications outlined. The table shows representative data-types and predicates.

	Qualifications of a query language	*Design Exemplar*
Components	Data-types	Real parameter, Integer parameter, Vector, Rotation Matrix (Algebraic), Point, Direction, Line, Plane, Circle, Ellipse, Cylinder, Sphere (Geometric), Solid volume (Topologic), Form Features, Part, Assembly (Semantic)
	Predicates	Scalar equations, Scalar inequalities, Fixed Tables, Vector equation, Cross Product(Algebraic) Distance Angle Radius, Focal Distance, Distance to resolved geometry, Control points, Knot values, Continuity conditions, In_Set, Map Coincident ,Incident Parallel ,Right Angle(Geometric) Boundary, Length, Area, Volume, Directed-Left-Of, Curve Direction, Curve Direction TC, Surface Normal, Surface Normal TC, Same Direction(Topologic)
	Logical Connectives	AND, OR, NOT
Tasks	Retrieval	Pattern Matching (Alpha/Match) Query Extraction (Alpha/Match and Alpha/Extract) Design Validation (Alpha/Match and Alpha/Extract)
	Modification, Addition, Deletion	Model Modification (Alpha/Match, Alpha/Extract, Beta/Match, Beta/Extract)

Tab. 4. Query Language Qualifications vs. Design Exemplar.

The design exemplar is also shown to satisfy much of the requirements for a spatial query language as specified by Egenhofer [11], as shown in Tab. 4.

Requirements of a spatial query language (based on [11])	*Does exemplar comply?*
Ability to treat spatial data at a level independent from internal coding such as x-y co-ordinates.	Yes
Display results in graphical form	Yes
Display of context in addition to information sought	Yes
Combine one query result with results of one or more previous queries.	Yes
Labels to aid understanding of models so that users are able to select specific instances of objects.	Limited
Extended dialog allowing selection by pointing and direct selection of a result as a reference to an upcoming query.	No

Tab. 5. Requirements of a Spatial Query Language vs. the Design Exemplar.

First, the design exemplar can be used to query models for geometric, topologic, and algebraic characteristics of interest in engineering design solid models, as shown in [24]. For example, if one wants to find a circle in a model, an exemplar can be authored to find all the circles irrespective of the position of the circle with respect to a global coordinate system. Further, the query can be refined to identify all circles with radii that are within a specific range. With respect to the second requirement, the results of the exemplar query are highlighted in the model and hence it can be argued that the results are displayed in a graphical form (see Fig. 8). It can be further seen that the region identified as a thin wall is displayed within the context of the complete design model, thus satisfying the third requirement about displaying the context of the query. Egenhofer suggest that query results should be able to be combined with results from other queries. A case-based design system, based upon the design exemplar, demonstrates how results from queries may be combined against sets of design models, allowing the user to query results from previous queries [22]. For operating upon a single design model, exemplar queries may combined through exemplar networks or through logical connectives [10, 23]. The fifth requirement is related to labeling objects in the model. This is manually supported in the design exemplar through the "ID Relation" that may be added to models, used to match specific entities for the query, or used to select matched entities to highlight. A design exemplar for identifying casting undercuts may use a label to select the parting plane.

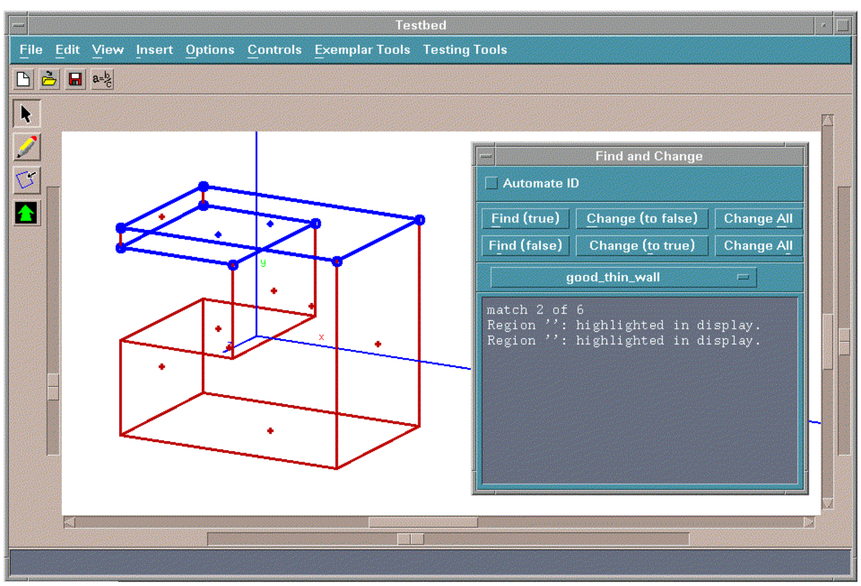


Fig. 8. Thin Wall Characteristic Highlighted in Design Model after Exemplar Query.

Finally, the idea of extending a dialog to allow selection by pointing and direct selection of a result as a reference to an upcoming query is not supported in the design exemplar. While there are tools for creating exemplars automatically by selecting regions in a design model, this functionality does not meet the requirements of Egenhofer.

The design exemplar may also be evaluated against query systems and languages that have been proffered in the literature for geometric interrogation, as seen in Tab. 6. Querying absolute positions and attributes of entities in the model is not explicitly supported by the design exemplar. However, the positions can be derived relative to the other entities and relations in the model. Thus, if a coordinate system is explicitly illustrated in the design model, the relative locations of retrieved entities can be determined. Another example task that is performed by systems detailed in the literature is that of interrogating a solid model for features that may be machined from a specific direction. In order to do this, a certain degree of preprocessing of the model is required by specifying the axis directions and labeling them to support the model interrogation. The design exemplar certainly can iterate through a database of CAD models and find all the potential matches for features with specified relations, as evidenced with the development of the exemplar facilitated case based reasoning system [22]. With respect to a fifth activity, the design exemplar supports region queries but support for part quantities is limited at this point in time. The design exemplar mechanism returns a list of the matched regions with a count of the number of matches made. This information is often duplicated if there is symmetry in the design exemplar (*plane A* parallel to *plane B* is the same as *plane B* parallel to *plane A*). Similarly, the design exemplar can retrieve files from a database [22], but the design exemplar itself does not include mechanisms for file management. External support using the design exemplar as a query basis can be developed to support file management based upon retrieval results. Finally, although the exemplar supports queries using volume relations to extract and define the volume of solid models, support for collision query and clearance query has not been implemented yet. Collision detection and clearance queries are dependent upon extending the definition of the "INCIDENT" relation to include solids, in addition to points, curves, and surfaces. While the design exemplar does not support completely all these typical tasks performed, it does demonstrate in a single system and approach the flexibility to address at least partially all these tasks. While this is not a comprehensive list of tasks performed by geometric query systems and languages, it is representative, showing the interrogative strengths of the design exemplar.

Query System or Language	*Typical tasks performed*	*Can exemplar do tasks?*
[7]	Querying positions and attributes of entities in the map.	Limited
[4]	Relative/ Absolute positions of entities	Limited
[21]	Querying a CAD model for features that can be machined from a certain direction	Yes
[17]	Querying a database of CAD models for features with specified relations	Yes
[19]	Region queries and part quantities	Limited
[14]	File management on STEP files	Limited
[15]	Volume Query; Collision Query; Clearance Query	Limited

Tab. 6. Comparison of Design Exemplar with Other Query Languages/Systems.

Finally, with respect to the design exemplar matching the characteristics of a standard query language, such as SQL, it has been shown that the design exemplar has the data types and the predicates that may be expected of a CAD query language. Further, the design exemplar is supported further with the inclusion of logical connectives, such as NOT and OR blocks [10]. Finally, the design exemplar is capable of querying explicit information (pattern matching), implicit information (query extraction), and adding, deleting, and updating information (model modification). These are facilitated through the validation and transformation axes of the design exemplar, illustrated in Fig. 4 and Tab. 3 and demonstrated with examples found in the literature (e. g. [22, 24]).

6. DISCUSSION

The design exemplar is essentially a representation that has been developed initially to represent geometric and parametric design problems. This representation, however, also offers CAD users a powerful tool for developing geometric queries, thus serving as a foundation for a CAD query language. As mentioned above the design exemplar does comply with all the tasks required by a CAD query language and compares well with a structured query language. Using this representation schema, engineers may interactively define queries against their design models incorporating geometric, topologic, semantic, and algebraic entities and relations. They can specify both the explicit and implicit desired characteristics. Similar to a structured query language, the design exemplar supports the tasks of retrieving and modifying the retrieved records or models.

With two types of design exemplar users identified, CAD modelers and design automation developers, the design exemplar has the potential to improve model development efficiency and to provide engineers, rather than software developers, the capabilities to develop their own design automation systems. This potential will only be fully realized through the continued application and use of the design exemplar in academia and industry. As a first step towards this demonstration, the design exemplar, as illustrated in the motivating example of the tire insert retrieval of Fig. 1, is being used as a basis for developing a commercial CAD system to retrieve tire mold inserts. Future work will continue to demonstrate the potential of the design exemplar and illustrate it as a true and complete CAD query language.

7. REFERENCES

[1] 3D Object Recognition, http://www.lems.brown.edu/~cmc/3DRecog/overview.html, Brown University, 2002, accessed: 2002.

[2] 3D Shape Retrieval Engine, *url*, http://www.cs.uu.nl/centers/give/imaging/3Drecog/3Dmatching.html, Universiteit Utrecht, 2002, accessed: May 24, 2004.

[3] Geometric Search Engine, *url*, http://www.sandia.gov/src/Working_with_Us/Organization_Chart/IS_Principles/Geometric_Search_Engine/geometric_search_engine.html, Sandia National Laboratories, 2002, accessed: May 24, 2004, 2004.

[4] PostgreSQL, *url*, http://techdocs.postgresql.org, PostgreSQL, 2002, accessed: May 24, 2004.

[5] Berchtold, S., Keim, D. and Kriegel, H., Using Extended Feature Objects for Partial Similarity Retrieval, *VLDB Journal*, Vol. 6, No. 4, 1997, pp. 333-48.

[6] Bettig, B., Shah, J. J. and Summers, J. D., Geometric Exemplars: A Bridge Between AI and CAD, in *From Knowledge Intensive CAD to Knowledge Intensive Engineering*, Cugini and Wozny (eds), Kluwer Academic Press, the Netherlands, 2001.

[7] Chan, E. and Zhu, R., QL/G: A Query Language for Geometric Data Bases, *1st International Conference on GIS in Urban Regional and Environmental Planning*, Samos, Greece, 1996, 271-86.

[8] Corney, J., ShapeSearch.net, *url*, http://www.hw.ac.uk/mecWWW/research/Dsearch/3Dsearch.htm, Heriot-Watt University, 2002, accessed: May 24, 2004, 2004.

[9] Divekar, A. and Summers, J. D., The Design Exemplar: A Foundation for a CAD Query Language, *Design Engineering Technical Conferences*, ASME, Vol. DETC-2003**,** Chicago, IL, 2003, CIE-48228.

[10] Divekar, A. and Summers, J. D., Logical Connectives for a CAD Query Language: Algorithms and Verification, *Design Engineering Technical Conferences*, ASME, Vol. DETC-2004**,** Salt Lake City, UT, 2004, CIE-57787.

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

[12] Hursch, C. and Hursch, J., *SQL Structured Query Language*, Windcrest, Inc., Blue Ridge Summit, PA, 1991.

[13] Ibm, T. R. L., 3D Geometry Search Technology, *url*, http://www.trl.ibm.com/projects/3dweb/SimSearch_e.htm, IBM, 1999, accessed: April 28, 2004, 2004.

[14] Koonce, D., Huang, J. and Judd, R., EQL: An Express Query Language, *Computers in Engineering*, Vol. 35, No. 1-2, 1998, pp. 271-4.

[15] Kriegel, H., Muller, A., Potke, M. and Seidl, T., DIVE: Database Integration for Virtual Engineering, *Demonstration Proceedings 17th International Conference on Data Engineering (ICDE)*, Heidelberg, Germany, 2001, 15-18.

[16] Mcwherter, D., Peabody, M., Regli, W. C. and Shokoufandeh, A., *Clustering Techniques for Databases of CAD Models*, Drexel University, Department of Mathematics and Computer Science, 2001.

[17] Ou-Yang, C. and Liou, P., Applying the Topological Relationships of Form Features to Retrieve Part Models from a CAD System, *IIE Transactions*, Vol. 31, 1999, pp. 323-37.

[18] Razdan, A., Rowe, J., Tocheri, M. and Sweitzer, W., Adding Semantics to 3D Digital Libraries, *International Conference on Digital Libraries*, Vol. 5th**,** Singapore, 2002,

[19] Rosenthal, A., Heiler, S. and Manola, F., An Example of Knowledge Based Query Processing in a CAD/CAM DBMW, *Proceedings of the 10th International Conference on Very Large Data Bases*, Vol. 1984**,** 1984, 363-70.

[20] Shah, J. J., Anderson, D., Kim, Y. S. and Joshi, S., A Discourse on Geometric Feature Recognition from CAD Models, *Journal of Computing and Information Science in Engineering*, Vol. 1, No. 1, 2001, pp. 41-51.

[21] Silva, R., Wood, K. and 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, 73.

[22] Summers, J. D., Lacroix, Z. and Shah, J. J., Case-Based Design Facilitated by the Design Exemplar, *International Conference on Artificial Intelligence in Design*, Kluwer Academic Press, Vol. 7, Cambridge, UK, 2002, pp. 453-476.

[23] Summers, J. D. and Shah, J. J., Exemplar Networks: Extensions of the Design Exemplar, *Design Engineering Technical Conferences*, Computers in Engineering, Vol. DETC-2004, Salt Lake City, UT, 2004, CIE-57786.

[24] Summers, J. D., Shah, J. J. and Bettig, B., The Design Exemplar: A New Data Structure for Embodiment Design Automation, *Journal of Mechanical Design*, Vol. 126, No. 5, 2004, pp. 775-87.

[25] Venkatamaran, S., Summers, J. D. and Shah, J. J., An Investigation of Integration of Design by Features and Feature Recognition, *Feature Modeling and Advanced Design for Life Cycle Systems*, IFIP, Valenciennes, France, 2001,