8-2008

SIMILARITY METRICS APPLIED TO GRAPH BASED DESIGN MODEL AUTHORING

Srinivasan Anandan
Clemson University, sananda@clemson.edu

SIMILARITY METRICS APPLIED TO GRAPH BASED DESIGN MODEL AUTHORING

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mechanical Engineering

---

by
Srinivasan Anandan
August 2008

Accepted by:
Dr. Joshua D. Summers, Committee Chair
Dr. Georges Fadel
Dr. John Ziegert
Dr. Brian Malloy
Dr. Mary Beth Kurz

ABSTRACT

Model reuse is typically facilitated by search and retrieval tools, matching the sought model with models in a database. This research aims at providing similar assistance to users authoring design exemplars, a data structure to represent parametric and geometric design problems. The design exemplar represents design problems in the form of a bi-partite graph consisting of entities and relations. Authoring design exemplars for relatively complex design problems can be time consuming and error prone. This forms the motivation of developing a search and retrieval tool, capable of retrieving exemplars that are similar to the exemplar that a user is trying to author, from a database of previously authored exemplars.

In order to develop such a tool, similarity measures have been developed to evaluate the similarity between the exemplar that a user is trying to author and target exemplars in the database. Two exemplars can be considered similar based on the number and types of entities and relations shared by them. However, exemplars meant for the same purpose can be authored using different entities and relations. Hence, the two main challenges in developing a search and retrieval tool are to evaluate the similarity between exemplars based on structure and semantics.

In this research, four distinct similarity metrics are developed to evaluate the structural similarity between exemplars for exemplar retrieval: entity similarity, relation similarity, attribute similarity, and graph matching similarity. As well, a thorough

understanding of semantics in engineering design has been developed. Different types of semantic information found in engineering design have been identified and classified. Design intent and rationale have been proposed as the two main types of semantic information necessary to evaluate the semantic similarity between exemplars. The semantic and structural similarity measures have been implemented as separate modules in an interactive modeling environment. Several experiments have been conducted in order to evaluate the accuracy and effectiveness of the proposed similarity measures. It is found that for most queries, the semantic retrieval module retrieves exemplars that are not retrieved by structural retrieval module and vice versa.

DEDICATION

I dedicate this work to my parents, Mr. Anandan Ananthasubramaniam and Mrs. Girija Anandan. I would not have persevered for all these years had it not been for their unconditional love and support.

ACKNOWLEDGEMENTS

I would like to thank my research advisor, Dr. Joshua D. Summers for his assistance and guidance throughout my study at Clemson University. His depth of knowledge and experience has greatly benefited my research. I am very grateful for the role he has played in my education.

I would also like to thank Dr. Mary Beth Kurz for the motivation and new insights she gave me in my research.

I would like to thank Sudhakar Teegavarapu, Vikram Bapat, and Shashidhar Putti for their assistance in carrying out experiments for my research. I am also thankful to my colleagues Ajit Singh Kanda, Prabhu Shankar, and Beshoy Morkos who made the AID Lab an exciting and fun environment to work in.

Finally, I am grateful to my committee members Dr. Georges Fadel, Dr. Brian Malloy, and Dr. John Ziegert for providing valuable scientific comments in the overall improvement of the research presented in this dissertation.

TABLE OF CONTENTS

Page

Table of Contents(Continued)

Table of Contents(Continued)

Table of Contents (Continued)

Table of Contents (Continued)

Table of Contents (Continued)

LIST OF FIGURES

List of Figures (Continued)

Figure Page

List of Figures (Continued)

List of Figures (Continued)

Figure Page

LIST OF TABLES

List of Tables (Continued)

List of Tables (Continued)

List of Tables (Continued)

List of Tables (Continued)

Chapter 1

INTRODUCTION

In engineering design, as with many other disciplines, knowledge or model reuse is often sought to reduce workload and development effort. Model reuse is typically facilitated by search and retrieval tools, often matching sought models based upon "similarity metrics" [1-3]. For example, engineers may want to retrieve components that are geometrically similar or similar in terms of material used or similar in terms of the manufacturing process followed to produce the component. This can be illustrated by examples of model retrieval problems illustrated below.

1.1. Research Motivation

Three distinct examples are drawn from contemporary research conducted at Clemson University in the Automation in Design (AID) group. The first, relates to an industry sponsored project where the objective is to retrieve components that are geometrically similar. This work is reported in [4].

1.1.1. Structural Retrieval Problem

A large tire manufacturing firm in North America designs mold inserts for its tire treads where the tooling cost for each mold insert is approximately $2,000. Hence, in order to reduce manufacturing cost, the tread designers would like to retrieve geometrically similar inserts from a database of existing mold inserts. An example of

such a target mold insert is shown in Figure 1.1. In order to find mold inserts that can be reused, just the projection is considered (Figure 1.2). As can be seen from the figure, the query mold insert in this case has a line-arc-line top-view. It is desired to find all inserts that fall within a tolerance envelope of the query projection.

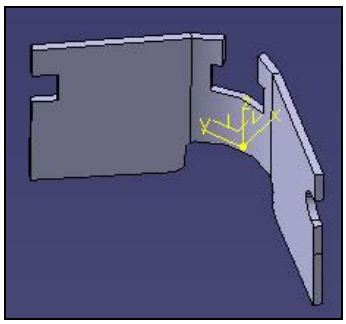


Figure 1.1: Tire mold insert

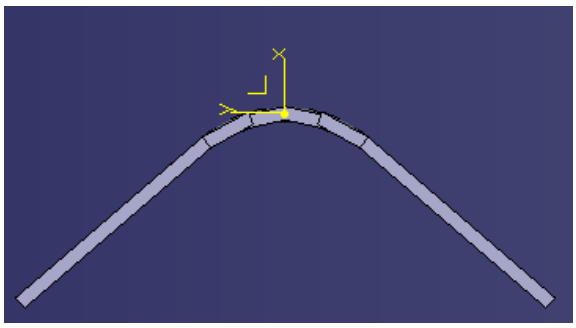


Figure 1.2: Projection of Tire mold insert

Figure 1.3 shows the query mold insert in green with the tolerance envelope in blue. Figure 1.4 shows an example of a mold insert in red that lies within the tolerance envelope and can be used in place of the target mold insert. Figure 1.5 shows a mold insert that does not fit within the tolerance envelope and hence cannot be used in place of the query mold insert.

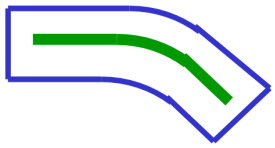


Figure 1.3: Query Mold Insert

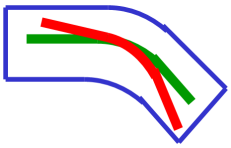


Figure 1.4: Valid Mold Insert from database

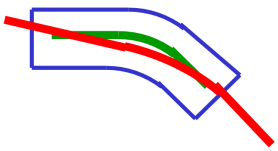


Figure 1.5: Invalid Mold Insert from database

To simplify the retrieval problem parametric models have been developed to represent the inserts as lines and arcs. An algorithm has been developed to first generate a bounding envelope for the query and then evaluate the target inserts' fitness within this envelope. Specifically, the algorithm evaluates parameters such as the minimum and maximum radius of the arc, the minimum and maximum length of the legs, and the distance between the end points of the leg, that target mold inserts can take in order to fit inside the tolerance envelope.

The primary challenge in this example is to first determine feasibility and then assess fitness by comparing two geometric models. This type of problem for matching similar geometries is common in industry, though seldom is it automated. Some examples of academic systems to geometric similarity retrieval are described in [5, 6].

1.1.2. Parametric Retrieval problem

The second motivating example relates to an industry sponsored project where the objective is to retrieve components that are similar with respect to certain parameters. This work is reported in [7]. A frame manufacturing company designs frames for transporting different types of vehicles such as ATVs, jet skis and mowers (Figure 1.6). Currently the frame design is done by a team of two people with vast specialized experience on frame fabrication though are not degreed engineers. The time required to design and make prototypes for the frames is approximately six months. Hence, in order to save on the time spent to design a frame, the designers would like to retrieve frames

that are similar to the desired frame with respect to criteria such as loading points, width of tire, wheel base, weight of vehicle, or vehicle type. The frame that best satisfies most of the requirements can be selected and adapted by making modifications to satisfy all the new design requirements.

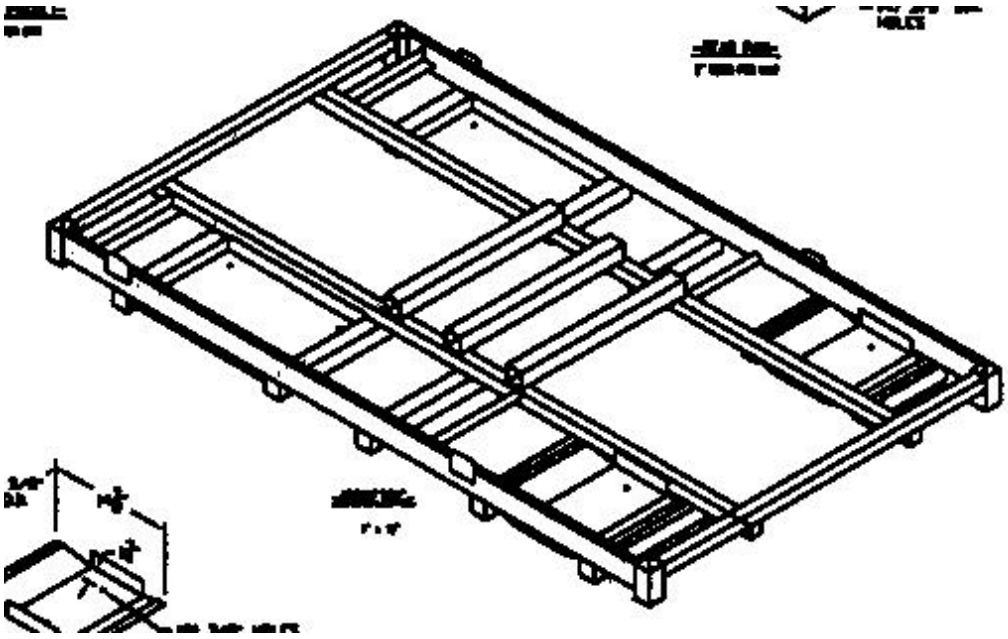


Figure 1.6: Base Frame for Honda

In this problem, the information that the frames are retrieved against is primarily parametric rather than geometric. In both the examples illustrated above, it is evident that there is a need to develop ways to evaluate the similarity between desired shape geometries of inserts and parametric vectors of frame requirements. In both cases similarity can be evaluated from different perspectives. Mold inserts are manufactured in

a two step process. The inserts are first stamped and then bent into a line-arc-line, or a line-arc-line-arc-line model etc. Thus the mold insert shown in Figure 1.1 can be considered similar to other mold inserts in terms of the shape of the stamped part, or it could be considered similar to other mold inserts in terms of the final geometry such as radius of curvature, the angle between the two lines, etc. Similarly, two transport frames can be considered similar to each other if they satisfy the same parameters. However, two frames designed to satisfy two completely different parameters may be considered similar in terms of structure. It may be possible to use such frames for the new design problems. Hence in this case as well, similarity can be evaluated from two perspectives; parameter and structure.

1.1.3. Combined Retrieval Problem

The third example combines both parametric and structural information in a common representation. Further, it is the motivating example that serves as the demonstration platform for this research. The design exemplar provides a standard representation of mechanical engineering design problem knowledge based upon a canonically derived set of entities and relationships [8]. Canonical implies a well accepted list, and not an exhaustive set of entities and relations [9]. These entities and relationships are represented in a bi-partite graph that captures geometric, topologic, or parametric characteristics in a design problem. Users of the design exemplar create these graphs to interrogate (query) or modify (transform) computer aided design (CAD)

models.

For illustration purposes, a simple exemplar to determine the distance between two planes is shown in Figure 1.7. The exemplar consists of three entities and three relations. The two planes are related through both the distance and parallel relations. The value of the distance is stored in the distance parameter. The id relation related to the distance parameter helps display the value of the distance. The planes are matched to planes in the target CAD model and the parallel relation is checked for satisfaction. If there are two parallel planes matched in the model, then the distance between the planes is extracted and displayed to the user.

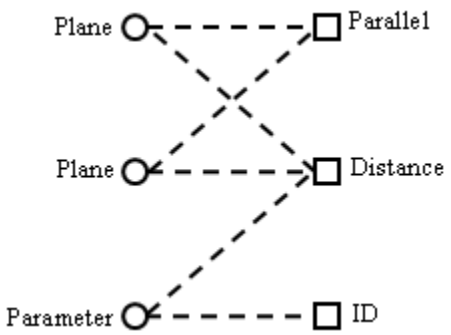


Figure 1.7: Exemplar for finding distance between two planes

The exemplar shown above consists of only three entities and three relations and is relatively easy to author. However, the entire procedure of adding entities and relations can be tedious if a user starts authoring an exemplar for representing a relatively large design problem. For instance, an exemplar for sizing a V-belt [10] could consist of 53 entities and 58 constraints (Figure 1.8). From visual inspection it can be

seen that this exemplar is substantially more complicated. Hence, it might be useful if there were a manner to provide some kind of assistance to the author while authoring such exemplars.

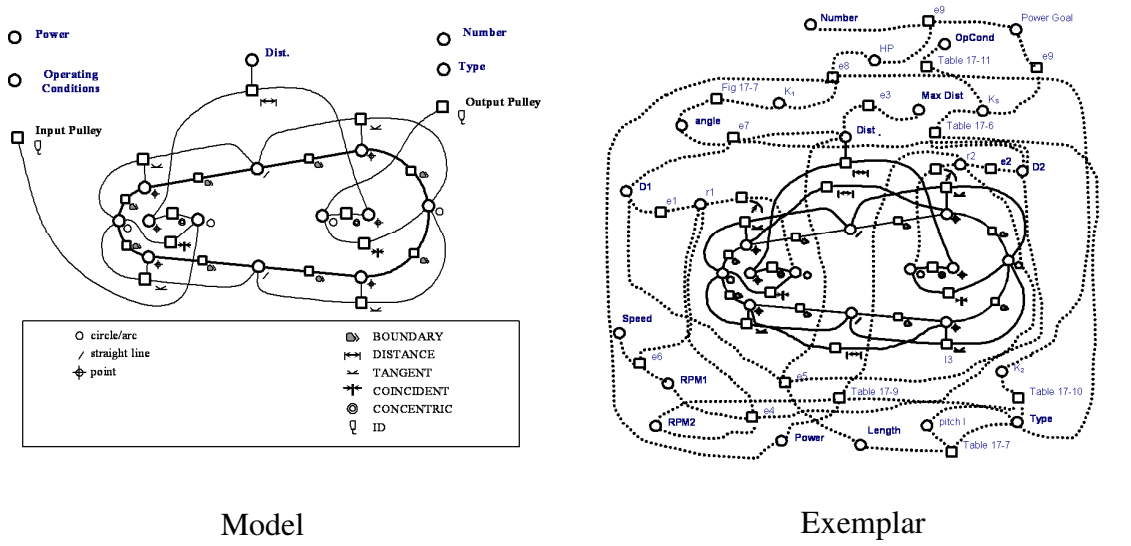


Figure 1.8: Exemplar for V-belt

An analogous tool may be the word completion function found in word processors such as Microsoft Word [11], when a person wants to type the word ‘December’. The moment the author types the letter sequence ‘dece’, a pop-up window appears on the screen that says ‘December’. On hitting the return key, the word ‘December’ replaces the ‘dece’ letter string in the document. This dissertation research is aimed at providing similar assistance in authoring exemplars for large design problems.

As noted before, authoring an exemplar for a large design problem can take a long time if starting from scratch. Instead, it would be useful if the author is allowed to make

changes to an existing exemplar to achieve their own new goals. It may be possible for exemplar authors to study all exemplars that are present in a database if the size of the database is not large. Conversely, the process of manually searching through the database for similar exemplars could be time consuming if the database is large. Hence, it would be useful to facilitate automatic browsing of the database of exemplars and offer alternative possible configurations based upon what has been authored thus far.

The rationale for developing this tool is analogous to variant design, which refers to the technique of adapting existing design specifications to satisfy new design goals and constraints [12]. The tool that is envisioned will help the author in authoring exemplars by providing him with examples of similar exemplars that have been authored earlier. From the options provided, the author can choose the exemplar that is most useful and modify it in order to satisfy the new requirements. Similar to the two retrieval problems mentioned above, exemplars can be considered similar from different perspectives. Since exemplars meant for a specific purpose can be authored in more than one way, structurally different exemplars can be considered similar in terms of meaning. As well, exemplars that are meant for different purposes, but which may consist of similar entities and relations, can be structurally similar.

1.2. Dissertation Overview

Thus, the prime objective of this research is to develop an understanding of similarity in engineering design, specifically looking at structural and semantic similarity

as it relates to the design exemplar. In order to develop an understanding of the design exemplar and the various aspects of the exemplar retrieval tool, the next chapter describes background information of the design exemplar. Chapter 3 explains the research questions that need to be answered in order to develop an effective retrieval tool. Chapter 4 explains the different measures of structural similarity developed in literature and the measures that have been proposed in order to evaluate the structural similarity between exemplars. Chapter 5 aims to develop an understanding of semantics in engineering design. The various representation schemes that have been developed in literature to represent semantics have been discussed in Chapter 6. Chapter 7 discusses the implementation of a dual retrieval system in order to retrieve exemplars that are both structurally and semantically similar to the query exemplar whereas chapters 8, 9, and 10 discuss the different experiments that were conducted to study and explore the idea of a dual retrieval system. Chapter 11 discusses the experiments conducted in order to evaluate the robustness of the dual retrieval system. The conclusions of this research, the contributions made towards other areas of engineering design, and future work is discussed in Chapter 12.

Chapter 2

THE DESIGN EXEMPLAR

Most queries about engineering designs related to parameter and geometry deal with some characteristic of interest for the designer [13]. This forms the motivation for the introduction of design exemplars to computer-aided design. The design exemplar is a data structure that has been developed to represent parametric and geometric design problems [8]. Entities and their relationships found explicitly or implicitly in the design artifact describe these characteristics. Design exemplars represent these characteristics as patterns of topologic (such as boundary constraints, edges, faces, etc.), geometric, algebraic, and semantic relationships. The design exemplar is composed of one bi-partite graph that is partitioned in two ways: match/extract partition (used for validation) and alpha/alpha_beta/beta partition (used for transformation). Bi-partite graphs are defined as graphs that may be divided into two distinct groups of nodes where each edge joins a node from the first group and a node from the second group [14]. There are no edges that join two nodes from the same group. The orthogonality between the two partitions is found between the two axes of operation for design exemplars [8] transforming the design model from one characteristic to another characteristic and validating that a characteristic in the design model exists. An instance of an entity or a relationship may exist in only one of these six possible combined partitions. Connections (edges) between the graph nodes (entities or relationships) may connect nodes from two different

partitions. One partition of the exemplar (the match part) consists of entities and relationships that are to be identified in the design model, which exist explicitly in it. The other partition (the extract part) shows relationships that must hold true in order for the characteristic to be true of the matched part of the design. The transformation axis of the exemplar represents the alpha and beta sub graphs of the exemplar and allow for modification of models from alpha state to the beta state. 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. They might also want to modify/add/delete information to an existing model.

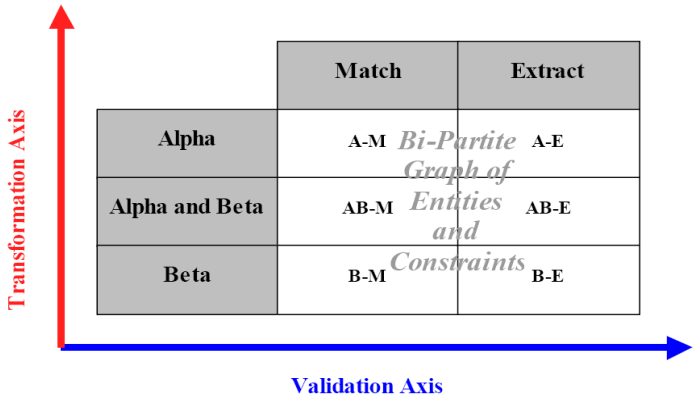


Figure 2.1: Components of an exemplar (adapted from [8])

These aspects of the design exemplar can be illustrates with the help of an example. Let us consider the exemplar of finding all lines less than one unit long, in the 2-D model shown in Figure 2.2. One way of authoring such an exemplar is as shown in Figure 2.3. The boundary relation binds the points “A” and “B” and the line “L”. These

entities and relations should exist in the model being queried. The distance relation finds the distance between the two points and the value is stored in the distance parameter. The equation relation checks whether the distance is less than one and the ID relation highlights those lines that are less than one unit long. The points A, B and the line L are alpha matches whereas all the other relations and entities are alpha extracts. Match entities and relations are illustrated with solid lines while extract entities and relations are represented with dashed lines. In this case, only the alpha characteristic is sought and no transformation is made to the model. Therefore, beta and alpha_beta are not included in this exemplar.

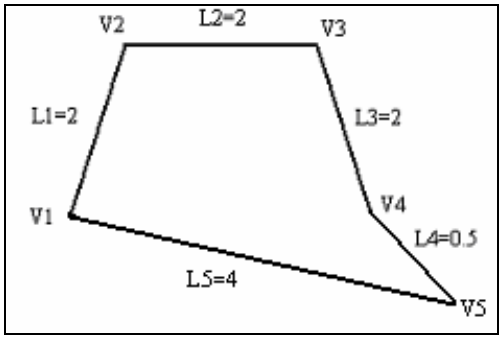


Figure 2.2: 2-D Model

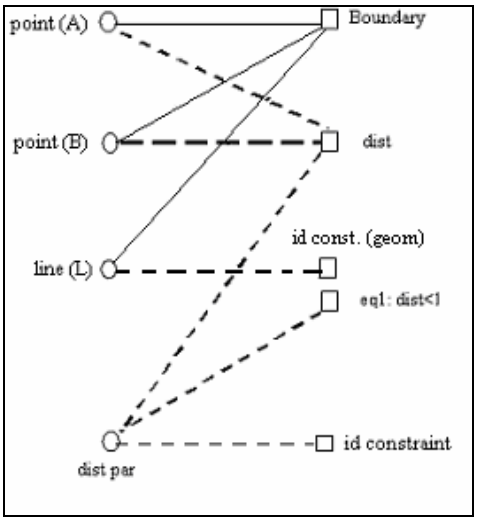


Figure 2.3: Exemplar for finding lines less than one unit long

Alpha Match:

Line, "L";
Point, "A";
Point, "B";
Boundary (L, A, B);

Alpha Extract:

Distance **(A, B);**
Parameter **(dist par)**
Equation **(dist par < 1);**
Id **(dist par);**
Id **(L);**

Figure 2.4: Textual representation of exemplar shown in figure 10

Since line L4 satisfies the problem, it will be highlighted when the above exemplar is applied to the model. The table of the matched entities and extracts is shown below (Table 2.1).

Table 2.1: Results obtained from applying exemplar (Figure 2.3) to model (Figure 2.2)

M = Match; E = Extract; T = True; F = False					
M	M	M	E	E	E
A	B	L	Dist Par	Dist	Eq1: dist < 1
V1	V2	L1	2	T	F
V2	V1	L1	2	T	F
V2	V3	L2	2	T	F
V3	V2	L2	2	T	F
V3	V4	L3	2	T	F
V4	V3	L3	2	T	F
V4	V5	L4	0.5	T	T
V5	V4	L4	0.5	T	T
V5	V1	L5	4	T	F
V1	V5	L5	4	T	F

As shown in the table, the authored exemplar when applied to the model will return 10 possible matches. As shown, although the pairs (V1, V2) and (V2, V1) are symmetric, they are treated differently by the exemplar. The distance parameter gives the value of the distance between each pair of points. Since L4 is the only line with distance less than one, the pairs (V4, V5) and (V5, V4) are the ones, which are highlighted in the table.

The following example illustrates the modification aspect of the design exemplar. As mentioned earlier, the design exemplar can be used to modify geometric and parametric features of a CAD model. Let us consider an exemplar to modify the radius of a circle. Figure 2.5 and Figure 2.6 show one way of authoring such an exemplar.

```
Match (Alpha and Beta)
        Circle "C1";
Extract (Alpha and Beta)
        Parameter "Temp";
Extract (Alpha)
        Parameter "Radius_Alpha";
        Radius (C1, Radius_Alpha);
        Equation "Eq_alpha" (Radius_Alpha, Temp);
Extract (Beta)
        Parameter "Radius_Beta";
        Radius (C1, Radius_Beta);
        Equation "Eq_beta" (Radius_Beta, Temp);
```

Eq_alpha:	Radius_Alpha = Temp
Eq_beta:	Radius_Beta = 2*Temp

Figure 2.5: Textual Representation of exemplar to double the radius of circle

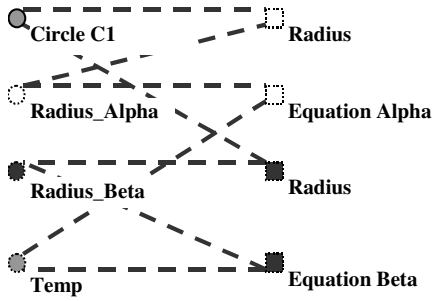


Figure 2.6: Exemplar to double the radius of a circle

The exemplar consists of a circle entity and a radius relation attached to that entity. The radius relation has the extract and alpha attributes since it is present in the model before modification and is not explicit. The radius relation extracts the radius of the circle and stores it in the parameter named 'Radius_Alpha'. This parameter also has the attributes alpha and extract, since it is present in the model only before modification and is not explicit. The exemplar also consists of two equations. The equation 'Eq_alpha equates the parameter 'Radius_Alpha' to the parameter 'temp' and has the attributes alpha and extract. The parameter 'temp' has the attributes alphabeta and extract because

it is not explicit and is present in the model both before and after modification. The equation 'Eq_beta' has the attributes beta and extract since it is present in the model only after modification. This equation doubles the value stored in the parameter 'temp' and stores the doubled value in the parameter 'Radius_Beta'. Similar to the second equation, this parameter also has the attributes beta and extract. This parameter is related to the circle though the 'radius' relation. On application of this exemplar to a model containing a circle, the radius of the circle will be doubled.

It has been proposed that the design exemplar can be used as a CAD query language [15]. The requirements and qualifications of a query language were studied and compared to the capabilities of the design exemplar.

Table 2.2: Requirements of a spatial query language satisfied by the Design Exemplar (adapted from [15])

Requirements of a spatial query language	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
Combine one query result with results of one or more previous queries.	Yes
Display of context in addition to information sought	Yes
Extended dialog allowing selection by pointing and direct selection of a result as a reference to an upcoming query	No
Labels to aid understanding of models so that users are able to select specific instances of objects.	Limited

Table 2.2 shows all the requirements of a query language and whether or not the design exemplar satisfies these requirements. Table 2.3 lists all the qualifications of a query language and the qualifications possessed by the design exemplar. It was concluded from the comparison that the design exemplar does comply with all the tasks required by a CAD query language and compares well with a structured query language.

Table 2.3: Query Language Qualifications possessed by the Design Exemplar (adapted from [15])

	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)

A procedural use of the design exemplar was proposed in 2007 [16]. As part of this research, the use of the design exemplar as a visual programming language was investigated. There are three important components of a visual programming knowledge. They are icons, iconic system, and compiler. It was found that the design exemplar has two of these three components. The design exemplar uses visual objects (icons) to represent geometric, parametric and topologic entities and their relations. This research introduces two new data structures, 'dynamic exemplar node' and 'dynamic network', that facilitate procedural programming with the existing design exemplar. An example of a dynamic exemplar node is shown in Figure 2.7. The research does not address the development of a compiler.

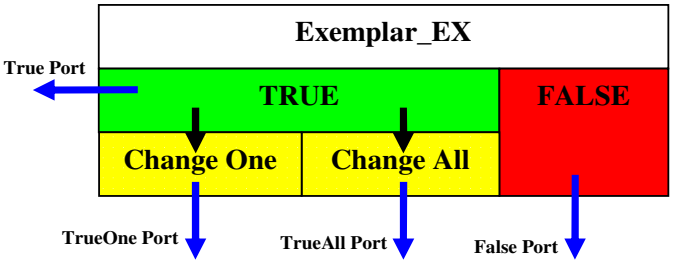


Figure 2.7: Complete exemplar node for dynamic networking

2.1. Aspects of Design Exemplar

In order to develop the exemplar retrieval tool, measures of similarity need to be developed that can be used to quantify the similarity between the exemplars in the database and the exemplar being authored. However there are some aspects of design

exemplars that need to be considered in order to understand how exemplars can be considered similar. These aspects are described below.

2.1.1. Different ways to author an exemplar meant for a specific purpose

There may exist multiple ways of authoring an exemplar meant for a specific purpose. For example, consider an exemplar for finding the distance between two planes. There are two ways of authoring such an exemplar. The exemplar shown in Figure 2.8 consists of two planes with a distance and a parallel relation between the two planes. The two planes form the match portion of the exemplar, since the planes are explicit in the model. The distance and the parallel relations form the extract portion, since these relations are implicit in the model. On applying this exemplar to a CAD model, distance between all sets of parallel planes are displayed. The value of distance between a set of parallel planes is stored in the distance parameter, while the ID relation is necessary to display the value. The second exemplar consists of two planes and two points such that they are incident on the two planes (Figure 2.9). The two points in turn are incident on a line which is perpendicular to one plane. The exemplar also consists of two surface normals from the two planes that are parallel to each other. In this case, the planes are match whereas the lines, the points, the surface normals, the incidence relations, the distance and the parallel relations form the extract portion. Applying this exemplar to the same model will also give the same results as obtained with the first exemplar. Hence, these two exemplars can be considered similar with respect to the intended use.

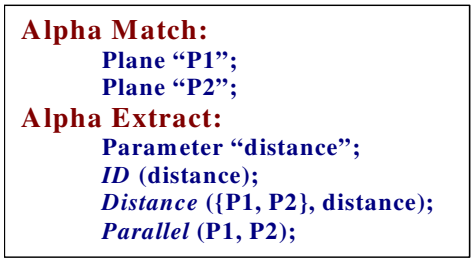


Figure 2.8: Exemplar for finding distance between two planes

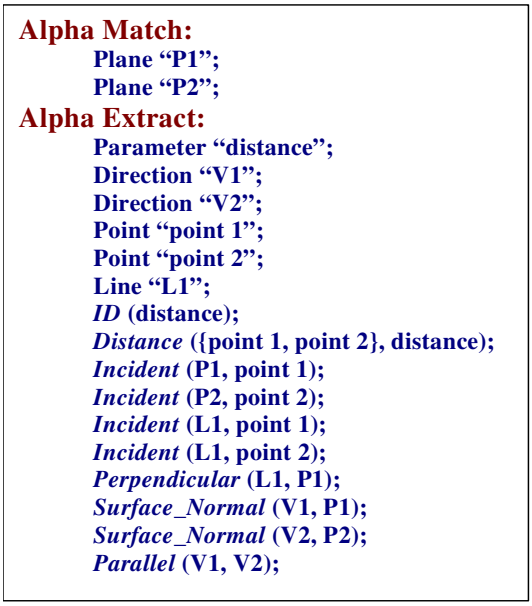


Figure 2.9: Alternate exemplar for finding distance between two planes.

2.1.2. Multiple uses for the same exemplar

An exemplar intended for a specific use could have multiple uses. The entities and relations used in an exemplar could represent different features in a CAD model. For example, a plane could either represent one side of a thin wall or one side of a rectangular boss. A line could either represent a parting line of a component that has been sand-cast, or it could represent an edge of a rib. Hence, it may be possible that an exemplar meant for finding thin walls can be used for finding ribs, or bosses (Figure 2.10). This illustrates that exemplars can be considered similar not only in terms of their structure, but also in terms of meaning or purpose.

```
Alpha Match:
        Solid "R1";
        Plane "R_surf";
        Plane "T_surf";
        Plane "L_surf";
        Line "R_edge";
        Line "L_edge";
        Point "R_start";
        Point "R_end";
        Point "L_start";
        Point "L_end";
        Boundary (R1, {R_surf, T_surf, L_surf});
        Boundary (R_surf, {R_edge});
        Boundary (T_surf, {R_edge});
        Boundary (T_surf, {L_edge});
        Boundary (L_surf, {L_edge});
        Boundary (R_edge, {R_start, R_end});
        Boundary (L_edge, {L_start, L_end});
Alpha Extract:
        Parameter "thickness";
        Parameter "angle";
        Vector "R_dir";
        Vector "L_dir";
        Line "virtual line";
        Point "virtual point";
        Surface_Normal_Topologically_Correct (R_dir, R1, R_surf);
        Surface_Normal_Topologically_Correct (L_dir, R1, L_surf);
        Angle (angle, R_dir, L_dir);
        Distance (thickness, R_surf, L_surf);
        Parallel (R_surf, L_surf);
        Equation "eq_1" (angle > PI - 0.05);
        Equation "eq_2" (angle < PI + 0.05);
        Equation "eq_3" (thickness < 0.1);
        ID (R_surf);
        ID (T_surf);
        ID (L_surf);
        Incident (virtual line, R_start);
        Incident (virtual point, L_edge);
        Incident (virtual line, virtual point);
        Perpendicular (virtual line, R_edge);
        Distance (D1, virtual point, L_start);
        Distance (D2, virtual point, L_end);
        Distance (D3, L_start, L_end);
        Equation "eq_4" (D3 < D1 + D2 + 0.05);
        Equation "eq_5" (D3 > D1 + D2 - 0.05);
```

Figure 2.10: Exemplar for finding thin walls

Based on the above discussion, it can be concluded that the exemplar retrieval system would have to satisfy three specific requirements. These requirements are

summarized below.

a Identify and retrieve exemplars that are structurally similar.

b Identify and retrieve exemplars that may be structurally different but meant for the same purpose.

Chapter 3

RESEARCH QUESTIONS AND RESEARCH TASKS

The discussion presented in Chapter 2 leads to a series of research questions that need to be answered in order to develop an effective exemplar retrieval system. As mentioned earlier, in order to retrieve exemplars from the database it is essential to evaluate the similarity between the query exemplar and the exemplars in the database. A number of solution strategies developed in literature to evaluate the similarity between CAD models have been reviewed and is discussed in detail in Chapter 4. As mentioned before, exemplars are graph-based representation of geometric and parametric information. Hence, concepts such as minimum common supergraph [14], maximum common subgraph [17], and edit distance [18] suggested in graph theory can be used to evaluate the similarity between exemplars.

As well, the different aspects of the design exemplar discussed in the previous chapter suggest that there are multiple ways in which design exemplars can be considered similar. As shown by the exemplars to find the distance between two planes, exemplars can be considered similar on the basis of structure and on the basis of meaning or semantics. This leads to the research questions and hypotheses listed below.

3.1. Research Questions

The first question relates to the nature and meaning of similarity as it relates to the

design exemplar. Specifically, the difference between structural and semantic similarity is of interest. The first research question can be found in Table 3.1

Table 3.1: Research Question 1

Research Question 1	Can similarity metrics be defined for graph-based models?
Hypothesis	Yes, similarity measures such as edit distance, maximum common subgraph and minimum common supergraph can be defined for graph-based models.
Sub-question 1.1	On what basis can graph-based models be considered similar?
Hypothesis 1.1	Graph-based models can be considered similar based on structure.
Hypothesis 1.2	Graph-based models can be considered similar based on semantics.

Having seen in Chapter 2, that exemplars can be considered to be similar based on semantics, the next research question that needs to be answered explores what additional information is necessary in order to represent the meaning or semantics of a design exemplar. In order to answer this question it is necessary to know what is considered semantic information in engineering design. There exist different definitions of semantics in engineering design which will be reviewed in Chapter 5. As well, it is necessary to classify the different types of semantic in engineering design in order to determine which type of semantic information is necessary to facilitate exemplar retrieval. Based on the exemplars illustrated to determine the distance between two planes it is hypothesized that information regarding design intent and rationale for authoring an exemplar in a specific manner is needed in order to retrieve exemplars that are semantically

similar to a query exemplar. The second research question and associated hypotheses and sub-questions are found in Table 3.2.

Table 3.2: Research Question 2

Research Question 2	What kind of semantic information is necessary to define similarity between exemplars?
Hypothesis	Intended use and rationale for authoring an exemplar in a particular manner is necessary to define similarity between exemplars.
Sub-question 2.1	What is semantics in engineering design?
Hypothesis 2.1	Semantics in engineering design is design knowledge needed to facilitate understanding of design.
Sub-question 2.2	What are the different types of information that can be considered semantic in engineering design?
Hypothesis 2.2	Design rationale and intended use can be considered semantic in engineering design.

Knowing the different types of semantic information that need to be associated with design exemplars to facilitate their retrieval, the next step is to determine the different ways in which this information can be represented in a computer. Different knowledge representation schemes have been suggested in literature including description logic, production systems, and semantic networks. The advantages and limitations offered by each of these and their suitability to exemplar retrieval are reviewed and presented in detail in Chapter 6. The research questions and the hypotheses are listed in Table 3.3.

Table 3.3: Research Question 3

Research Question 3	How can semantic information be associated with exemplars?
Hypothesis	Exemplars can be textually annotated with semantic information.
Sub-question 3.1	What are the different ways of representing semantics in engineering design?

3.2. Research Tasks

To answer the research questions discussed above, five research tasks have been undertaken in this research. These tasks are found in Table 3.4.

Table 3.4: Research Tasks

Task 1	Develop metrics of structural similarity.
Task 1.1	Review existing solution strategies for 3D model retrieval and different graph matching concepts.
Task 1.2	Establish measures to evaluate structural similarity between exemplars.
Task 2	Develop semantic similarity measures.
Task 2.1	Review different definitions of similarity in design.
Task 2.2	Study different types of semantic information present in different design documents and classify them.
Task 2.3	Use the classification scheme to identify the kinds of semantic information useful for design exemplar retrieval.
Task 3	Select and extend semantic representation schema
Task 3.1	Review various knowledge representation schemes
Task 3.2	Use textual representation to represent semantic information to facilitate exemplar retrieval
Task 4	Implementation of a conjoined retrieval system
Task 5	Validation and Testing

3.3. Contributions

Apart from contributing toward exemplar technology, the broader impact of this

research will extend across various areas of engineering design such as engineering model reuse, semantics in engineering design, and similarity in engineering design. As mentioned earlier, authoring exemplars for large design problems can be tedious and time consuming. The exemplar retrieval tool will provide assistance to the exemplar author in authoring complex exemplars and thus facilitate authoring exemplars that are not only precise but also less time consuming. The measures of similarity developed as part of this research can be used for graph-based representations in general. For example, the similarity metrics used for retrieving exemplars can be used to evaluate the similarities between graph-based representations such as function structures and process plans. As well, this research forms the motivation for exploring the area of similarity in engineering design. This may lead to development of a method to evaluate the similarity between two designs. This research will also help in achieving a better understanding of engineering design semantics and the ways in which semantics can be represented in a computational environment. Semantically rich CAD tools have the potential to radically impact current practice, providing an integrative environment where multiple facets of a design project can be considered, resulting in reduced time-to-market, fewer design iterations, fewer costly mistakes, and overall improved quality. Specifically, this work will create a tight integration of semantic and geometric information to support common reasoning.

			Research Questions				
	1	1.1	2	2.1	2.2	3	3.1
TASKS							
1.1	x	x					
1.2		x					
2.1			x	x			
2.2			x	x	x		
2.3					x		
3.1						x	x
3.2						x	
4	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x

Figure 3.1: Research Questions and Research Tasks

The research tasks and research questions are arranged in a matrix as shown in Figure 3.1. It can be seen that at least two tasks are carried out in order to answer each research question.

Chapter 4

DEVELOPING MEASURES OF STRUCTURAL SIMILARITY

The first task is to develop measures in order to evaluate the structural similarity between exemplars. In case of exemplar models, entities and the relations between entities determine the structure of the graphs. For example, consider the exemplars shown in Figure 4.1, Figure 4.2, and Figure 4.3. Exemplar 1 (Figure 4.1) consists of two planes, a distance relation between them, a distance parameter and two equation relations. Exemplar 2 (Figure 4.2) has all the entities and relations contained in exemplar 1 except it has only one equation relation. Exemplar 3 (Figure 4.3) also has all the entities and relations contained in 1 but it does not have any equation relation. The number of modifications required to completely transform exemplar 2 into exemplar 1 is one less than the number of steps required to convert exemplar 3 into 1. Hence, in terms of the structure of the graph, exemplar 2 can be considered to be more similar to 1.

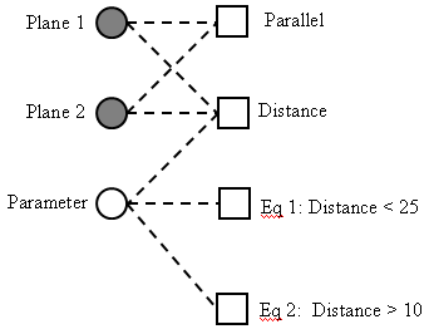


Figure 4.1: Exemplar 1 in database

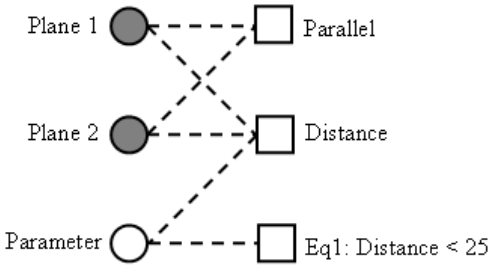


Figure 4.2: Exemplar 2 in database

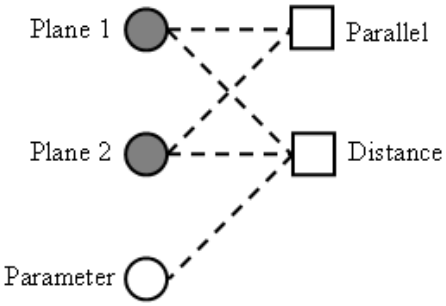


Figure 4.3: Exemplar 3 in database

In order to develop effective structural similarity measures, it is would be useful to review the various solution strategies existing in literature for retrieving CAD models.

4.1. Similarity Strategies

A review of existing 3D shape matching techniques is presented here. Shape matching methods can be broadly divided into two categories; feature based methods and graph-based methods. The similarity between two CAD models can be captured in terms of a distance measure between the two models. The distance between two objects is a measure of the dissimilarity between two objects. Small distances imply less dissimilarity and more similarity. A dissimilarity measure between two 3d models should have some of the following properties; identity, positivity, symmetry, and triangle inequality. Identity states that a 3d model is completely similar to itself. Positivity states that different shapes are never completely similar. Symmetry states that an object A is as similar to object B as B is to A. Triangle inequality states that for three

objects A, B, and C, d (A, B) <= d (A, C) + d (C, B), where d is the distance function.

4.1.1. Feature-Based Systems

Evaluating the similarity between CAD models generally consists of extracting features and feature matching. In case of shape matching features denote the geometric and topological properties of 3D models. Feature-based methods can be further classified as; global features; global feature distribution; spatial maps; and local features.

Global Features: The global shape of a 3D model is characterized by global features such as volume of the model, volume-to-surface ratio, statistical moments, bounding boxes, and fourier transform coefficients [19, 20]. Convex-hull based indices such as hull crumpliness, and hull packing have also been used as global-shape descriptors [21]. Hull crumpliness is defined as the ratio of the 3D model's surface area to the surface area of its convex hull, whereas hull packing is defined as the percentage of the convex hull volume not occupied by the 3D model. These methods characterize only the global shape of the 3D model and can be used as a preliminary filter after which detailed comparisons can be made. All similarity measures based on global features require property extraction or property valuation.

Global feature distribution: One way to look at shape similarity is to compare the distribution of global features rather than the features themselves. Shape distributions measure properties based on distance, angle, area, and volume measurements between random surface points [22]. The similarity between CAD models can be evaluated based

on the distances between the global feature based distributions. Shape descriptors can be based on shape histograms. For example, shape histograms [23] can be based on; 1. the moment of inertia about the axis; 2. average distance from the surface to the axis; 3. variance of the distance from the surface to the axis.

Spatial map based similarity: Spatial maps are used as representations in order to capture the spatial location of a 3D model. Shape-based search methods are used to retrieve 3D geometric models and sketches in which spherical harmonics are used in creating discriminating similarity measures [24]. As part of this approach a 3D model is decomposed into collection of functions defined on concentric spheres. Orientation information is discarded using spherical harmonics. The resulting shape descriptors of models are compared to evaluate the similarity between the models.

Local features: Local feature based methods take into account the surface shape in the vicinity of points on the boundary of the shape. A spherical coordinate system is used to map the surface curvature of 3D objects to a unit sphere [25]. A distance measure is computed by searching over a spherical rotation space and used as a measure of similarity between the objects. 3D shape contexts are applied in order to compute the similarity between 3D models [26]. The shape context of a point is defined as a histogram of the relative co-ordinates of the remaining surface points. Matching consists of a global matching step and a local matching stage. The local matching step for a point on a model consists of finding a best matching point on the other model, whereas the global matching

stage consists of finding correspondences between similar points on the two shapes.

Fingerprint Representations: As an example of a graph-based approach in engineering, the application of model dependency graphs that store machining features has been investigated. These approaches compare the manufacturing plans of solid models in order to evaluate the similarity between the models [1]. An Engineering Advisory System (EAS) has been developed, which involves the retrieval of knowledge associated with 3D models [27]. The system takes a 3D shape as a query and converts it into two simpler "fingerprint representations": feature vectors and skeletal graph. Feature vectors represent the global shape or geometry of the object, whereas the skeletal graphs minimally represent the topology and the local geometry of the object. Search for similar models can be performed through a combination of these two representations. Similarly, curvature distributions of 3D models can be compared to evaluate the similarities between CAD models. This is done by mapping the surface curvature of 3D objects a unit sphere and searching over a spherical rotation space, which in turn gives a distance measure between the curvature distributions.

In case of design exemplars, methods developed to evaluate the similarity between shape geometries are not directly applicable. The main reason for this is that design exemplars are essentially graph-based representations and do not have a specified shape. The nodes and the edges of the graph can be placed anywhere, and two exemplars may have the same entities and relations but look completely different. However, the entities and relations can be considered similar to features in 3D models and feature-

based methods can be adapted to evaluate the similarity between exemplars. The next section discusses graph-matching concepts from graph theory that may be useful to measure the similarity between exemplars.

4.1.2. Graph Matching Concepts

In this research, since exemplars are graph-based models the concept of similarity may be investigated with respect to relational graphs. A relational graph consists of finite number of edges and nodes. A brief summary of the basic concepts of graph matching is provided below based on [28]. A bijective mapping from the nodes of a graph g to the nodes of a graph g^1, which also preserves all labels and the structure of the edges, is defined as a graph isomorphism from a graph g to a graph g^1. Another important concept in graph matching is maximum common subgraph. A maximum common subgraph of two graphs, g and g^1, is defined as a graph g'' that is a subgraph of both g and g^1, which has the maximum number of nodes as compared to all the possible subgraphs of 'g' and 'g^1'. The concept of graph isomorphism is a useful concept to find out if two objects are the same, or if one object is part of another object, or if one object is present in a group of objects.

The concept of the minimum common supergraph of two graphs was introduced in [29]. Intuitively similar to the concept of subgraph, a graph g containing two graphs, g^1 and g'', as subgraphs, is defined as a supergraph of g^1 and g''. The minimum common supergraph of g^1 and g'' is a graph that is a supergraph of both g^1 and g'' and has the

minimum number of nodes as compared to all those supergraphs.

Instead of computing the maximum common subgraph of two graphs, the error-tolerant graph matching can be evaluated using graph edit distance. A graph edit operation can be a deletion, insertion, or substitution. An example of substitution could be a label change. Both nodes and edges can be subjected to edit operations. The shortest sequence of edit operations needed to transform graph g into graph g^1 is defined as the edit distance of the two graphs. The shorter the edit distance, the more similar two graphs are considered to be. Often, there may be costs associated with individual edit operations. Typically an edit operation with a more likely occurrence has a smaller cost associated with it. This association of costs with the individual edit operations is often defined in terms of a cost function. Thus, a simple definition of graph edit distance computation is to find a set of edit operations that convert one graph to another graph with minimum cost.

A common approach of expressing the concept of similarity or dissimilarity is by means of a distance function [30]. If D is a domain, and x and y are two objects belonging to that domain, then the similarity between them can be expressed by dist(x, y). Often, the distance function is assumed to satisfy the three properties of non-negativity, symmetry, and triangular inequality [31]. Graph isomorphism, subgraph isomorphism, and maximum common subgraph detection can be considered to be instances of graph edit distance computation under special cost functions [32]. As well,

weighted graph matching can be regarded a special case of graph edit distance [33, 34].

The concept of graph similarity is independent of the algorithms that have been developed for use in graph matching. Most of the algorithms that have been developed rely on some kind of tree search that make use of various heuristic look-ahead techniques in order to narrow down the search space. A standard algorithm for graph and subgraph isomorphism detection is the one by Ullman [35]. The problem of maximum common subgraph (MCS) detection has been addressed in [36-38]. In [36] a MCS algorithm that uses a backtrack search is introduced. A different strategy for deriving the MCS first obtains the association graph of the two given graphs and then detects the maximum clique (MC) of the latter graph [37]. The work in [38] is centered on formulating the maximum clique problem in terms of a continuous optimization problem. A class of continuous and discrete time "replicator" dynamic systems is developed in evolutionary game theory and it is shown how they can be employed in order to solve the relational matching problem.

Classical methods for error-tolerant graph matching can be found in [39-41]. In [39] the graph distance measures are grouped into two categories. First, Feature-Based Distance is where a set of features is extracted from the structural representation. These features are then used as an n-dimensional vector where the Euclidean distance can be applied. The second, cost-based distances is where the distance between two objects measures the number of modifications required to convert the first object to the second. An optimal mapping between graphs can hardly be defined [40]. An inexact graph

matching algorithm is used to evaluate the structural similarity between graphs.

These methods are guaranteed to find the optimal solution but require exponential time and space due to the NP-completeness of the problem. In this paper, much effort is concentrated on the accuracy of the retrieved exemplars rather than the speed with which the exemplars are retrieved. As well, the underlying logic behind the algorithm is to minimize the edit-distance or the number of steps required to change exemplar A into exemplar B.

4.2. Structural metrics for design exemplar

Based on the discussion in the above section, similarity measures have been widely recognized as key to case retrieval. For the problem domain of design exemplar authoring, three similarity metrics have been suggested, namely elemental (entity and relational), edit-distance, and value-based feature similarity, all of which may be used in combination. The objective is to reduce the number of exemplars in the case base to a manageable number. Hence a series of algorithms are to be developed to implement these filters.

4.2.1. Elemental Similarity (Based on features)

The retrieval system needs to suggest alternative configurations as the user starts authoring exemplars. First, a filter on all available exemplars is employed that will eliminate those exemplars that do not contain at least as many entities and relations of as

many types as specified by the user.

Entity Similarity: The entity similarity measure is defined as the number of entities that are shared between the query and the case. For example, three exemplars A, B, and C can be considered, the entities of which can be represented as three sets.

Set A: (p, q, r, s); Set B: (p, q, r); Set C: (p, r);

Size (Set A ∩ Set B) = 3; size (Set A ∩ Set C) = 2;

In this case, exemplar A is more similar to B than to C since the number of common entities shared by A and B is more than the number of entities between A and C. This metric serves as the first level of parsing the exemplars. So, for example, it can be considered that the user in the process of authoring an exemplar has thus far authored three circles. On application of this filter, all the exemplars in the case base that do not have at least three circlers are filtered out of the case base. This poses a limitation on the filtering of the exemplar. The assumption behind having this filter is that the exemplar author would not wish to look at design exemplars having less than three circles. For example, it is possible that there may be an exemplar in the database which is similar to the one that the exemplar author is trying to author except that it has two circles instead of three. The author would be able to use this exemplar by adding one entity. The application of this filter will however filter out this exemplar. As well, it is possible that the exemplar author authors three edge curves. A circle is a type of an edge-curve, but the application of the entity filter would not return such exemplars.

Relation Filter: Second, a filter will be employed that will eliminate those exemplars that do not contain at least as many relations of as many types as specified by the user. Similar to the entity similarity metric, the relation similarity measure is defined as the number of relations that are shared between the query and the case. For example, in addition to the three circles the user decides to author two 'tangent' relations. On application of this filter, of those exemplars that are left in the case base after the first filter, those exemplars that do not have at least two 'tangent' relations are filtered out of the case base. In case equality relations are present in the exemplar being authored, all exemplars not having equality relations are filtered out on application of this filter. However, the filter does not evaluate the similarity of the equations. However, an investigation of similarity evaluation of the equations is out of scope.

The order in which the filters will be applied is not known yet. It is not clear as to whether the entity similarity should be applied first followed by the relation similarity or vice versa. Applying both filters at the same time can also be considered. The ideal order of application of the filters would be known only during the implementation and validation phase. The experiments conducted in order to determine the accuracy and application of these filters are discussed in Chapter 8.

4.2.2. Attribute similarity

Having employed the above mentioned similarity metrics, the third measure is based upon the attributes (alpha, alpha-beta, beta, match, and extract) of the entities and

relations. Inclusion of this metric is important, since the number of changes made to exemplar 'a' in order to change it to exemplar 'b' includes the changes made to the attributes of the entities and relations. The attribute 'match' corresponds to characteristics that are explicit in the design model, whereas the attribute 'extract' refers to characteristics that are implicit in the model. The attributes alpha and beta correspond to the states before and after modification of some characteristics of the model, whereas alpha-beta corresponds to characteristics that are present in the model both before and after modification. The attribute filter is employed to eliminate those exemplars that do not have entities and relations with the same attributes as authored by the user. Thus, for example, the circles in the above exemplar are alpha-match. On application of this filter, those exemplars that have three circles but the circles are not alpha-match are filtered out of the case base. The limitation to this algorithm is the same as that for the entity similarity metric. For example, if there is an exemplar that is similar to the one that the user is trying to author except that the attributes of the circles are not alpha-match, then the filter would filter out this exemplar.

4.2.3. Edit distance

There is a possibility that even after the application of these filters, there will be a sizeable number of exemplars left in the case base. To determine which among these exemplars are most similar to the exemplar that is being authored, it is necessary to determine how the entities and relations are structurally related. Hence a filter is

employed to screen those exemplars that are not structurally similar to the one that is being authored. To do so, the existing pattern matching algorithm is used. The current exemplar is passed as a query to the exemplars in the case base. On finding a match, that exemplar is loaded in a separate window for viewing. If no match is found then the query is refined by removing the relations one by one, as each time the refined exemplar is supplied as a query to the pattern-matching algorithm. Once a match is found, it is loaded in a separate window. However, if no matches are found even at this stage, then the query is refined by removing two relations at a time. In case, if there were more than twenty matches at any point, then the exemplars with least number of entities and relations in total are displayed.

4.3. Theoretical Session

This section is aimed at demonstrating the application of all the metrics mentioned above, with the help of a sample case base. Figure 4.5 shows the cases in the case base. Each exemplar shown has been described below:

c Finds all models that have a circle, a line tangent to the circle, and another line perpendicular to that line. All the entities and relations are match.

d Finds all models that have two planes parallel to each other. The planes and the parallel relation are match in the design model. Having found parallel planes, the exemplar finds the distance, checking if the distance is less than 10 units. The parameter entity and distance and equation relations are extract.

e Finds all models that have two circles that are tangent to each other. All the entities and relations are match.

f Finds the angle between all the planes in the model that share a boundary. The plane entities and the boundary relation are match, while the angle relation is extract.

g Finds the distance between all pairs of lines that are parallel to each other. The lines and the parallel relation are match. The distance parameter and the distance relation are extract.

h Finds all models that have two lines parallel to each other. The lines are bound to two points by the boundary relation. All the entities and relations are match.

i Finds the distance between pairs of parallel planes in the model, as all entities and relations are match.

j In this exemplar, two points are coincident on two lines respectively, which in turn are coincident on two planes that are parallel to each other. The distance relation finds the distance between the two points. The distance relation and the distance parameter are extract, while other entities and relations are match.

k Finds the distance between pairs of parallel planes in the model, but the distance relation and the parameter entity are extract. The planes and the parallel relation are match.

l Finds all models that have two circles tangent to two planes respectively which in turn are parallel to each other. The planes and the circles are match. The parallel

relation is match as well.

The legend for the exemplars is as shown in Figure 4.4.

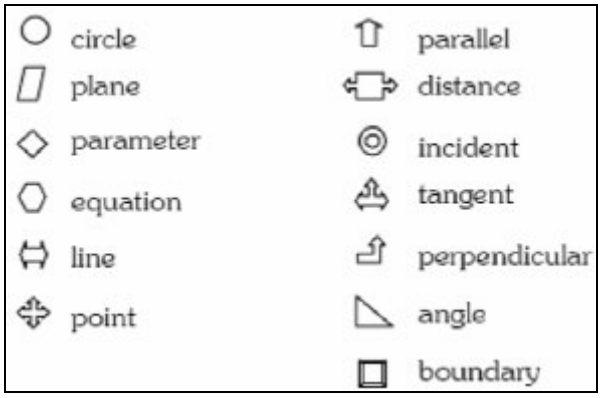


Figure 4.4: Legend for exemplars in table 4.1

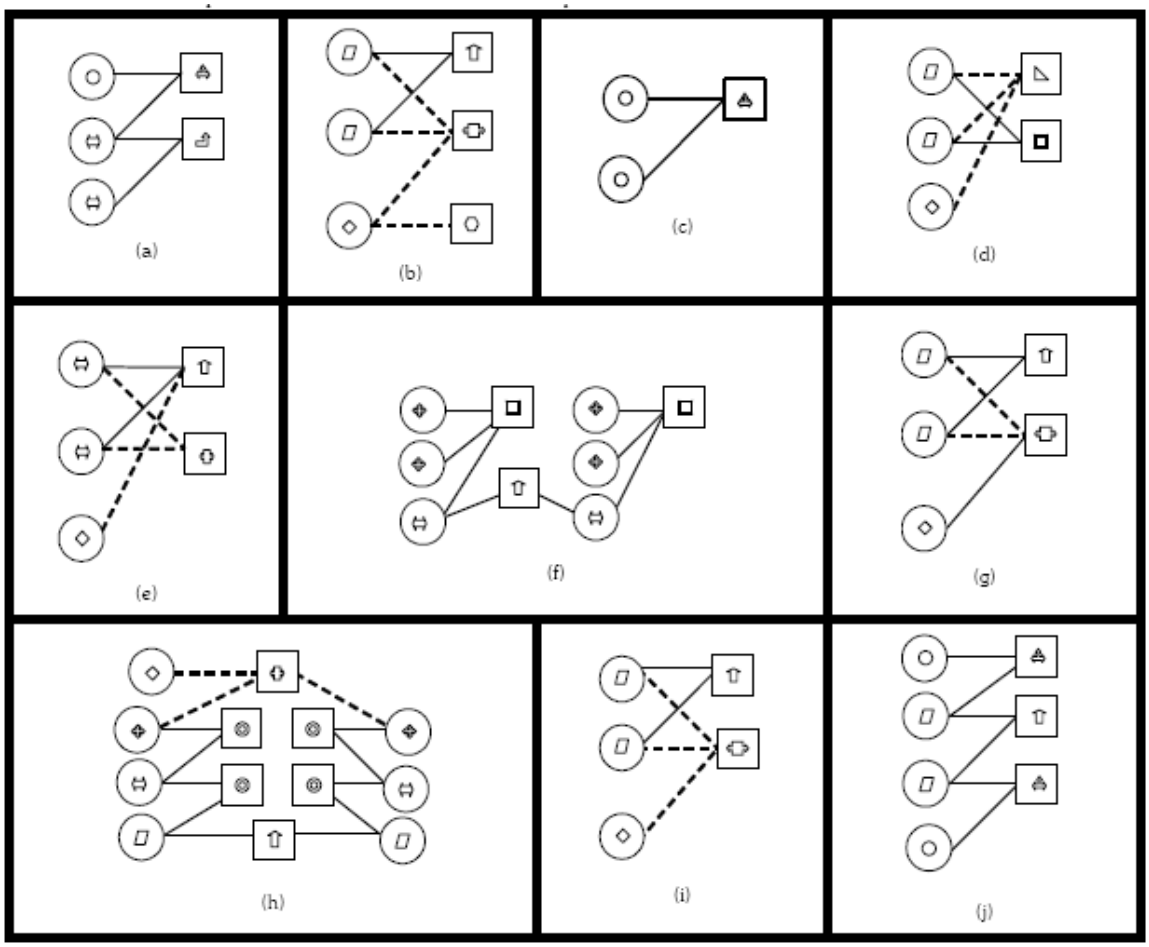


Figure 4.5: Sample Case Base

As an example, a design exemplar user is trying to author an exemplar to find pairs of parallel planes in a model, the distance between which is less than 25 units and more than 10 units. The exemplar that is the objective for the exemplar modeler is found in Figure 4.6.

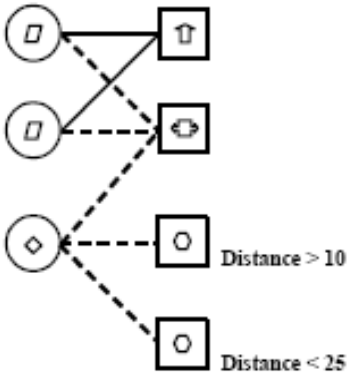


Figure 4.6: Designer's idealized design exemplar.

This exemplar will consist of two planes constrained by the parallel and distance relations. It will also consist of two equality relations to constrain the value of the distance. Figure 4.7 below shows the state when the user has authored two plane entities and the parallel relation between them. The solid lines imply that the attributes of the entities and the relation are match.

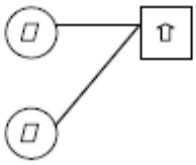


Figure 4.7: Initial state of the Design Exemplar Authoring Session

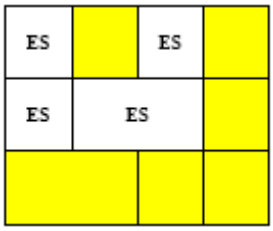


Figure 4.8: Application of Entity Filter

If the user decides to retrieve exemplars from the collection of existing design exemplars, the similarity metrics are evaluated. The Entity Similarity metric filters out

exemplars a, c, e, and f, since these exemplars do not contain at least two planes (Figure 4.8). The Relation Similarity metric filters out exemplar d, as it does not contain at least one parallel relation (Figure 4.9). Further, the attributes of exemplars b, g, h, i, and j match those of the exemplar authored by the user since the planes and the parallel relation are match. Finally, the planes and the parallel relation in these exemplars are structurally similar to the planes and the parallel relation in the exemplar being authored. Hence the Attribute Similarity filter and the Structural Similarity filter do not filter out any exemplars. Hence after the application of all the filters, exemplars b, g, h, i, and j are left in the exemplar collection (Figure 4.10).

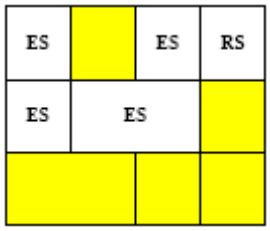


Figure 4.9: Application of Relational similarity

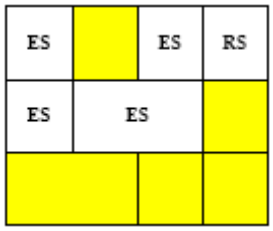


Figure 4.10: Exemplars similar to initial exemplar

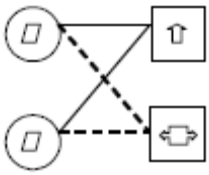


Figure 4.11: State of Design Exemplar Authoring Session

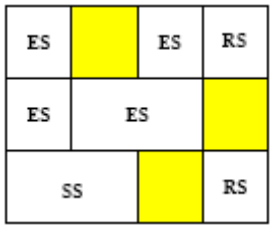


Figure 4.12: Exemplars similar to exemplar in Figure 4.11

As a next step, the user decides to add the distance relation between the parallel

planes. If the user decides to get help from the computer at this point, the filters will be again applied as mentioned in the first step. Figure 4.11 below shows the state when the user adds the distance relation. Since exemplars b, g, and i have at least one 'distance' relation and it is 'extract', these exemplars get retained in the case base. As well, since exemplar h is not structurally similar to the one being authored, it gets filtered out (Figure 4.12).

In the next step, the user adds the 'parameter' entity and makes it 'extract' (Figure 4.13). At this instance, of the exemplars left in the case base, exemplars b and i match the exemplar authored by the user, since the parameter entity in exemplar g is not extract. When these exemplars get displayed, the user decides to accept exemplar b, since there is only one more step needed to attain the user's objective (Figure 4.14).

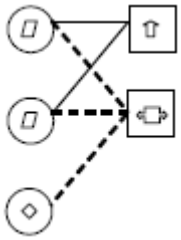


Figure 4.13: State of Design Exemplar Authoring Session

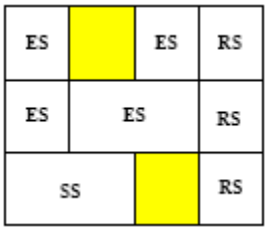


Figure 4.14: Exemplars similar to exemplar in Figure 4.13

The metrics mentioned thus far are useful in mainly evaluating the structural similarity of exemplars. However these metrics do not capture the semantics of the exemplar and hence may not be sufficient to satisfy the purpose of the retrieval tool. The

order of applying these filters is not yet known. As well, it is not clear as to which filter is the most influential in filtering out the exemplars and keeping the number of retrieved exemplars to a manageable number. As mentioned earlier, experiments need to be conducted in order to determine the accuracy and effectiveness of these structural filters. Implementation and experimental validation of the structural similarity measures are described in Chapter 7 and Chapter 8.

Chapter 5

SEMANTICS IN ENGINEERING DESIGN

The second step in developing an exemplar retrieval tool is to understand and define semantics in engineering design. In linguistics, the study of semantics is purely concerned with the meaning of sentences or phrases and not with the syntax or vocabulary [42-44]. Meaning can be defined as a relationship between words and expressions used or it can be indirectly defined by the responses that people make resulting from these words and expressions [42]. Another view of meaning is that it is the mapping between words and expressions and mental images based on previous experience [44]. It is important to note that the words and expressions can imply different meanings within different contexts and for different users. Meaning of a sentence can change depending upon whether it is taken in isolation, whether it is taken in a given context of utterance, or whether it is meant as a form of communication in a social setting [42, 44-46]. This is referred to as the principle of compositionality. It states that the meaning of a sentence is determined by the meaning of its component sections and the manner in which these words are syntactically arranged [47]. For example, the following two sentences mean completely different things because they are arranged differently.

1. Every hole in this solid model is for a bolt.

2. There is at least one hole for every bolt in this solid model.

As well, consider the word “design”. This word when taken in isolation may assume different meanings such as the final product or the actual design process. However, when used in a sentence such as, “The focus of this research is to design the cooling system of a car”, it is natural to assume that the word refers to the actual design process. In other words, “semantics” may mean different things in different domains, through different contexts, and to different people. A useful way to look at semantics in the field of machine translation is associating words with semantic features corresponding to their sense [48]. For example, the word ‘man’ can be associated with a set of features such as adult, human, and masculine. Similarly, the word ‘woman’ can be associated with features such as adult and human but not masculine. These semantic features constrain other words with which these words can appear. For example, the words “drill” and “hole” may appear together, but the words “drill” and “shaft” may not always appear together. For example, a sentence such as “The machinist drilled a hole” may be valid; however the sentence “The machinist drilled a shaft” is not. These examples from linguistics suggest that humans tend to attach meanings to ‘concepts’ or words.

5.1. Definitions of Semantics

Various definitions of semantics exist in the literature. These definitions of semantics are reviewed below and include views of semantics as: design knowledge, text, relations, and design intent.

5.1.1. Semantics as design knowledge

Design semantics has been defined as "representations of knowledge regarding the product and process" [49]. Consider a collaborative environment that has been developed, wherein a group of designers collaborating over the internet model a 3D layout and semantically grounded behavioral product description. It is argued that three knowledge level descriptors are required to capture the meaning of components; function, behavior, and structure. Function is defined as the role of the object within overall assembly; the object could be a single component, a group of components or a subassembly. Behavior is defined as the actions through which this function is accomplished. Structure is defined as what specific attributes of the physical object are related to achieving this behavior and functionality [49].

5.1.2. Semantics as text

Semantic representation is also defined as "verbal or textual representation of the object" [50]. For example, the word "bolt" or the sentence "the stress on the billet is proportional to the strain" can be considered semantically rich. The initial need is usually expressed in a semantic language in the form of a written specification or a verbal request. As the design develops, other representations are used leading to the final physical product. As the product development process continues, the product continues to get refined. Hence, semantic representation is abstract as compared to the physical form. As the design progresses from the initial sketch to the final drawing, the level of

abstraction is refined. This is illustrated in Table 5.1.

Table 5.1: Semantic information at varying levels of abstraction (adapted from [50])

	Levels of Abstraction		
	Abstract ----------	----------	---------- Concrete
Semantic	Qualitative words . Eg: A bolt	Reference to specific parameters or components. Eg: A short bolt	Reference to the values of the specific parameters or components. Eg: ¼ -20 SSbolt.

5.1.3. Semantics as Relations

Semantic information has also been defined as "information of knowledge contents" [51]. Relations between entities and functions are an example of knowledge content. It is argued that there is no knowledge that directly connects the functional knowledge to the attributive knowledge (color, size, material) [51].This connection is mediated by entities. It is argued that this knowledge content is useful in abductive reasoning. Abductive reasoning is useful in reasoning about entities performing a required function [52]. Semantic information helps in adding another restriction to the abduction process.

5.1.4. Semantics as Design Intent

Design Semantics has also been defined as the design intent of the designer. Design intent in this case is defined as "the function and structures that the product must

have" [53]. For example, the intent of a trash compactor is given by the statement: ''A trash compactor is used for reducing the volume of the trash collected and must have appropriate strength, section type, dimensions, and material properties''. Design intent has also been defined as the reasons behind decisions [54]. Design intent and rationale, are nearly synonymous as design rationale captures not only the results of the design decisions, but also the reasons behind them and the alternatives considered [55]. Feature-based design intents associated with a solid model include the designer's concerns that explain a specific geometric attribute or configuration [56]. It has been argued that every feature has some intent associated with its form. In other words, the size and geometric relations between features and form capture the designer's quantitative and qualitative thoughts [57].

Synthesized Definition:

Based on these definitions, semantics can be defined as what is understood from what is being said. To try to represent semantics is to try to represent "understanding". It may not be possible to represent "understanding" in any form. But it may be possible to facilitate understanding by representing knowledge precisely. 'Precise representation of knowledge' means representing knowledge in such a manner that there is only one 'understanding'. In other words, semantic information is information represented textually that relates concepts to express a unique interpretation. Design Semantics is defined as design knowledge that is inferred from design information (e.g. the CAD model, design artifact mode, design documents, etc.). It can be inferred that domain

knowledge is useful in understanding the meaning of a sentence or an expression. It is the nature of this knowledge that is central to the study of semantics. This knowledge, for example, could include intended function or actual behavior of the model. One type of product knowledge could be the "design intent" or "design rationale", whereas a different type could include non-geometric descriptions of the product such as design specifications or function structures. Process knowledge could include knowledge about the manufacturing or design processes used to create the artifact.

5.2. Types of Semantic information in engineering design

In order to determine the different types of design knowledge associated with design documents were studied. These documents include a requirements list of a geostationary satellite from Northrop Grumman [58] and requirements' lists generated through interviews with graduate students working on industry sponsored projects at Clemson University. Other documents included a senior design report from a senior design class at Clemson University, and a patent [59]. The information and knowledge present in the documents could be broadly classified into two categories: 1. Product semantics (design knowledge about the product being designed), and 2. Process semantics (semantic information about the process).

5.2.1. Product Knowledge

If the statement contains semantic information regarding the product then it can be further classified as:

1. Rationale for design decisions: Design rationale can be defined as the reasoning behind design decisions taken during the design process. For example, consider the statement, "Because of the length of time involved, it may be impractical to conduct a comprehensive electrical functional test during spacecraft level thermal-vacuum verification". This statement provides reason as to why it may be impractical to conduct a comprehensive electrical function. This includes the reasoning for the decisions taken at any stage in the design process. It could involve reasoning for choosing a particular concept over the other, reasons for including specific features in the design, etc. Other examples of statements containing rationale may include:

i. *Door must be in a locked position to prevent outside forces from closing the door at unscheduled times. This statement provides a reason for keeping the door in a locked position.*

ii. *Because of the length of time involved, it may be impractical to conduct a comprehensive electrical functional test during spacecraft level thermal-vacuum verification.*

2. Requirements: Statements that explain or describe the requirements on the product to be designed can be grouped in this category. These statements can then be further classified as per the type of requirement. These types include purpose of the design purpose, function or behavior of the product, tests that the product must satisfy, and other requirements.

2a. Objective / Purpose of the design process

The objective of the design process can be gleaned from the design problem statement. If we look at the functional decomposition diagram of the entire design problem, then the objective of the design process can be gleaned from the root node of the tree. One may argue that this may be classified as the "main function of the design". However, we need to acknowledge that understanding the objective of the design process helps in clarifying the various sub-functions that need to be satisfied in order to achieve the main 'objective' or the 'purpose' of the design process. It also helps in identifying design knowledge that is relevant to solve the design problem. It can be argued that this is basically a high level requirement, but my argument is that we need to represent this 'high level requirement' explicitly in the CAD model or associate this with the final form. Examples of such statements include:

i.	*To design a simulator to rotate the body-in-white (BIW) through 360 degrees.*
ii.	*To design a door brake for holding car doors open at specified angles.*
iii.	*To re-design an engine cooling system to reduce weight.*

2b. Function or Behavior of the product

Statements that describe the functionality that the product must have or explain the desired behavior that the product should exhibit can be included in this class of design requirements. Some examples are listed below.

i. *One of the functions of the simulator is to hold the BIW at specified angles.*

ii. *"Support the telemetry and command interfaces with the GTACS via a direct Ethernet connection using the Transmission Control Protocol/Internet Protocol (TCP/IP)."*

2c. Tests that the product must satisfy.

This class of design requirements includes statements explaining the tests that the product must satisfy. Examples of such statements are:

i. *Prior to mating with other hardware, the electrical harnessing shall be tested to verify proper characteristics, such as routing of electrical signals, impedance, isolation, and overall workmanship.*

2d. Other Requirements on the product.

Statements that can be grouped in this category include statements that include detailed level requirements those need to be satisfied by the final design.

i. *The door must be in a locked position.*

3. Information useful while designing the product.

Background information is sometimes needed in order to develop a better understanding of the design problem. This background information may be also included in design reports, requirements lists etc. Background information that is necessary in order to explain or understand the problem correctly could include information about the

use environment of the product, the amount of savings resulting from designing the product etc. An example of such a statement is shown below.

i.	*BMW employs a variety of fixtures to assist in the painting of automobile bodies. This is done to streamline the process. During the paint cycle robotic arms open and close the doors as necessary.*

4. Descriptions.

All types of design documents such as design reports, patents, requirements lists, etc., explain the material with words and with the help of figures. Final design reports may contain explanations of the final form of the design. An example of such a statement is the handle of the cup is circular. As well, documents may also contain sentences that explain what a specific figure in a document represents, such as "Figure 1 shows a schematic diagram of a flexible flow line". Documents also contain sentences that tell the reader what to expect in a particular section of the document. An example could be "Section 2 of this report explains the application of the different design tools used in the design process". To summarize, textual descriptions could be broadly classified into the following three categories.

i.	*Textual Description of the final design*
ii.	*Textual description of the figures in the document.*
iii.	*Textual description or explanation of the content of a particular section(s) in a document.*

5. Definitions or Terminology used.

Design documents may sometimes introduce some new terms that are relevant to the specific design problem at hand. As well, the same terms may have different meanings depending on the type of document and the organization. These terms may be defined explicitly in the document to avoid any kind of ambiguity. These definitions could be regarded as a separate type of semantic information as they are needed to improve the understanding. An example is listed below.

i.	*An outage is anything that prevents execution of instrument operations during normal on-orbit mode or anything that prevents the transmission or relay of instrument science data.*

6. Classification

Class Headings or section headings used in design documents such as problem statement, general requirements, etc. can be considered as a separate type of semantic information as they contribute to the understanding of those sections.

5.2.2. Process Knowledge

If the statement contains semantic information regarding the process followed while designing the product, then those statements can be further classified as follows:

1. Requirements on the process followed.

Requirements lists could include requirements of how the design should be carried out or may have a specific set of guidelines that should be followed while designing the product. These requirements can be considered as a separate type

of semantic information since following a specific guideline to design the product could affect the design outcome.

i.	*NASA technical paper 2361 titled "Design Guidelines for Assessing and Controlling Spacecraft Charging Effects" should be used as a guide for the prevention of spacecraft charging".*

2. Description of the design process.

Design reports often contain textual or schematic descriptions of the design process that was followed while designing the product. These descriptions could also include an explanation of the design tools that were employed while designing the product. An example is listed below.

i.	*Quality Function Deployment (QFD) is a tool which helps to ensure the creation of a quality product; it aids in the determination of the engineering quantities most critical to customer satisfaction. In this case, a QFD was used to deploy customer input throughout the design process.*

3. Result of the design process at the various stages.

Each stage of the design process yields some kind of result which is utilized in the latter stages of the design process. The outcome of using these design tools may be explained or described in the design documents and can be considered as process semantic information.

i. *The team sought to only evaluate five designs in stage two of the conceptual design process. The top five designs promoted to the next stage were "BMW Prototype," "Spring Clip," "Friction Plunger," "Torsional Cam Follower," and "Pen."*

4. Rationale for the choice of design tool.

This is analogous to the rationale behind the design decisions at the various stages of the design process. At each stage of the design process, there may be several design tools that could be employed. There is always some kind of rationale associated with choosing one design tool over the other. This rationale could also be considered as process semantic information.

i. *Although a system perspective is paramount to any design process, focus at the component level is a helpful simplification; individual components can be analyzed and redesigned with greater ease than an entire system.*

5. Classification

Headings of the various stages of the design process can be grouped together in this category.

5.2.3. Parametric vs. Non-parametric information

The different types of semantic information mentioned in the classification scheme can be further distinguished into information that can be parameterized in their existing state and those that need to be decomposed further. For example, consider the requirement, "Spare components shall undergo a test program in which the

number of thermal cycles is equivalent to the total number of cycles other flight components are subjected to at the component, subsystem, and spacecraft levels of assembly". This requirement contains design knowledge since the designer can infer some information on how the test is conducted. As well, the design knowledge inferred from the above statement can be represented in terms of an equation such as Nthermal = Ncomponent + Nsubsystem + Nspace_assembly; where N stands for the number of cycles. This is a type of requirement that is semantic and can be parameterized. A requirement that cannot be parameterized in its present form may be broken down into detailed level requirements such that each of the sub-requirements can be parameterized. For example, it may seem that the requirement, "The test configuration shall reflect, as nearly as practicable, the configuration expected in flight", cannot be parameterized. However, it may be possible to decompose the meaning of "as nearly as" into quantitative terms and then this requirement can be parameterized. An example of a requirement containing semantic information and that cannot be parameterized could be, "Cycling between acceptance temperature extremes or qualification temperature extremes has the purpose of checking performance at other than stabilized condition". This statement states the purpose of cycling between temperatures. Consider the statement, "Testing at lower levels of assembly has many advantages: it uncovers problems early in the program when they are less costly to correct and less disruptive to the program schedule; it uncovers problems that cannot be detected or traced at higher levels of assembly; it characterizes box-to-box EMI performance, providing a baseline that can be

used to flag potential problems at higher levels of assembly; and it aids in troubleshooting." This sentence states the rationale for testing early on in the design process. It may not be possible to parameterize this type of information. This example shows that some types of semantic information may be best represented textually.

5.3. Implications

In order to develop an effective exemplar retrieval system, it is necessary to identify the different types of semantic information that should be associated with design exemplars. The exemplars for finding the distance between two planes show that there are multiple ways of authoring an exemplar meant for a specific purpose. These exemplars may be structurally different but can be considered similar based on intended use. As well, the exemplar retrieval tool can be used to author precise exemplars. In order to help the exemplar author evaluate the usefulness of the retrieved exemplars, it would be useful to represent the rationale for authoring exemplars meant for the same purpose in different manners. Thus types 1 and 2a are the types of semantic information that are most relevant to this research. All other types of semantic information are out of scope for modeling semantics in this research.

This task helped achieve an understanding of design semantics and identify the different types of semantic information necessary for the purposes of the exemplar retrieval tool. The next step is to review the different knowledge representation schemas developed in literature. Evaluating the different knowledge representation schemes in

order to identify the most appropriate representation scheme for representing semantic information is out of scope for this research.

Chapter 6

KNOWLEDGE REPRESENTATION

Traditional commercial CAD systems do not fully allow for models to convey information about the actual physical artifact that it represents. The various types of information may include the artifact's use by people, the artifact's relationship in the environment (especially with respect to sustainability issues), how the artifact is manufactured, and the artifact's life cycle issues such as maintenance, recycling, or eventual disposal. A geometric description alone does not carry this semantic information. Features technology attempts to bridge this gap, but is limited to finite domains and only for explicitly captured geometric regions of interest [60, 61]. As mentioned earlier, semantics is the inference of design knowledge as facilitated by the representation of design knowledge.

Knowledge representation can be defined as a mapping between the concepts and relations in a problem domain and the computational objects and relations in a program [62-64]. There have been different representation schemes that have been developed, each having its benefits and limitations. Each scheme should be studied and evaluated in order to determine the appropriate representation scheme for representing the different types of design knowledge. For example, if the representation can represent the attribute "is tough" of a laptop, then can the argument "SS-316 is tougher than SS-304" be represented using the same language? The choice of an appropriate representation

language is necessary in order to facilitate the intended interpretation. However, it may be necessary to use different representation schemes in order to represent different types of design knowledge.

There are two aspects to a knowledge representation scheme, namely a syntactic aspect and an inferential aspect [65]. The syntactic aspect is concerned with the way knowledge is represented explicitly. While defining the syntax of a knowledge representation scheme, it is necessary to specify precisely what expressions are part of the language and how these expressions can be combined together to form new expressions. It is necessary to tell the user what expressions are valid in order to represent knowledge while using a specific knowledge representation language.

The second aspect of a knowledge representation scheme is the inferential aspect. The inferential aspect refers to the way in which explicit knowledge is used to infer knowledge that is implicit. A knowledge base of any representation scheme will always contain pieces of information which can be used to interpret knowledge that is implicit. For example, if it is stated that "all laptops have a 15.4" screen" and "Chhavi owns a laptop" then it can be inferred that the "Chhavi's laptop has a 15.4" screen". The interpreter of any knowledge representation language behaves according to some abstract rules referred to as "inference rules". Inference rules are domain independent and applied by the interpreter according to some abstract rules. These abstract rules define the way in which the interpreter works independent of the domain-dependent information stored in the knowledge base.

This said, however, the syntax and the inferential aspect of a knowledge-based representation are closely related. The syntax of every knowledge representation language specifies some primitives which can be combined to form more expressions. The interpreter for the language is developed such that it understands the meanings of these primitives and their relations.

6.1. Knowledge Representation Criteria

As mentioned above, different knowledge representation schemes can be used to represent knowledge. There are different criteria which a knowledge representation language should satisfy in order to be used for representing knowledge [66, 67]. These nominal criteria are discussed below.

1. Metaphysical Adequacy: This criterion states that there may not be any contradiction between the knowledge that is represented and their actual representation. This criterion does not determine which representation is better as there may be more than one representation schemes that satisfy this criterion. If the laptop example is considered again, the fact that needs to be represented is that laptops have a 15.4" screen. If the representation allowed a laptop to be represented that has a 19" screen it would contradict fact and would be considered metaphysically inadequate as a 19" screen does not exist.

2. Epistemic Adequacy: This criterion refers to the need of being able to represent what needs to be represented. Considering the laptop example again, the

representation should not only allow the size of the laptop screen to be represented but also the color of the laptop if necessary. The intended use of the representation represents the epistemic adequacy.

3. Heuristic Adequacy: This criterion refers to the need for the representation to be capable of expressing the rationale behind solving a problem. This criterion would be a requirement if it is needed to build systems that are capable of "self-consciousness" or capable of reflecting on their own reasoning. In case of the laptop example explained above, the representation is not heuristically adequate. It is possible to represent only the size of the laptop screen, but not why the laptop screen has a size of 15.4".

4. Computational Tractability: The criteria discussed above would hold good for representing knowledge for any purpose. However in this case it is necessary to incorporate the knowledge in a computer. Hence in this case the representation language should be computational tractable. Computational tractability implies that it should be possible to manipulate the representation efficiently within a computer system.

All the above criteria are aspects that a knowledge representation scheme must fulfill. There are other desirable features that a knowledge representation scheme may fulfill. These criteria are listed below.

1. Lack of Ambiguity: This requirement specifies that there is only one interpretation possible from one valid expression. For example the statement "Every person in the office drives a car" could mean that every person in the office drives the

same car or every person drives their own car. There are multiple interpretations possible for this statement. However the statement "Every computer in the lab has its own 19" monitor" facilitates only one interpretation and hence can be considered to be unambiguous.

2. Clarity: In order to use the representation in a computer system it is necessary to input information in the knowledge base by an expert in the subject domain. However it is important to represent the knowledge in a manner such that it is understood by people who are not experts in the domain. The ideal scenario is when a user interacts with the knowledge representation scheme his or her understanding is improved rather than being clouded. An increased benefit is when an expert interacts with the system his own understanding of concepts is increased and improved. Therefore clarity of a representation scheme is to be able to represent knowledge such that it is easily understood by people using the system.

3. Uniformity: There may be different types of knowledge that need to be represented. It is desired that the chosen language of representation should be able to represent all kinds of knowledge. The more kinds of knowledge that a language can represent the better it is. It may not be possible to achieve this. However it is important to be consistent so that all knowledge of a specific type can be represented in the same manner.

4. Notational Convenience: An important criterion while choosing a certain representation language is the convenience of notation for representation.

Ideally both the users as well as the people supplying the knowledge should find the notation easy to use.

There are different types of knowledge representation schemes that have been developed in literature. The different types of knowledge representation schemes discussed in this section include logic representation schemes, procedural representation schemes, structured, and network representation schemes [64].

6.2. Logic Representation Schemes

The representation schemes that fall in this category use formal logic to represent knowledge. Logic can be defined as the study of correct inference. It may not be always possible to agree on what is a correct inference, but it is necessary that if the inference is true then the premise on which the inference is based must also be true. Similarly, if the inference is false then the premise on which the inference is based must also be false. This implies that the conclusion must be "truth preserving". To illustrate these concepts, the following examples can be considered.

Gears can be used for speed reduction.

Speed reduction is required.

Therefore gears can be used.

Gears are used.

Therefore speed reduction must be required.

The first argument can be considered to be truth preserving. If it is true that speed

reduction is required then use of gears is one way of achieving it. But the second argument cannot be considered truth preserving since gears can be used to transmit motion without reducing speed.

There are three basic components of logic [46, 68]; syntax, semantics, and proof theory. Syntax is a set of atomic symbols and rules for combining them to form a meaningful expression. Semantic is the meaning of the atomic symbols and rules for inferring the meaning of the expressions from the meanings of the components of the expression. Proof theory is a set of specifications, called "rules of inference", that can be used to determine what meaningful expressions can be added to the set of initial collection of well-formed expressions, called "proof".

Description logic is the name given to a group of knowledge representation languages that can be used to represent knowledge of any application domain in a formal and well structured manner. One type of logic that has been developed is propositional logic. Propositional logic uses connectives such as 'and', 'or', or 'not'. First order predicate logic is a common logic representation scheme that carries the analysis down to classes, objects, or relations. It uses connectives such as "for all x", and "for some x". Description logic is primarily used in the development of ontologies. For example, sub languages of the web ontology language, such as OWL-DL and OWL-Lite are based on description logic [69]. Ontology provides a vocabulary in order to represent knowledge about a domain and specifies a set of relationships between the terms of the vocabulary. The use of description logic in engineering design is being investigated. For example, it

has been proposed that description logic can be used for constructing design repositories because of its inference capabilities [70]. Design repositories overcome the limitations of traditional databases by applying knowledge representation techniques. Design knowledge such as function, rationale, and behavior can be captured in these repositories and reasoned on to help the designer search, retrieve and categorize solutions to previous design problems. Inferences such as classification, subsumption, and least common subsumer prove to be useful to support these tasks. There are several examples of using description logic for representing design knowledge in engineering design [71, 72]. One such exe example is discussed below.

Description logic has been used for the retrieval of finite element analysis (FEA) models [71]. In this paper description logic has been used to describe archived models and to build expandable classification hierarchies. The proposed approach uses description logic to represent design problems associated with FEA models. The description logic concepts are derived from an extensible set of concepts that describe the various aspects of the design problem such as structural components, relationships between them and the applied loads. The subsumption relationships that exist between the description logic concepts are used to create concept hierarchies. These concept hierarchies are then traversed to retrieve similar FEA models. As part of this research, the first step was to choose description logic capable of expressing the information associated with models that need to be represented. This was done by evaluating the different types of knowledge that need to be represented and then mapping this information to specific

description logic. The types of information that need to be represented include structural elements, part-of relationships, relationships between the structural elements, materials, and applied loads. The description logic ALE was chosen to represent FEA models. The vocabulary is expanded by creating description logic concepts completely subsumed by a set of basic concepts. Constructors permitted in ALE were used to combine these concepts in order to form physical context representations. These physical context representations form the indices to classes of models that represent the same physical problem at varying levels of detail. The class hierarchy can be developed and expanded due to the use of description logic to create these indexes. Having developed the class hierarchy, a domain-independent algorithm developed by the authors is used to traverse the description logic hierarchy to retrieve relevant models.

There are two arguments in favor of using logic as a knowledge representation language. The first argument is that logics have semantics. Semantics enables one to determine exactly what each expression means and allows one to find out if a given piece of information is being represented adequately. As well, it allows one to check whether the procedure used by the inference engine is precise or not. An inference procedure is correct if, when the input sentences are true, the inferences are also true. Semantics allows one to specify the conditions under which sentences are true and this allows one to check the soundness of the inference engine.

The second argument in favor of logic as a representation language is its expressiveness. Two main aspects to the expressiveness of logics are the ability to

express incomplete knowledge and the fact that there exist different logics that can be used. Logics allow one to express incomplete knowledge or information about incompletely known situations [73]. For example it is possible to represent the statement, “the tire tread is made of either rubber or nylon” using logics. As well different types of logics can be used to represent different types of knowledge. For example there are temporal logics that can be used to represent information about time. Using temporal logics, the information that laptop screens used to be 15.4” in size five years ago as well as the information that there are other sizes possible today can be represented.

Apart from the arguments in favor of using logic as a knowledge representation language there are some arguments that go against logic representation schemes. The first argument is that logic is too weak. In earlier experiments in using logic as a general problem solver it was found that the resulting programs were very slow. Even simple problems took a long time to solve. Hence, the general consensus was that the use of logic was inefficient. Further research showed that the reason for these inefficiencies was the use of inefficient control regimes [73]. Another argument that goes against the use of logic as a knowledge representation scheme is that it is declarative in nature. Certain types of knowledge are better represented in a procedural format.

6.3. Procedural representation schemes

In this formalism, knowledge is represented as a set of instructions for solving a problem in a procedural scheme [66]. This is in contrast with declarative logic

representations. For example, a rule based system consisting of if-then rules can be considered to be a procedural representation. Some of the arguments in favor of procedural representation schemes include the fact that most of things that are known are considered as procedures. For example, the knowledge of adding two numbers can be represented declaratively using axioms. However, while adding two numbers, humans tend to follow a procedure instead of using axioms. Thus it is simpler to represent this knowledge in a procedural format. As well, procedural representations offer a computational advantage [74]. A program can solve problems faster if some information is stored in a procedural manner as compared to a declarative manner.

Production systems are an example of procedural representation schemes. Production rules are particularly suited to model what is known as "pattern-induced inference". There are three basic components of any pattern-directed inference system. These include the working memory, pattern-directed modules, and the interpreter [65].

The first component is the working memory which includes a set of working elements. These elements will have knowledge that the system has about the problem as hand. For example, if the system is trying to solve a problem of motion transmission, then the working memory will contain elements that represent knowledge relevant to the problem at hand. In this case, the elements will have knowledge about gears, chain drives, belt drives etc.

The second component is a set of pattern-induced modules. Each module consists of a set of instructions and a triggering pattern. As soon as the triggering

pattern is matched in the working memory the corresponding set of instructions is carried out. The result of each instruction is a modification of the working memory. It could result in addition, deletion or modification of working memory elements. For instance, in the example of motion transmission, if the solution selected is gears then working memory elements representing information about chain drives and belt drives will be deleted.

The third component of a pattern-inducing inference system is the interpreter. The main function of the interpreter is to solve the control problem. While solving a problem it is possible that the working memory may match with a number of different triggering patterns. The interpreter's function in this case is conflict resolution and evaluation of which set of instructions should be fired [64, 75]. If several rules are fired at the same time, then the ultimate behavior of the system will depend on which rule was fired first. This is a distinct limitation of the procedural representation scheme.

The possibility of developing the design exemplar into a visual programming environment for mechanical engineers has been investigated. In order to do so, research has been conducted in order to investigate the possibility of extending the design exemplar to support procedural processing of design data [16]. The different components needed for a visual programming language that are not supported by the design exemplar are identified. In order to develop a visual programming language, programming constructs such as conditional branching and looping are identified to be important. Hence, the new feature in the design exemplar system extends the exemplar to support

conditional branching and looping operations. However a compiler has not been developed yet. Extending the design exemplar to support conditional branching and looping, aids the exemplar author in developing complex exemplars by linking smaller exemplars dynamically.

The main advantages of production rules is the fact that they can represent very closely the rules of thumb or heuristic knowledge that experts use to solve problems in a domain [65]. As well, production systems are very modular and hence new knowledge can be easily incorporated in the rule base [76]. The main disadvantage of a production system is conflict resolution [68]. Resolving conflicts in a large rule base may require extensive computation power.

6.4. Network Representation schemes

In case of network representations, knowledge is captured in the form of graphs, where the nodes represent concepts in the domain of interest and the edges represent the relations between the concepts. Examples of network representations include semantic networks, or conceptual graphs. The motivation behind introducing semantic networks as a knowledge representation scheme is understanding natural language as opposed to problem solving.

Two kinds of semantic networks have been developed: inheritance networks and propositional semantic networks. Inheritance networks have some nodes that represent categories (birds, animals, cars), whereas some other nodes represent specific instances

(pigeon, tiger, Ford Fusion) It is important to note that the relation between an individual and its category is of type “instance of” whereas the relation between a category and its super category is of type “is a”. Inheritance networks usually represent information about the nodes but do not represent information about the relations. For example, it is not possible to represent the information that “is a” relations are transitive. These shortcomings can be overcome by using propositional semantic networks. Propositional networks use nodes to represent propositions, individuals, categories, and properties [77, 78].

6.5. Structured Representation Schemes

Structured representations are an extension of network representations, in that, each node in the case of structured representations is a complex data structure consisting of named slots with associated values. Examples of structured representations include scripts or frames. Frames are structured representations of objects. These are similar to semantic networks in the sense that the nodes in this case are called frames, the labeled arcs are called slots, and the nodes pointed to by the arcs are called slot fillers. One difference between frames and semantic networks is that frames allow procedural attachments. For example, frames could have slots filled by procedures such as “if-needed” or “if-added”. If a slot with an “if-needed” procedure is accessed, then the procedure is executed and returns the slot filler. If a slot with an “if-added” procedure is accessed, then the procedure is executed and is expected to return the slot fillers of other slots in the frame depending on the value added to this slot. Scripts are also

structured representations, but unlike frames, these represent activities and not objects [79, 80].

It is claimed that frame-based knowledge representation schemes allow representation of knowledge in the same manner as experts think [81]. Each entity class in the domain there is a corresponding class in the knowledge base. Each entity in the domain is represented by an instance of the class. This type of organization of knowledge makes structured representation schemes easy to use [82]. The use of inheritance to draw inferences about implicit knowledge has its own advantages. Computationally it is more efficient than first-order predicate logic. However it is not as expressive as first-order predicate logic [65].

Apart from the advantages there are three main disadvantages associated with using frames. The first disadvantage is the lack of semantics for frame-based representation languages. The use of a logical constant for each instance frame and one-place predicate for each class frame was proposed in 1979 [83]. For example, the member-of logical constant can be used to infer that the instance has all the properties associated with the class. The second limitation is the use of default inheritance for inference. It is proposed that there is difference between a subclass not inheriting a value from a slot in its superclass and not inheriting a slot itself. With default inheritance this may not be possible. The third disadvantage associated with frame-based knowledge representation languages is the limited expressiveness while representing incomplete knowledge. For example, the only types of disjunctions that can be represented are

disjunctions about the value of a property. A statement such as "the monitor of my computer is either 17" wide or 21" wide" can be represented using this representation by incorporating certain restrictions on what value can the 'size' slot take in the instance frame of my computer. But it may not be possible to represent a statement such as "Either my computer's monitor is 17" wide or it is an LCD monitor".

6.6. Textual Representation

Finally, different techniques have been developed to represent and use design knowledge textually [84, 85]. The process of associating semantic information with the CAD models consists of three steps. First, one must Process the text annotated to the CAD model in order to glean semantics; Second, class structures are generated based on clustering relevant properties together. Finally, a decomposition of the design using belief networks is done.

Some of the advantages of using natural language as a knowledge representation scheme are hypothesized in [86]. First, natural language is very expressive and computationally very tractable. A high correspondence between the syntax of natural language structure and semantic relations contributes towards the expressiveness of natural language. As well, verbal and nonverbal information can be combined effectively using natural language. Each kind of nonverbal information can be represented as a natural language utterance. Natural language allows contradiction and redundancy in knowledge representation whereas other knowledge representation schemes are designed

to prevent contradiction and redundancy from being represented. Finally, natural language reflects the nature of human knowledge acquisition since it is context dependent.

For the purposes of developing the retrieval tool, intended use and rationale for authoring the exemplar need to be represented explicitly. Comparing the various representation schemas to determine the most appropriate one for intent and rationale is out of scope for this research because the focus is on similarity and not knowledge representation. Further, design intent and rationale may be represented using different representation schemes in the future in order to evaluate the most suitable representation scheme. Therefore, it has been decided to represent semantic information by annotating the exemplars with textual information about the intended use and the rationale. This would require establishing a controlled vocabulary and ontology for the domain of engineering design. A controlled vocabulary is a list of terms that have been enumerated explicitly, whereas ontology is a controlled vocabulary expressed in an ontology representation language that has a grammar for using vocabulary terms to express something meaningful within the specified domain [87]. Ontologies for engineering design have been developed in literature [88, 89]. For example, a knowledge representation model based on function-behavior-structure-ontology has been developed [90]. This ontology is used for automatically extracting meta-knowledge from design documents to facilitate design reuse. Some relationships that have been used for creating the ontology include is_a, is_part_of, and has_function_of. In order to maintain

consistency while annotating the exemplars and to ensure accuracy of retrieval, it would be necessary to use a controlled vocabulary and ontology. Developing a complete controlled vocabulary and ontology for the purposes of retrieval is out of scope for this research. Hence, an existing vocabulary is adapted for implementing the exemplar retrieval system.

6.7. Semantic Similarity

Having decided to use a textual representation to represent the intent and rationale in authoring an exemplar the next step is to use this representation to evaluate the semantic similarity between exemplars. Use of text analyzing software was considered to evaluate the similarity between the textual information associated with different exemplars. Most software generate semantic networks or graphs based on some measures such as frequency of words [91, 92]. However in order for these measures to be effective, the intended use and rationale have to be sufficiently large in length. It may be safe to assume that the intended use of an exemplar may be usually written in one or two sentences. An example of intended use may be "to find thin walls in the model for purposes of casting." The rationale behind authoring exemplars in a specific manner may be explained in a paragraph. The rationale may not be lengthy enough for text analyzing software to be effective. Hence, in order to retrieve exemplars that are semantically similar to the query exemplar, it is necessary to review various measures of textual similarity developed in literature [93-95].

6.7.1. Vector Space Model

Use of the Vector Space model is suggested in order to determine the similarity between pieces of textual information [96]. In this model, each document is represented by a vector (wn1, wn2,…,wnm), where wnk is the weight or importance of the term tk in the document dn and M is the size of the indexing term set. The importance or weight of the term can be determined by the number of occurrences of the term in the document. The query is also represented as a vector which is computed in a similar manner. The similarity between the query and a specific document is given by the cosine of the angle between the two vectors.

6.7.2. Distributional Semantics Model

An alternate model called the distributional semantics model has been developed [97]. The hypothesis of this model is that two words can be considered similar as much as their contexts are considered similar. The context of a word is captured using the co-occurrence frequency. The co-occurrence frequency between two words is defined as the number of times the two words appear together in a document. For a specific term, the co-occurrence profile is defined by a vector of the co-occurrence frequencies between the term and a predefined set of terms known as indexing features. The whole document is represented by the weighted average of the co-occurrence profiles for each of the terms. The similarity between the query and the document is given by the cosine of the angle between the two vectors.

6.7.3. Phonetic Codes

Measures of textual similarity based on phonetic codes have been proposed [93, 98]. In 1918, the soundex algorithm was patented by Robert Russel [99]. This algorithm provides an index wherein names are grouped based on phonetics and not on the alphabetical spellings. The algorithms based on phonetics require the generation of codes for all the words in the database. The code of the query word is generated and compared to the pre-processed codes of the words in the database. This does not allow ranking the words in order of decreasing or increasing similarity. Words are considered similar if they have the same code and are considered dissimilar if their codes are different. Thus it is not possible to retrieve substrings from the text corpus.

6.7.4. Edit Distance

One of the most widely used textual similarity measures is the edit distance approach which is a full – text search method. Edit distance between two strings is defined as the number of changes that need to be made to one string in order to completely convert it to the other string [100]. The different full-text search methods have been classified into three different categories: neighborhood generation, exact search partitioning and intermediate partitioning [101]. Neighborhood generation techniques retrieve all patterns from the text, where the edit distance between the query pattern and the retrieved patterns is less than a given edit distance, for example, k. However, for these techniques to be effective, care should be taken that the value of k is

not too large. Exact search partitioning algorithms search for text in which the given pattern of text appears unaltered. Intermediate partitioning lies between the above two techniques. Parts of the pattern are extracted and neighborhood generation is applied to these small pieces. Since these parts are smaller their neighborhoods would be smaller than the neighborhoods of the whole pattern. Exact search is later performed on the generated pattern pieces. However the edit distance techniques are more useful in detecting spelling mistakes rather than in retrieving words that have the same meaning.

6.7.5. Sentential Similarity

Textual similarity measures are also developed for the purposes of web document retrieval. It has been proposed that the sentential information can be used for this purpose [94]. The retrieval score of the document is computed based on the similarity values between the sentences in the document and the query. The similarity between each sentence in the document and the query is computed. In addition, the similarity between the document as a whole and the query is also computed. These two similarity measures are incorporated in calculating the overall similarity score.

6.7.6. Contextual Similarity

It has been proposed that two text units can be considered similar if they focus on the same common concept or object [102]. Some measures of similarity have been suggested to detect whether two small textual units contain common information or contain information about the same topic [102, 103]. These algorithms can be classified

into knowledge based systems and word-based systems [103]. Knowledge-based systems require creation of a knowledge base and this may not be possible for domains that are not well-known. To overcome these limitations, word-based approaches have been developed [104]. Word distribution in a text is used to find a thematic segmentation. These word-based approaches are particularly suited for technical or scientific texts that are based on a specific vocabulary. Each textual unit is characterized by a set of single and compound words that form a vector. A textual unit could be short segments or paragraphs of text. Descriptor values are given by the number of occurrences of the words in the textual unit modified by the word distribution in the text. Different units of text are compared with respect to their descriptors to know if they refer to the same topic. Apart from word co-occurrence vector features can also be defined based on matching noun phrases, synonyms, and shared proper nouns [102]. Based on the existing literature, measures of textual similarity need to be adapted in order to implement the semantic retrieval module as part of this research.

6.8. Measures of Semantic Similarity between Exemplars

As mentioned earlier, it may be safe to assume that the intended use may be written in one or two sentences, whereas the rationale may be better explained in a short paragraph. If the user starts with a graph query the structural retrieval module will first retrieve exemplars that are structurally similar to the query exemplar. The semantic retrieval module will then retrieve exemplars that are semantically similar by evaluating the similarity between the textual descriptions of the intents and

rationales associated with exemplars in the case base. If the user starts with a textual query, then the same measures of similarity can be used to search for the search string in the database of semantic definitions of exemplars.

The appropriate way to evaluate the similarity between sentences would be to compare their meanings. However, there does not exist any system that can interpret the meanings of arbitrary sentences appropriately. However, the textual similarity metrics developed should capture meaning of words as much as possible in order to be effective. One way of doing so would be to develop a fixed vocabulary which can be used while annotating the exemplars with textual information of intent and rationale. However, developing an exhaustive vocabulary would be outside the scope of this research. A fixed vocabulary is developed for the purposes of validation. The vector space model and edit distance measures are used in order to evaluate the similarity between sentences.

1. Vector-space model: As mentioned above, the similarity between two pieces of textual information can be computed by computing the cosine angle between two vectors representing the pieces of information. In this case, each document is represented by a vector (wn1, wn2,…,wnm), where wnk is the weight or importance of the term tk in the document dn and M is the size of the indexing term set. In this case the weight of each term in a document is computed by multiplying the weight of each term in the indexing term set by the number of times each term occurs in the document. This value is then divided by the total number of terms in the document in order to account for the length of the document. This ensures that the length of each document does not influence

the weight of each term. Thus each document in the database is represented by a vector of the same length as the size of the indexing term set. The similarity between two documents is evaluated by computing the cosine angle between the corresponding vectors. In this case, for each exemplar the intended use and the associated rationale can be regarded as one textual unit. After the structural retrieval module retrieves exemplars that are structurally similar to the query exemplar, the semantic retrieval module will evaluate the similarity between the retrieved exemplar's textual unit and the textual units of the rest of the exemplars in the case base.

This measure of textual similarity is similar to the feature-based similarity measure useful while evaluating the structural similarity between exemplars. In this case, the features are the words that appear in the indexing feature set.

2. Edit distance: The edit distance measure provides a mathematically well-defined measure of string similarities. The edit distance measure can be used to evaluate the similarity between the textual units in order to account for words that are not present in the fixed vocabulary. It is also necessary to include this measure in order to account for random typing mistakes. The neighborhood generation approach may be the most useful edit distance measure, in order to account for intended spelling variations or mistakes. Therefore, instead of looking for exact matches, it would be more useful to look for strings that are almost similar to the search query.

Stop words such as is, an, the, etc. should not be taken into account while evaluating both the co-occurrence frequency as well as the edit

distance. This would ensure that the similarity between strings is computed with respect to relevant words. This measure of textual similarity is similar to the graph edit distance measure used while evaluating the number of changes required converting one exemplar to the other.

This task is mainly intended to identify and develop an understanding of the different ways to represent design knowledge in a computational environment in order to provide semantic support for engineering design. As mentioned earlier, identifying the most appropriate knowledge representation schema is out of scope for this research. Use of textual representation has been identified as the most appropriate representation scheme for the purposes of developing the exemplar retrieval tool.

Chapter 7

IMPLEMENTATION OF A CONJOINED RETRIEVAL SYSTEM

The primary objective of implementing a conjoined retrieval system is to provide assistance to the exemplar author in authoring exemplars for relatively large design problems. A secondary objective of the retrieval system is to provide a testing platform to conduct experiments to evaluate similarity between exemplars. Based on the different similarity measures proposed in Chapter 4 and Chapter 5, the main variables that may influence the similarity between exemplars include: type of structural filter in use, the threshold value for each structural filter, combination of structural filters, the controlled vocabulary, and the manner in which the structural and semantic similarity measures are used in conjunction. The impact of these variables on the retrieval results are discussed in detail in chapters 8, 9, and 10.

The main requirement of the conjoined retrieval system is to allow the exemplar author the flexibility to retrieve exemplars that are either structurally similar to the query, or semantically similar to the query exemplar, or both. In order to facilitate the retrieval of exemplars, three separate modules have been developed. The first module retrieves exemplars that are structurally similar to the query exemplar, whereas the second module retrieves exemplars that are semantically similar to the query exemplar. The third module will conjoin the two modules in order to retrieve exemplars that are both structurally and

semantically similar to the query exemplar. The proposed system architecture is shown in Figure 7.1.

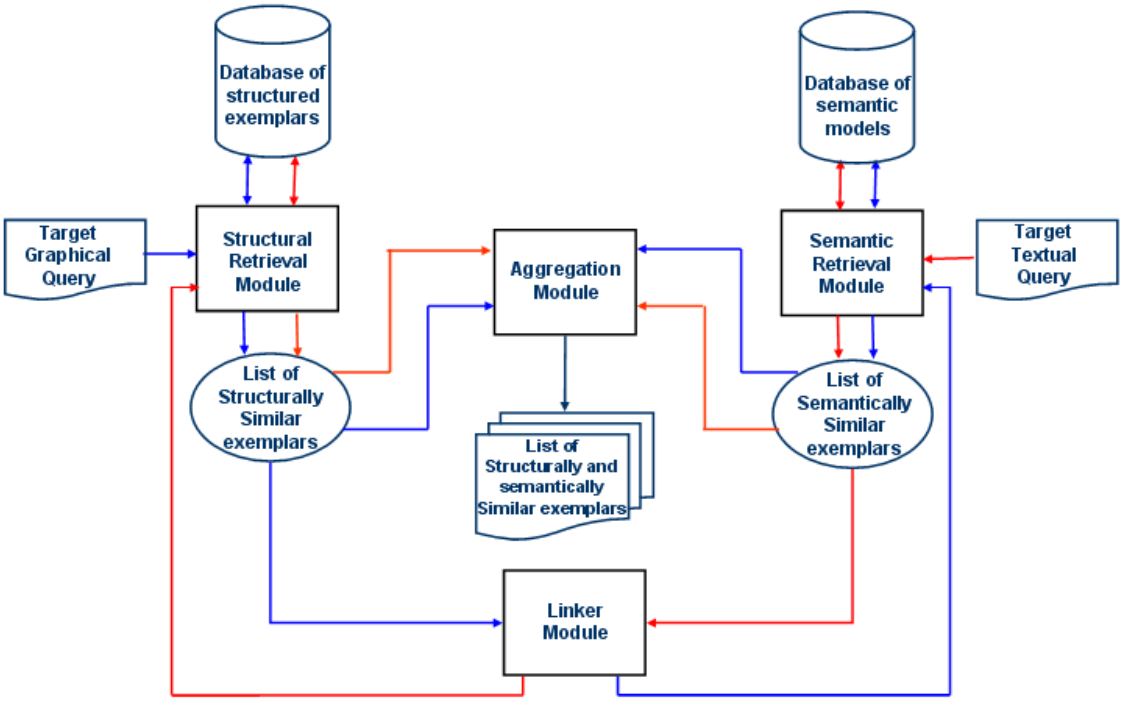


Figure 7.1: System Architecture

As can be seen from Figure 7.1, the structural retrieval module takes an exemplar graph as input. This module consists of the structural filters discussed in Chapter 4. The structural retrieval module interacts with a database of structured exemplars to return a list of exemplars that are structurally similar to the query exemplar. The semantic retrieval module takes as input the semantic description of an exemplar. This module consists of the text similarity measures proposed in Chapter 6. The semantic retrieval module interacts with the database of semantic models to retrieve a list of exemplars that

are semantically similar to the query exemplar. The linker module links the structurally similar exemplars to their corresponding semantic definitions. The aggregation module conjoins the structural similarity module and the semantic similarity module to retrieve a combined list of exemplars that are structurally similar and semantically similar to the query exemplar. In Figure 7.1 the flow of data for graph queries is shown by blue lines whereas the flow of data for text queries is shown by red lines.

7.1. Database of exemplars

The structural and semantic retrieval modules which consist of the proposed similarity measures, evaluate the similarity between the query and the exemplars in a database. For purposes of illustration only 2 exemplars are illustrated in this section.

Figure 7.2 shows a text representation of an exemplar to find thin walls in an exemplar. The exemplar consists of a solid manifold bound by 3 planes. Of the three planes, two planes represent the two sides of the wall, whereas the third plane represents the top surface. There exists a parallel and a distance relation between two planes. The distance between the two planes represents the thickness of the wall and is stored in the parameter ‘thickness’. An equation relation is applied to the parameter to check whether the thickness is less than 0.5 units.

```
Alpha Match:
        Solid Manifold Brep R1;
        Plane S1;
        Plane S2;
        Plane S3;
Alpha Extract:
        Parameter "thickness";
        Boundary (R1{S1, S2, S3});
        Parallel (S1, S3);
        Distance (S1, S3);
        Equation "eq1" (thickness < 0.5);
```

Figure 7.2: Exemplar for finding thin walls

Figure 7.3 shows an exemplar for finding circular holes in a model. The exemplar consists of a cylindrical surface which is bound by two circles. The exemplar also consists of two planes bound by one circle each. An id relation is attached to the cylindrical surface. On application of the exemplar to a model, all holes in the model will get highlighted.

```
Alpha Match:
        Cylindrical Surface S1;
        Plane S2;
        Plane S3;
        Circle C1;
        Circle C2;
        Boundary (S1 {C1, C2});
        Boundary (S2, C2);
        Boundary (S3, C1);
Alpha Extract:
        Id (S1);
```

Figure 7.3: Exemplar for finding holes

.

Other exemplars in the database include exemplars for finding features such as thin walls, holes, and bosses. The database also consists of exemplars that are authored to extract parametric information such as radius and depth of holes, reduction ratios of gears, radii of circles, and distance between parallel planes.

The semantic similarity module uses the proposed text similarity measures to evaluate the similarity between the text description of the query and text descriptions of all exemplars in the database. All text descriptions are written using Microsoft Notepad and saved with a ".des" extension. The name of the semantic description is the same as the name of the structured exemplar. Thus, for every structured exemplar in the database there exists a text description of the same name but with a ".des" extension. For example if an exemplar is named "q_radii_ratio.stp", then its semantic description is stored as a text file named "q_radii_ratio.des".

A complete description of all exemplars in the database may be found in Appendix A.

7.2. Structural Retrieval Module

As can be seen from Figure 7.1, the system architecture consists of a structural retrieval module and a semantic retrieval module. The flow of data in order to retrieve exemplars that are structurally similar to a query exemplar is shown in Figure 7.4. As can be seen from Figure 7.4, the structural retrieval module takes an exemplar graph as input. The structural retrieval module then interacts with the database of structured exemplars to

return a list of structurally similar exemplars. The implementation of each of these components and the results obtained from the experiments conducted is described in the following sections.

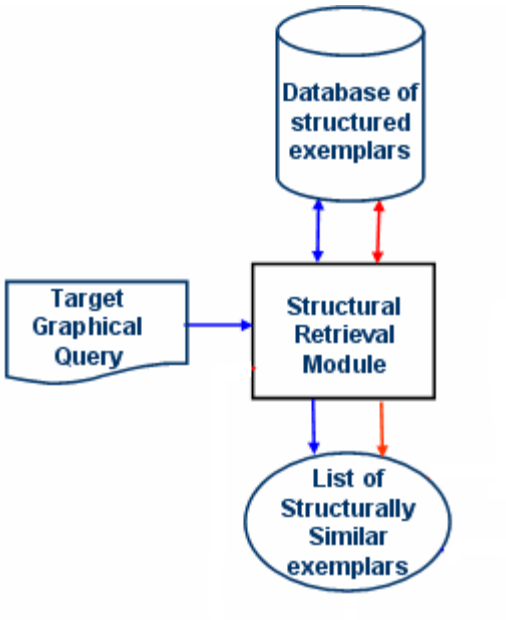


Figure 7.4: Flow of data in Structural Retrieval

7.2.1. Query Exemplar

The structural similarity module evaluates the structural similarity between a query exemplar and target exemplars in the database. Figure 7.5 shows a screen shot of a graph input. This exemplar is authored to find the distance between two planes.

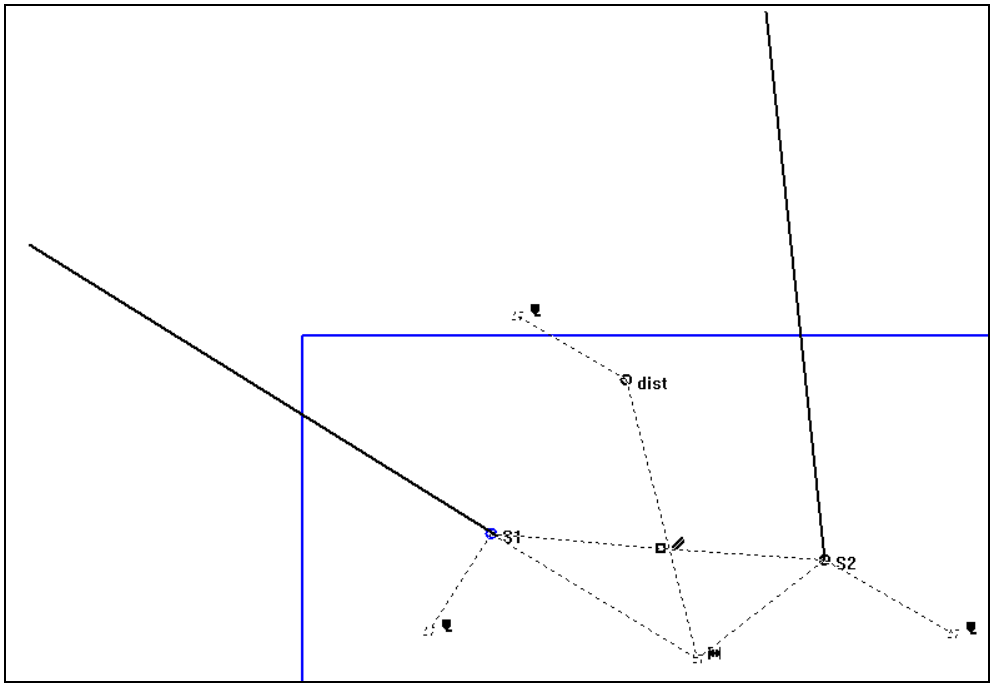


Figure 7.5: Exemplar for finding the distance between two planes

Alpha Match:
Plane "S1"
Plane "S2";
Alpha Extract:
Parameter "dist"
ID **(dist);**
ID **(S1);**
ID **(S2);**
Distance **({S1, S2}, dist);**
Parallel **(S1, S2);**

Figure 7.6: Text representation of exemplar for finding the distance between two planes

The exemplar consists of two planes with a parallel and a distance relation between them. The two planes have the match attribute since they are explicitly present in the model. The distance and parallel relations have the extract attribute. The distance relation extracts the distance between the two planes and stores it in the parameter 'dist'.

Id relations are attached to the two planes and the parameter and have the extract attribute. When this exemplar is applied to a model, the id relations help in highlighting the planes and displaying the value of the distance between them. Figure 7.6 shows a text representation of the same exemplar.

Figure 7.7 shows a screen shot of an exemplar authored to find a pair of circles that are tangent to each other. The exemplar consists of two circles that have the match attribute with a tangent relation between them. Id relations are attached to both circles which help in highlighting the circles when this exemplar is applied to a model. The tangent relation and the id relations have the attribute extract.

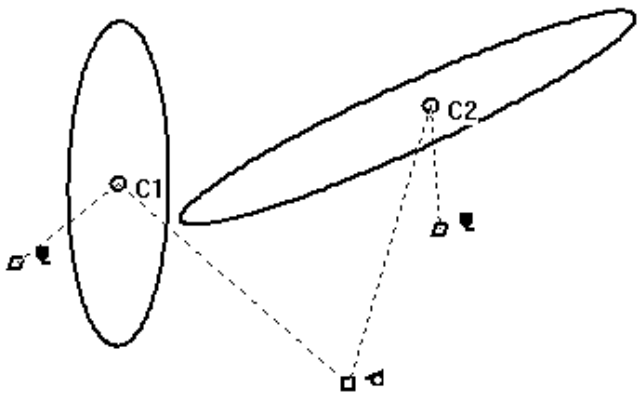


Figure 7.7: Exemplar for finding two circles tangent to each other

Alpha Match:
Circle C1
Circle C2
Alpha Extract:
Tangent **(C1, C2);**
Id **(C1);**
Id **(C2);**

Figure 7.8: Textual representation of exemplar for finding two circles tangent to each other

7.2.2. Implementation of Structural Similarity Measures

On receiving an exemplar graph as a query, the structural retrieval module uses

the proposed structural similarity measures to evaluate the structural similarity between the query and exemplars in the database. The elemental similarity measure and the attribute similarity measure serve as filters in order to prune the search space. This section describes the functionality and flow of data that occurs in this module. The different retrieval experiments that were conducted in order to test the accuracy and effectiveness of this module are discussed in Chapter 8.

As described in Chapter 4, the structural similarity module consists of entity filter, the relation filter, the attribute filter, and the pattern matching filter. Figure 7.9 shows the flow of data within the structural retrieval module.

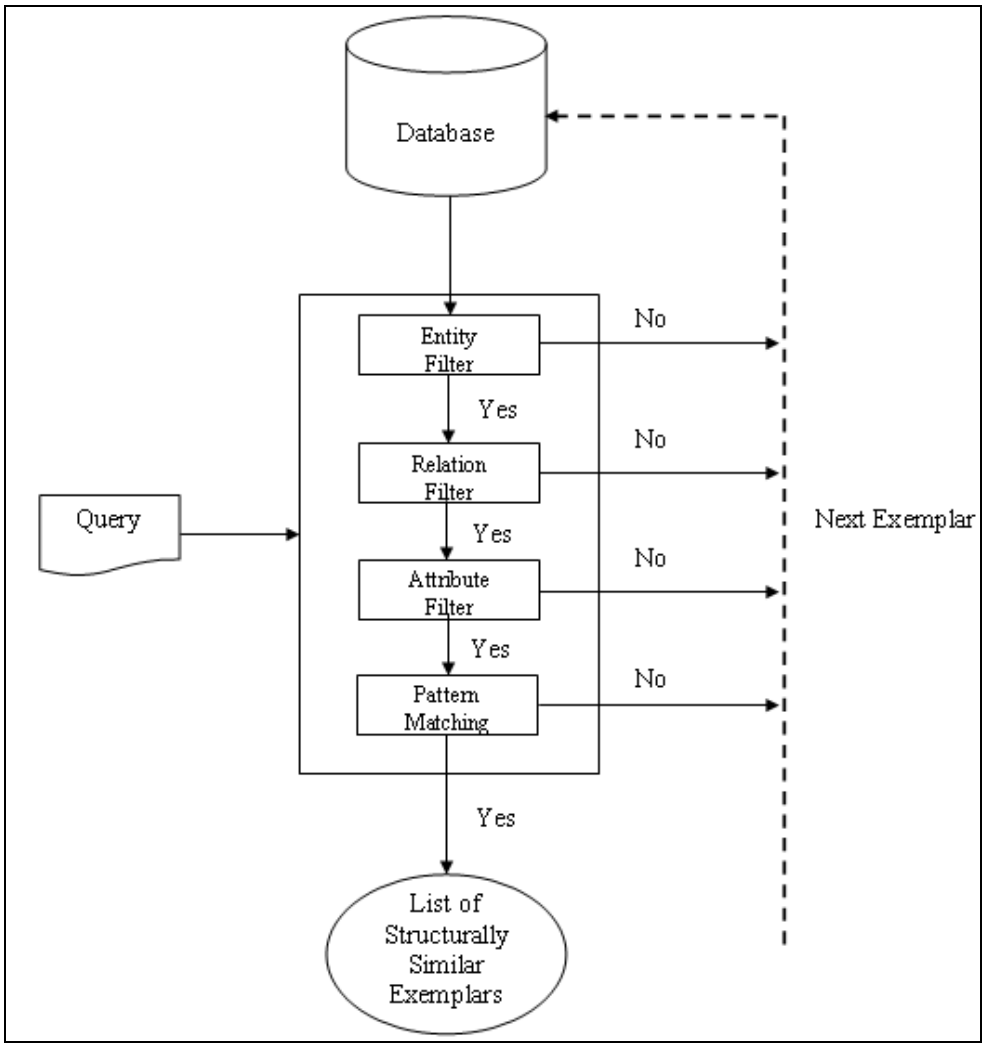


Figure 7.9: Flow of data in the structural retrieval module

As seen from Figure 7.9, the structural retrieval module takes as input a structured exemplar as query and evaluates the similarity between the query and each exemplar in the database. Each exemplar in the database is subjected to the structural filters in the order shown in Figure 7.9. The entity and relation filters eliminate those exemplars from the database that do not have at least as many entities and relations as the

query exemplar. Elimination of an exemplar implies removing the exemplar for the purposes of search and retrieval for the current query, but not deletion of the exemplar from the database altogether. The attribute filter eliminates those exemplars from the database that do not have at least as many attributes of each type as the query itself. Exemplars that pass through all filters then go through the pattern matching filter. This filter compares the way entities and relations are related in the query with the way entities and relations are related in exemplars in the database. Those exemplars that have at least half of the entities and relations related in the same manner as the query exemplar are returned to the user as matches.

Entity filter: The entity filter eliminates those exemplars from the retrieval process that do not have at least as many entities of each type as the query exemplar itself. In order to do so, the first step is to count the number of entities of each type present in the query. The second step is to count the number of entities of each type present in each exemplar in the database and compare them with the query exemplar. If the number of entities of each type present in the query is less than or equal to the corresponding number of entities in an exemplar in the database, the next step is to invoke the relation filter. The query exemplar is processed only once for information about entities and relations. Each exemplar in the database is processed for information and then compared with the query exemplar before proceeding to the next exemplar. The algorithm for the implementation of this filter is illustrated in Figure 7.10.

```
Line 1: Load Query Exemplar;
Line 2: Process Query Exemplar;
Line 3: For each entity type count number of entities;
Line 4: for (all exemplars in database)
Line 5:     Load target exemplar from database;
Line 6:     For each entity type count number of entities;
Line 7:     Compare number of entities (num_entities) of each type with those in
            query exemplar (num_entities_query);
Line 8:     if (num_entities_query ≤ num_entities) for all entity types proceed to next
            filter;
```

Figure 7.10: Algorithm for entity filter

Relation filter: The relation filter is implemented in the same manner as the entity filter. The number and types of relations in the query is counted. The next step is to count the number and types of relations in each exemplar in the database and compare with the number and type of relations in the query. If the number and type of relations in query is less than those in the exemplar in the database then the next filter is evoked or else the next exemplar in the database is processed for information. The algorithm is similar to the algorithm used to implement the entity filter and is illustrated in Figure 7.11.

```
Line 1: Load Query Exemplar;
Line 2: Process Query Exemplar;
Line 3: For each entity type count number of entities;
Line 4: for (all exemplars in the database)
Line 5:     Load target exemplar from database;
Line 6:     For each relation type count number of relations;
Line 7:     Compare number of relations (num_relations) of each type with those in
            query exemplar (num_relations_query);
Line 8:     if (num_relations_query ≤ num_relations) for all entity types proceed to
            next filter;
```

Figure 7.11: Algorithm for relation filter

Attribute Filter: The attribute filter is implemented in the same manner as the entity and relation filters. The algorithm is illustrated in Figure 7.12.

```
Line 1: Load Query Exemplar;
Line 2: Process Query Exemplar;
Line 3: For each attribute type count number of attributes;
Line 4: for (all exemplars in database)
Line 5:     Load target exemplar from database;
Line 6:     For each attribute type count number of attributes;
Line 7:     Compare number of attributes (num_attributes) of each type with those in
            query exemplar (num_attributes_query);
Line 8:     if (num_attributes_query ≤ num_attributes) for all entity types proceed to
            next filter;
```

Figure 7.12: Algorithm for attribute filter

Partial graph matching filter: If an exemplar passes though all three filters described above, the next step is to compare the manner in which entities and relations are related in the query with the manner in which they are related in the target exemplars in the database. In order to do so, the pattern matching algorithm used to match entities and relations in a CAD model is used. In this case, the CAD model is the target exemplar in the database. On application of the query exemplar to the exemplar in the database, if the manner in which entities and relations are related is the same in both the query and the target exemplar, the pattern matching algorithm returns a match. As well, the pattern matching algorithm is used to do partial graph matching between the query and the target exemplar. In order to do so, the entities and relations in the query exemplar are removed one at a time and the remaining entities and relations are matched against the target exemplar. The retrieved exemplars are ranked in the order of the num of entities and relations present in the exemplar when a match is returned. As well, since id relations are used only to highlight some elements or display the value of a parameter, all id relations are removed from the query exemplar before performing a pattern match. This ensures that those target exemplars that differ from the query exemplar only with respect to id relations don't get filtered. Similarly, fixed relations are also removed from the query exemplar since fixed relations only fix the value of some parameters. The algorithm for the pattern matching algorithm is illustrated in Figure 7.13.

```
Line 1: Load Query Exemplar;
Line 2: Process Query Exemplar;
Line 3: for (all exemplars in database)
Line 4:         Load target exemplar from database;
Line 5:         number of matches = 0;
Line 6:         while (number of entities and relations in query exemplar not equal to
                zero)
Line 7:                 remove all id and fixed relations;
Line 8:                 do pattern matching with target exemplar as model;
Line 9:                 if (match) number of matches = number of matches + 1;
Line 10:                remove one element (entity or relation) from query exemplar;
Line 11: if (number of matches > 0) add exemplar to list of structurally similar
         exemplars;
```

Figure 7.13: Algorithm for partial graph matching

The filters are implemented in such a manner that they can either be used in conjunction or they can be used one at a time. In order to use them in conjunction, the query exemplar passes through the filters in the order shown in Figure 7.9. The algorithm for using the filters in conjunction is shown in Figure 7.14.

```
Line 1: Load Query Exemplar;
Line 2: Process Query Exemplar;
Line 3: count number of entities of each type, count number of relations of each type,
        count number of attributes of each type;
Line 4: for (all exemplars in database)
Line 5:     Load target exemplar from database;
Line 6:     count number of entities of each type, count number of relations of each
            type, count number of attributes of each type;
Line 7:     if(num_entities_query ≤ num_entities) && if(num_relations_query ≤
            num_relations) && (num_attributes_query ≤ num_attributes)
Line 8:         number of matches = 0;
Line 9:         while (number of entities and relations in query exemplar not
                equal to zero)
Line 10:            remove all id and fixed relations;
Line 11:            do pattern matching with target exemplar as model;
Line 12:            if (match) number of matches = number of matches + 1;
Line 13:            remove one element (entity or relation) from query
                    exemplar;
Line 14:            go to line10
Line 15:        if (number of matches > 0) add exemplar to list of structurally
                similar exemplars;
```

Figure 7.14: Algorithm for using all filters in conjunction

The following example illustrates the implementation of the structural retrieval module. For purposes of illustration, the database can be considered to consist of only four exemplars. The first exemplar is an exemplar to find a gear and a pinion in a model. The exemplar consist of two circles; one representing the gear and the other circle representing the pinion. Radius relations are attached to each circle and the value of the

radii is stored in parameters, “pinion”, and “gear”. An equation relation is attached to the two parameters that checks whether one circle is smaller than the other. The smaller circle represents the pinion and the larger circle represents the gear. This exemplar is illustrated in Figure 7.15.

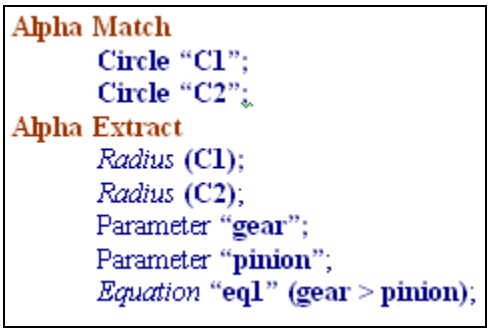


Figure 7.15: Exemplar for finding a gear and pinion

Of the three exemplars in the database, the second exemplar is authored to determine the ratio of the radii of two circles. The exemplar consists of two circles that have the match attribute. Radius relations are attached to both circles and the values of the radii are stored in parameters, “r1” and “r2”. An equation is applied to both parameters in order to determine the ratio of the radii. The value of the ratio is stored in parameter, “r_1_2”. Id relations are attached to the three parameters in order to display the value of the radii and their ratio. The third exemplar in the database is an exemplar to find distance between two points. The two points are related with a distance relation that extracts the distance between them. The value of the distance is stored in the parameter, “d1” which is displayed on application of the exemplar to a model because of the ‘id’ relation.

```
Alpha Match
        Circle "C1";
        Circle "C2";
Alpha Extract
        Radius (C1);
        Radius (C2);
        Parameter "r1";
        Parameter "r2";
        Parameter "r_1_2";
        Id (r1);
        Id (r2);
        Id (r_1_2);
        Equation "eq1" (r_1_2 = r1/r2);
```

Figure 7.16: Exemplar for finding the ratio of the radii of two circles.

```
Alpha Match
        Point "p1";
        Point "p2";
Alpha Extract
        Distance (p1, p2);
        Parameter "d1";
        Id (d1);
```

Figure 7.17: Exemplar to find distance between two points

The fourth exemplar is authored to find holes in a model. This exemplar is illustrated in Figure 7.3. For purposes of illustration, the exemplar to find gear and pinion will be used as the query exemplar. As well, all four filters will be used in conjunction. The number and type of entities and relations for each exemplar is summarized in Table 7.1.

Table 7.1: Number of entities of each type for each exemplar in database

Exemplar	Entities	Relations	Attributes
Gear and pinion (query)	two circle entities, two parameters	two radius relations, one equation relation	Two alpha match, five alpha extract
Ratio of Circles	Two circle entities, three parameters	Two radius relations, one equation, three ID relations	Two alpha match, nine alpha extract
Distance between points	Two points, one parameter	One distance, one ID relation	Two alpha match, three alpha extract
Hole	Two circle entities, two planes, one cylindrical surface	Three boundary, one ID relation	Eight alpha match, one alpha extract

When the user decides to retrieve exemplars that are structurally similar to the query exemplar, all exemplars in the database are subjected to the four filters. As can be seen from Table 7.1, the query exemplar has two circles, two parameters, two radius relations, and one equation relation. On subjecting the target exemplars in the database to the entity filter, the exemplar for finding distance between points gets filtered since it does not have any circle entities. Similarly, the exemplar for finding a hole gets filtered out since it does not have a parameter. The exemplar for finding the ratio of circles has two circle entities, and three parameters. Hence this exemplar and the query itself pass through the entity filter and are subjected to the relation filter (Table 7.3). The exemplar for finding the ratio of two circles has two radius relations, one equation relation, and one id relation and hence passes through the relation filter and is subjected to the attribute

filter (Table 7.3).

Table 7.2: Exemplars in database subjected to the relation filter

Exemplar	Entities	Relations	Attributes
Gear and pinion (query)	two circles, two parameters	two radius, one equation relation	Two alpha match, five alpha extract
Ratio of Circles	Two circles, three parameters	Two radius, one equation, three ID relations	Two alpha match, nine alpha extract

Table 7.3: Exemplars subjected to the attribute filter

Exemplar	Entities	Relations	Attributes
Gear and pinion (query)	two circles, two parameters	two radius, one equation relation	Two alpha match, five alpha extract
Ratio of Circles	Two circles, three parameters	Two radius, one equation, three ID relations	Two alpha match, nine alpha extract

In case of the attribute filter also, the target exemplar has at least as many attributes of each type as the query exemplar. Hence, both the query and the target exemplar are subjected to the pattern matching filter. The target exemplar has entities and relations related in the same manner as the query exemplar. Both exemplars have radius relations attached to two circles and the values of the radii are stored in two parameters. Hence both exemplars get retrieved after the pattern matching filter.

Different experiments were conducted in order to verify the accuracy and

effectiveness of the structural similarity filters. These experiments are described in detail in Chapter 8.

7.3. Semantic Similarity Module

On receiving a text description of an exemplar as a query, the semantic retrieval module uses the semantic similarity measures to evaluate the semantic similarity between the query and exemplars in the database. As can be seen from Figure 7.1, the system architecture consists of a structural retrieval module and a semantic retrieval module. The flow of data in order to retrieve exemplars that are semantically similar to a query exemplar is shown in Figure 7.18.

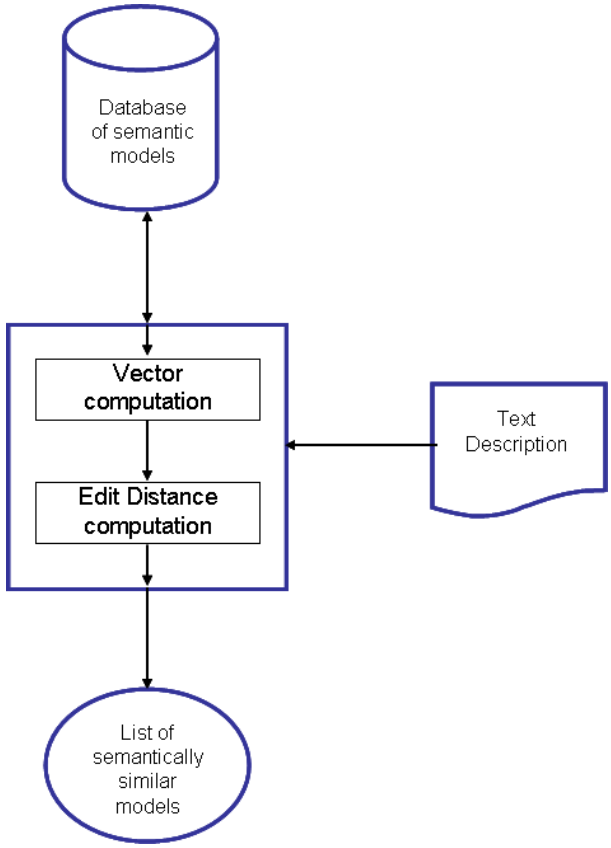


Figure 7.18: Flow of data in semantic retrieval module.

As can be seen from Figure 7.18, the semantic retrieval module takes as input a text file that contains a description of the intent and rationale of authoring an exemplar and serves as the query. For each structured exemplar in the database, there exists a text file that contains a description of the intent of the exemplar and the rationale for authoring the exemplar in a specific manner. The semantic description of each exemplar is written using a controlled vocabulary. The semantic retrieval module uses the proposed semantic similarity measures to compute the similarity between the query and the

exemplars in the database. Each of these components is described in detail in the following sections.

7.3.1. Semantic Descriptions

In order to evaluate the semantic similarity between exemplars, a text file describing the intent and rationale for authoring an exemplar in a specific manner is supplied as input to the semantic similarity module. Natural language is used to describe the intent and rationale. A controlled vocabulary has been developed for the exemplar author to use while writing the semantic description using specific words. These words are used to evaluate the similarity between descriptions of two exemplars using the vector space model. The implementation of the text similarity measures is discussed with the help of examples later in the chapter.

The controlled vocabulary contains terms that are unique to different domains. All terms that pertain to the design exemplar technology are part of the vocabulary. The other domains include casting, machining, polymer processing, and structural analysis. The vocabulary also includes terms used in CAD and terms used to describe features. The vocabulary includes singular and plural forms of terms. Both the singular and plural forms are assigned the same weights while computing the vector similarity between two descriptions. A thorough discussion of the rationale for assigning weights to the terms in the vocabulary can be found in Chapter 9. There are a total of 288 words in the vocabulary. Table 7.4 shows only a part of the vocabulary used in this research. These

terms are adapted from [105-107].

Table 7.4: Controlled Vocabulary

alpha	line	cavity	gear	stress
beta	plane	sprue	wall	strain
match	circle	riser	thin	fatigue
extract	radius	mold	hole	drill
relation	distance	pattern	boss	shaft

For example, consider the exemplar for finding the ratio of two circles illustrated in Figure 7.16. One way to write the semantic description of this exemplar is illustrated in Figure 7.19. The highlighted terms are the words that are present in the vocabulary.

*"The **intent** of this **exemplar** is to find the **ratio** of the **radii** of two **circles.** The **exemplar** consists of two **circles** that are **explicitly present** in the model. A **radius relation** is attached to each **circle.** The value of the **radius** of each **circle** is stored in a **parameter** each. An **equation relation** is applied to both **parameters** which determines the **ratio** of the two **radii** and stores the value in another **parameter. Id relations** attached to all **parameters** help display the value of the **radius** of each **circle** and the value of the **ratio**".*

Figure 7.19: Semantic description of exemplar for finding ratio of two circles

7.3.2. Implementation of Semantic Similarity Measures

The semantic similarity between the query exemplar and target exemplars in the database is computed by evaluating the similarity between the text descriptions of the query and target exemplar. In order to do so, the vector space model and the edit distance measures discussed in Chapter 6 are used.

Vector Computation: The vector space model involves computing the cosine angle between two vectors: one vector representing the query exemplar and the second vector representing the target exemplar. The size of each vector depends upon the size of the controlled vocabulary. Each document is represented by a vector (wn_1, wn_2,…,wn_m), where wn_k is the weight or importance of the term t_k in the semantic description Sn and M is the size of controlled vocabulary. Each term in the controlled vocabulary is assigned a certain weight. The weighing scheme followed for the purpose of validation is discussed in detail in Chapter 9.

The importance or weight of a term in a specific text description is influenced by the number of occurrences of the term in that document. As well, the value of the importance of a term normalized in order to account for the length of the description. Hence, if a term in the vocabulary is assigned a weight of "p", and if it occurs "n" times in a description, its importance in the description is "{(p * n)/L}", where L is the number of words in the description. The algorithm for implementing the vector space model is illustrated in Figure 7.20.

```
Line 1: Load text description;
Line 2: Covert all words to Upper Case;
Line 3: Remove all punctuation marks;
Line 4: Load Vocabulary;
Line 5: document_length = 0; term_count = 0; weight_of_term = 0;
Line 6: for every term in vocabulary;
Line 8:         Read term;
Line 9:         while (end of text file not reached)
Line 11:                Read word;
Line 12:                document_length = document_length + 1;
Line 13:                compare term in document with word in vocabulary;
Line 14:                if match
Line 16:                        weight_of_term =  assigned weight of term;
Line 17:                        term_count = term_count + 1;
Line 19:                importance = ( weight_of_term *term_count) / document_length;
Line 21:        Push weight at the back of a vector;
Line 23: Return vector;
```

Figure 7.20: Algorithm for computing vector form of a semantic representation

The above algorithm is used for representing both the query and the target exemplar in the form of a vector. The implementation of the algorithm is explained using an exemplar to find the distance between two planes. This exemplar is illustrated in Figure 7.5. One way of writing the semantic description of this exemplar is shown in Figure 7.21.

*"The intent of this exemplar is to find the **distance** between two **planes.** The exemplar consists of two **planes** with a **parallel** and a **distance relation** between the **planes** that have the **attribute extract** since they are not **explicitly present** in the model. The **parallel relation** ensures that the two **planes** are **parallel** to each other and the **distance relation extracts** the **distance** between the **two planes.** When the exemplar is applied to the model, the **distance** between the **planes** is displayed as well as the **planes** are **highlighted.**"*

Figure 7.21: Semantic Description of exemplar to find distance between two planes

This description can be represented in the form of a vector using the algorithm for vector computation and the controlled vocabulary developed for purposes of validation. Some of the words in the vocabulary along with the weights assigned to them are shown in Table 7.5.

Table 7.5: Sample vocabulary with weights

Word	**Weight**	**Word**	**Weight**
Distance	2	Planes	2
Parallel	2	Lines	2
Radius	2	Circles	2
Extracts	1.5	Relation	0.5
highlight	1.5	Match	1.5

The vector formed as a result of using this vocabulary is shown in table xxx. The size of the vector is the size of the vocabulary. Since the vocabulary contains 290 words the complete vector is not displayed in Table 7.6.

Table 7.6: Semantic description represented in vector form

0, 0, 0, ….. , 0, 0.011, 0, 0.016, 0.016, 0.0168, 0, ……, 0, 0.0674, 0, .., 0.157, 0, …, 0, 0.016, 0, …, 0, 0.016, 0, 0.005, 0,……., 0, 0.0112, 0, ……..0, ……0

.

The similarity between the query exemplar and target exemplar is evaluated by computing the cosine angle between the query vector and vector of target exemplar. The cosine angle is computed by taking the dot product of the two vectors and dividing the dot product by a product of the magnitudes of each vector. A separate function is used to compute the dot product of two vectors. The algorithm for computing the dot product of two vectors is illustrated in Figure 7.22.

```
Line 1: Load first vector;
Line 2: Load second vector;
Line 3: dot_product = 0.0;
Line 4: pointer to the first element of both vectors;
Line 5: while(pointer not pointing to null)
Line 6:         dot_product = element in first vector * corresponding element in second
                vector;
Line 7:         point to next element in both vectors;
Line 8: return dot product;
```

Figure 7.22: Algorithm for computing dot product of two vectors

Similarly, the magnitude of a vector is computed in a separate function. The algorithm used for computing the magnitude of a vector is shown in Figure 7.23.

```
Line 1: Load Vector;
Line 2: magnitude = 0.0;
Line 3: pointer to the first element in vector;
Line 4: while (pointer not pointing to null)
Line 5:         magnitude = magnitude + (element in vector)^2;
Line 6:         point to next element in vector;
Line 7; return sqrt(magnitude);
```

Figure 7.23: Algorithm for computing magnitude of vector

As mentioned earlier, the cosine angle between the query exemplar and a target exemplar is computed by dividing the dot product of the two vectors by the magnitudes of the two vectors. The higher the value of the cosine angle, the more similar the target

exemplar is to the query exemplar.

Edit Distance: The edit distance between two text descriptions is defined as the minimum number of changes required to convert one text description into the other. In this case, the number of changes required is simply calculated as the number of different words between the query description and the target description. The total number of changes required to convert one text description into the other is computed as per the algorithm shown in Figure 7.24.

```
Line 1: Load first text description;
Line 2: total_changes = 0;
Line 3: while (end of first file not reached)
Line 4:         number_of_changes = 1; n = 0;
Line 5:         read first word;
Line 6:         Load second text description;
Line 7:         while (end of second file not reached)
Line 8:         compare both words;
Line 9:         if (match)
Line 10:                        number_of_changes = 0;
Line 11:        if (number_of_changes = 0)
Line 12:                n = 0;
Line 13:        else (n = 1);
Line 14:        total_changes = total_changes + n;
```

Figure 7.24: Algorithm for computing edit distance between two semantic descriptions

As seen from the algorithm, each word from the first description is compared to

all words from the second description. If a word in the first file matches any word in the second file, the total number of changes required does not change. However, when there is no match, the total number of changes required is increased by one. The computation of the edit distance measure between two text descriptions is implemented as a separate function in the program. For example, consider the semantic description of an exemplar to find a gear and a pinion (Figure 7.25).

*This exemplar is intended to **find** a pair of **gears** in the model and determine which the pinion is and which the **gear** is. The exemplar consists of 2 **circles** that have the **attribute match.** One **circle** represents the **gear** and the other **circle** represents the pinion. The **equation relation** is applied to the **radii parameters** to **check** which the smaller **circle** is. The smaller **circle** is the pinion and the bigger **circle** is the **gear.***

Figure 7.25: Semantic description of exemplar to find a pinion and gear

It is desired to find the edit distance between this text description and a text description for an exemplar to find a pair of spur gears (Figure 7.26).

*The intent of this exemplar is to find a pair of spur **gears** in a model. The exemplar consists of two **circles** that have the **attribute match** and represent the two **gears.** An **equation** is applied to the **radius parameters** to ensure that the pinion is the smaller of the two **gears.** The **circles** are made **tangent** to each other by applying the **tangent relation.** Both the **circles** are made **incident** on a **plane** and hence are **coplanar.** When this exemplar is applied to a model, the pinion and **gear** are **highlighted** and their **radii** are displayed.*

Figure 7.26: Semantic description of exemplar for finding a pair of spur gears

The edit distance between the two exemplars is 23. This implies that the number of words that are not common to both descriptions is 23. Different experiments were conducted in order to verify the usefulness of the semantic similarity measures. The main objective of the experiments was to evaluate the influence of the weights of each term in the controlled vocabulary. These experiments along with the results obtained are discussed in detail in Chapter 9.

7.4. Aggregation Module

The structural retrieval module and the semantic retrieval module are implemented such that they are independent of each other and can be used as stand alone retrieval modules. Since they are developed to incorporate different views of similarity, these modules retrieve different exemplars when used independently. This is evident

from the results obtained from using these modules independently. These results are discussed in chapters 8 and 9. The aggregation module combines the semantic similarity module and the structural similarity module in order to retrieve a list of exemplars that are either structurally similar to the query exemplar or semantically similar or both.

The main objective of the aggregation module is to retrieve exemplars from the database that are similar to the query exemplar from different perspectives. The semantic retrieval module retrieves exemplars that differ from the query exemplar in structure but have the same purpose. This helps the exemplar author to discover different ways of authoring an exemplar meant for the same purpose. On the other hand, the structural retrieval module may retrieve exemplars that differ from the query exemplar in intent but are similar with respect to type of entities and relations and the manner in which entities are related to one another. This helps in discovering new uses for an existing exemplar. A secondary objective of combining the two modules of similarity is to expand the number of retrieved exemplars. These objectives may be achieved by using the two modules either in series or in parallel. The algorithms for using the two similarity modules are discussed in detail in the following sections.

7.4.1. Using the two modules in parallel

As mentioned before, the primary objective of combining the two retrieval modules are discovering new uses for an existing exemplar or discovering new ways of authoring exemplars meant for a specific purpose. In order to achieve this objective, both

the similarity modules are used in parallel. Using the two modules in parallel implies that the same query serves as an input to both modules. The structural retrieval module takes as input a structured exemplar as query and retrieves exemplars that are structurally similar to the query exemplar. As well, the semantic retrieval module takes as input the semantic description of the same query and retrieves exemplars that are semantically similar to the query. Thus the user is provided with a combined list of exemplars that are both structurally and semantically similar to the query exemplar. The algorithm for using the two modules in parallel is listed in Figure 7.27.

```
Line 1: Load structured query exemplar;
Line 2: for (all exemplars in database)
Line 3:             Load target exemplar;
Line 4:             Compute Structural Similarity;
Line 5:             If (structurally similar) push back in list of structurally similar
                    exemplars;
Line 6: Load semantic description of query exemplar;
Line 7: for (all exemplars in database)
Line 8:         Load target exemplar;
Line 9:         Compute Semantic Similarity;
Line 10:            If (semantically similar) push back in list of semantically similar
                    exemplars;
Line 11: return list of structurally similar exemplars;
Line 12: return list of semantically similar exemplars;
```

Figure 7.27: Algorithm for using structural and semantic similarity modules in parallel

Experiments for verifying the usefulness of using the two similarity modules were conducted. A detailed discussion of the results obtained from these experiments can be found in Chapter 10.

7.4.2. Using the two modules in series

As mentioned before, a secondary objective of using the two retrieval modules in conjunction is to expand the number of retrieved exemplars. This objective may be achieved by using the two retrieval modules in series. Using the two modules in series implies that one of the modules takes as input all the exemplars that were retrieved using the other module. For example, if the structural retrieval module is used to retrieve exemplars that are structurally similar to a query exemplar, then the semantic descriptions of each retrieved exemplar is supplied as input to the semantic retrieval module. The semantic retrieval module then retrieves a list of exemplars that are semantically similar to each of those exemplars, thus giving an expanded list of both structurally and semantically similar exemplars. Similarly, if the semantic retrieval module is used to retrieve exemplars that are semantically similar to a query exemplar, then the structured exemplars of each retrieved exemplar is supplied as input to structural retrieval module. The algorithm for using the two modules in series is listed in Figure 7.28.

```
Line 1:  Load structured query exemplar;
Line 2:  While (database not empty)
Line 3:          Load target exemplar;
Line 4:          Compute Structural Similarity;
Line 5:          If (structurally similar) push back in list of structurally similar
                 exemplars;
Line 6:  Load list of structurally similar exemplars;
Line 7:  for (every element of the list)
Line 8:      Load semantic description of exemplar;
Line 9:      Create new list for semantically similar exemplars;
Line 10:         Compute Semantic Similarity;
Line 11:         If (semantically similar) push back in list of semantically similar
                 exemplars;
Line 12: return list of structurally  similar exemplars;
Line 13: return all lists of semantically similar exemplars;
```

Figure 7.28: Algorithm for using structural and semantic similarity modules in series

Different experiments were conducted to verify the usefulness of using the two modules in series. The results obtained from these experiments are discussed in detail in Chapter 10.

Chapter 8

EXPERIMENTAL VALIDATION OF STRUCTURAL SIMILARITY FILTERS

Having implemented the structural similarity measures as described in Chapter 7, different experiments were conducted in order to study and evaluate the accuracy and effectiveness of these measures. The experiments were aimed at studying different ways to increase the effectiveness of using the filters. The similarity measures may be considered effective if the exemplar author is provided with an appropriate number of alternative exemplar configurations. If the exemplar author is provided with a large number of alternative configurations, it may be a relatively tedious task for the exemplar author to evaluate which of the retrieved exemplars satisfies most of his requirements. Similarly, if the similarity measures retrieve only a few alternative configurations, then there is a possibility that the exemplar author may not be able to adapt any of the retrieved exemplars to satisfy the new requirements. Studies in human factors suggest that a user should be provided 7 ± 2 alternative configurations [108]. A secondary objective was to verify whether the filters performed as intended. This helped in verifying whether the algorithms were implemented correctly.

8.1. Database

For purposes of conducting these experiments, the database consisted of thirty

exemplars (Appendix A). Table 8.1 lists the exemplars that were part of the database along with a brief explanation of each. The exemplars used in this database were chosen from exemplars that have been authored by different people in the Automation in Design Group (AID) at Clemson University, and the Design Automaton Laboratory (DAL) at Arizona State University. Some of the exemplars in the database were specifically authored for purposes of validation of this research. The database was specifically designed to consist of exemplars having different entities and relations with respect to number and type. This ensures that the database has breadth with respect to the types of exemplars present in the database. As well, the exemplars that were part of the database were authored for different purposes.

As seen from Table 8.1, eight exemplars were specifically authored to be part of the database. The remaining twenty two exemplars were chosen from previously authored exemplars. As well, multiple variations of exemplars authored for the same purpose were included in the database in order to validate the semantic similarity measures. For example, exemplars 24 to 29 were authored to find thin walls in a model. However, all these exemplars are structurally different with respect to number and types of entities and relations they have. Similarly, exemplars 2 and 8 can both be used for finding the distance between two planes. However, exemplar 8 has got more entities and relations than exemplar 2. Exemplars 18 and 19 can be used to find the radius and depth of a hole, but they differ from each other in terms of entities and relations.

Table 8.1: List of Exemplars in database

Exemplar Number	Exemplars	Year
W1	**2_lines_dist_2_planes.stp**: This exemplar finds two planes and two lines in a model, and determines the distance between the lines that are incident on the two planes	SA, Clemson, 2008
W2	**2_planes_with_distance.stp:** This exemplar finds the distance between two planes.	SA, Clemson, 2008
S1	**2_points.stp:** This exemplar finds two points in a model	SA, Clemson, 2008
S2	**2_points_2_lines_with_dist.stp:** This exemplar finds two points and two lines in model and determines the distance between the two lines.	SA, Clemson, 2008
S3	**2_points_with_distance.stp:** This exemplar finds the distance between two points.	SA, Clemson, 2008
G1	**gear_pinion_02.stp:** This exemplar finds a pair of gears and determines which the gear is and which the pinion is.	JDS, ASU, 2002
G2	**gears_double_ratio.stp:** This exemplar finds the gear ratio of a gear train consisting of five gears	JDS, Clemson, 2003
W3	**planes_lines_points_distance.stp**: This exemplar finds the distance between two planes.	SA, Clemson, 2008
B1	**q_belt_radii.stp:** This exemplar is used to size a transmission belt	JDS, Clemson, 2003

F1	**q_boss_radius.stp:** This exemplar finds the radius and height of a cylindrical boss	JDS, Clemson, 2003
G3	**q_compound_5_gears.stp:** This exemplar sizes a gear train of 5 gears	BB, ASU, 2003
F2	**q_connecting_rod_thick_enough.stp:** This exemplars determines if a connecting rod is thick enough	AD, Clemson, 2003
G4	**q_coplanar_gears.stp:** This exemplar finds a pair of coplanar gears	JDS, ASU, 2002
F3	**q_cylinder.stp:** This exemplar finds the radius and height of a cylinder	AD, Clemson, 2003
W4	**q_dist_bounded_planes.stp**: this exemplar finds the distance between two planes of a solid manifold	SA, Clemson, 2008
M1	**q_double_radius.stp:** This exemplar doubles the radius of a circle	JDS, ASU, 2002
F4	**q_hole.stp:** This exemplar finds cylindrical holes in a model	JDS, ASU, 2000
F5	**q_hole_depth_radius.stp:** This exemplar finds the radius and depth of a cylindrical hole.	JDS, ASU, 2000
F6	**q_hole_radius_depth.stp:** This exemplar finds the radius and depth of a cylindrical hole. This exemplar has fewer entities and relations than exemplar 18.	JDS, ASU, 2000

S4	**q_radii_ratio.stp:** This exemplar finds the ratio of the radii of two circles	AD, Clemson, 2002
S5	**q_radii_ratio_w_large_check.stp:** This exemplar finds the ratio of radii of two circles and determines which the larger circle is.	AD, Clemson, 2002
F7	**q_radius_cylindrical_hole.stp:** This exemplar finds the radius of cylindrical holes.	SA, Clemson, 2008
S6	**q_tangent_circles.stp:** This exemplar finds the radii of two circles tangent to each other	AD, Clemson, 2002
W5	**q_thinwall_medium_with_boundary.stp:** This exemplar finds thin walls in a model.	JDS, ASU, 2000
W6	**q_thinwall_parallel_planes.stp:** This exemplar finds thin walls where the two sides of the wall are represented as two parallel planes	SA, Clemson, 2008
W7	**q_thinwall_simple_boundary.stp:** This exemplar finds thin walls in a model	JDS, ASU, 2000
W8	**q_thinwall_solid_bound_parallel_planes.stp:** This exemplar finds thin walls in a model.	JDS, ASU, 2000
W9	**q_thinwall_thick_less_0.5.stp:** This exemplar finds thin walls in a model where a wall is represented as a set of parallel planes and is considered thin if the distance between the planes is less than 0.5 units	JDS, ASU, 2000
W10	**q_thinwall_two_loops_simple.stp:** This exemplar finds thin walls in a model	JDS, ASU, 2000

G5	**q_spur_gears.stp:** This exemplar finds a pair of spur gears in a model and finds the radius of the two gears	JDS, Clemson, 2002

The number of entities and relations of each type in each exemplar is summarized in Table 8.2. This information is useful in verifying the accuracy of the structural similarity measures. The semantic description of each exemplar is listed in Appendix A.

Table 8.2: Number and type of entities, relations, and attributes in each exemplar of database

Exemplar Number	Entities	Relations	Attributes
W1	2 lines, 2 planes, 1 parameter	1 distance, 1 parallel, 2 incident, 3 id	4 alpha match, 8 alpha extract
W2	2 planes, 1 parameter	1 distance, 1 parallel, 3 id	2 alpha match, 6 alpha extract
S1	2 points		2 alpha match
S2	2 lines, 2 points, 1 parameter	1 distance, 1 parallel, 1 id	4 alpha match, 4 alpha extract
S3	2 points, 1 parameter	1 distance, 1 id	2 alpha match, 3 alpha extract
G1	2 circles, 2 parameters	2 radius, 1 equation	2 alpha match, 5 alpha extract
G2	5 circles, 9 parameters	5 radius, 8 id, 5 equations	5 alpha match, 27 alpha extract
W3	3 lines, 2 planes, 2 points, 1 parameter	1 right angle, 6 incident, 1 distance, 1 parallel, 1 id	6 alpha match, 12 alpha extract
B1	2 lines, 4 parameters	1 distance, 1 parallel, 3 equation	2 alpha match, 9 alpha extract

F1	1 cylindrical surface, 2 planes, 2 circles, 2 parameters	3 boundary, 1 distance, 1 parallel, 1 radius, 1 equation	8 alpha match, 6 alpha extract
G3	5 circles, 6 parameters	5 radius, 5 id, 3 equations	5 alpha match, 19 alpha extract
F2	2 cylinders, 2 circles, 1 plane, 3 parameters	1 coincident, 1 parallel, 4 incident, 2 radius, 2 equation	4 alpha match, 14 alpha extract
G4	2 circles, 1 plane	2 coincident, 2 parallel, 2 id	2 alpha match, 7 alpha extract
F3	1 cylindrical surface, 2 planes, 2 circles, 2 parameters	3 boundary, 1 distance, 1 parallel, 1 radius	8 alpha match, 5 alpha extract
W4	1 solid manifold, 2 planes, 1 parameter	1 boundary, 1 distance, 1 parallel, 1 id	4 alpha match, 4 alpha extract
M1	1 circle, 3 parameters	2 radius, 2 equation	1 alpha beta match, 1 alpha beta extract, 3 alpha extract, 3 beta extract
F4	1 cylindrical surface, 2 planes, 2 circles	3 boundary, 1 id	8 alpha match, 1 alpha extract
F5	1 cylindrical surface, 2 planes, 2 circles, 1 line, 2 points, 2 parameters	4 boundary, 2 incident, 1 coincident, 1 distance, 1 parallel, 1 radius	8 alpha match, 11 alpha extract
F6	1 cylindrical surface, 2 planes, 2 circles, 2 parameters	3 boundary, 1 distance, 1 radius	8 alpha match, 4 alpha extract
S4	2 circles, 3 parameters,	2 radii, 3 id, 1 equation	2 alpha match, 9 alpha extract
S5	2 circles, 3 parameters	2 radius, 3 id, 2 equation	2 alpha match, 10 alpha extract

F7	2 planes, 2 circles, 1 cylindrical surface, 1 parameter	4 incident, 1 id, 1 radius	5 alpha match, 7 alpha extract
S6	2 circles, 2 parameters	2 radius, 1 tangent, 2 id, 1 equation	2 alpha match, 8 alpha extract
W5	1 solid manifiold, 3 planes, 2 vectors, 2 parameters	1 distance, 1 parallel, 1 angle, 2 surface normals, 3 equations	4 alpha match, 12 alpha extract
W6	2 planes	1 parallel	2 alpha match, 1 alpha extract
W7	1 solid manifold, 3 planes, 2 vectors, 3 parameters	1 boundary, 2 surface normals, 1 opposite direction, 1 distance, 1 parallel, 2 equations	5 alpha match, 12 alpha extract
W8	1 solid manifold, 3 planes	1 boundary, 1 parallel	5 alpha match, 1 alpha extract
W9	1 solid manifold, 3 planes, 1 parameter	1 boundary, 1 parallel, 1 distance, 1 equation	5 alpha match, 4 alpha extract
W10	1 solid manifold, 3 planes, 2 parameters, 2 surface normals	1 boundary, 1 opposite direction, 1 parallel, 2 id, 1 equation	5 alpha match, 13 alpha extract
G5	2 circles, 1 plane, 2 parameters	2 radius, 1 equation, 2 incident, 1 tangent	2 alpha match, 9 alpha extract

Four exemplars from the database were chosen as query exemplars for all the experiments. The first query was to find the distance between two planes as shown in Figure 7.5 and reproduced in Figure 8.1. This is “exemplar 2” in the database and is named “2_planes_with_distance.stp”. The second query was to find a gear and a pinion

as illustrated in Figure 7.15 and reproduced in Figure 8.2. This query is "exemplar 6" in the database and has the name "gear_pinion_02.stp". The third query was to find holes in a model. This exemplar is described in Chapter 7 and reproduced in Figure 8.3. This exemplar is named "q_hole.stp" and is "exemplar 17" in the database. The fourth query was to find thin walls in a model as described in Chapter 7. The same exemplar is shown in Figure 8.4. This exemplar is named "q_thinwall_thick_less_0.5.stp" and is "exemplar 28" in the database. These exemplars are highlighted in yellow in Table 8.2.

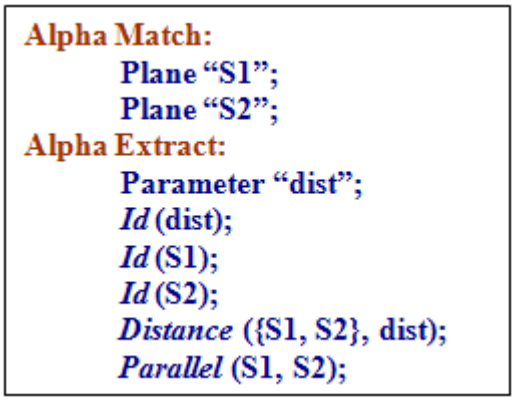


Figure 8.1: 2_planes_with_distance.stp

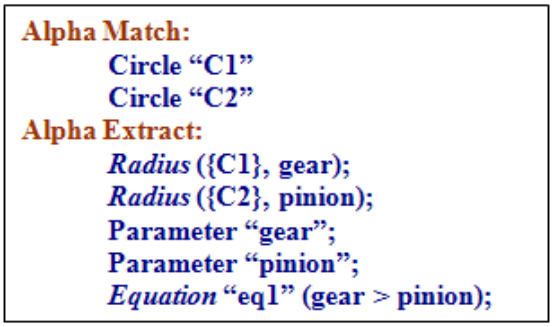


Figure 8.2: gear_pinion_02.stp

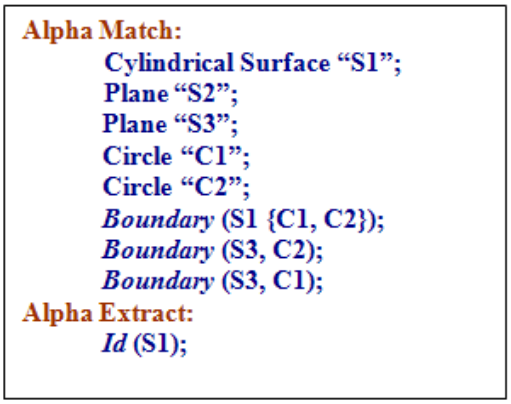


Figure 8.3: q_hole.stp

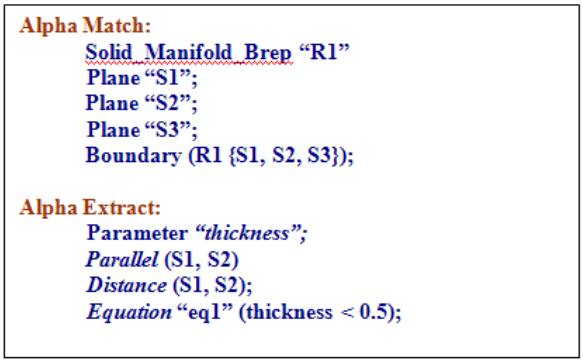


Figure 8.4: q_thinwall_thick_less_0.5.stp

Of these four exemplars, only the exemplar "2_planes_with_distance.stp" was authored specifically for purposes of validation of this research. The remaining three exemplars were previously authored by other exemplar authors. As seen from Table 8.2, all queries were different from each other in terms of entities and relations and in terms of intended use. Since the types entities and relations present in these queries cover the different types of entities and relations present in the remaining exemplars of the database, these four queries can be considered to be representative of the entire database. As well, there are multiple variations of each of these exemplars in the database. Hence, using these exemplars as queries provides a good basis for testing both the structural similarity measures and semantic similarity measures.

8.2. Experiment 1: Verifying accuracy of all filters

The objective of this experiment was to verify that all the structural filters performed as intended. This experiment was aimed at validating whether the implemented algorithms performed as expected. For this experiment, all structural filters were used in conjunction as described in Chapter 7. This implies that all exemplars in the database were subjected to the entity filter, relation filter, attribute filter, and partial graph-matching filter.

In this case, it was desired to retrieve all exemplars that were structurally similar to each of the four queries, "2_planes_with_distance.stp", "gear_pinion_02.stp", "q_hole.stp", and "q_thinwall_thick_less_0.5.stp". In this case, the target exemplars that

do not have at least as many entities, relations, and attributes of each type as the query exemplar, were filtered out. All exemplars that passed through these filters were then subjected to the partial graph-matching filter. The accuracy of the structural similarity filters were verified by theoretically evaluating, which exemplars will get filtered out due to the structural filters.

The query "2_planes_with_distance.stp" has two planes and one parameter. The two planes have the attributes "alpha match" and the parameter has the attribute "alpha extract". Based on the number of entities and relations in each exemplar shown in Table 8.2, it can be seen that exemplars W1, W2, W3, F1, F3, W4, F5, F6, F7, W5, W7, W9, and W10 pass through the entity filter. The names of the exemplars can be read from Table 8.1 based on the exemplar numbers. As well, the query has one distance relation, one parallel relation, and three id relations. Of all the exemplars that passed through the entity filter, only exemplars W1 and W2 pass through the relation filter. The two planes in the query have the attributes "alpha match", whereas the remaining entities and relations have the attributes "alpha extract". Exemplar W1 has two planes and two lines with attributes "alpha match" and the remaining eight entities and relations have the attribute "alpha extract".

Having passed through the entity, relation, and attribute filters, exemplars W1 and W2 are subjected to the partial-graph matching filter. As mentioned in the algorithm, all id relations are removed from the query. That leaves a total of five entities and relations in the query. A full graph match is performed using the five entities and relations in the

query. Exemplar W1 has a distance and parallel relation between two lines. Hence a full graph match does not return a match. Next, a partial graph match is performed by removing one entity or relation at a time. In order to perform a partial graph match, it is necessary to have at least one entity or relation in the query. This implies that a partial graph match can be performed a maximum of four times, since a total of five entities and relations are present in the query. On removing the distance and parallel relations from the query exemplar, a match is returned. Three entities and relations are left in the query when a match is returned. The similarity score given to exemplar 1 is calculated by dividing the number of entities and relations left in the query exemplar by the maximum number of times a partial graph match can be performed. Hence, in this case, the similarity score given to exemplar 1 is 0.75.

Similarly, the query "gear_pinion_02.stp" has two circles, two parameters, two radius relations, and one equation relation. Based on the number and types of entities in the query, target exemplars G1, G2, F1, G3, F2, F3, F5, F6, S4, S5, S6, and G5 pass through the entity filter since they all have at least two circles and two parameters. This can be observed from Table 8.2. Exemplars F1, F3, F5, and F6 have only one radius relation and hence do not pass through the relation filter. The remaining exemplars pass through the attribute filter because they have at least as many entities and relations with the "alpha match" and "alpha extract" entities as the query exemplar. All the target exemplars that have passed through the entity, relation, and attribute filters are subjected to the partial graph matching filter. All target exemplars except F2 return a full graph

match and hence have a similarity score of 1. Exemplar F2 has a similarity score of 0.5 since a total of three entities and relations out of a maximum of six, are left in the query exemplar when a match is returned.

For the query "q_hole.stp", exemplars F1, F2, F3, F4, and F7 pass through the entity filter since they all have at least the same number of entities of each type as the query itself. However, exemplars F1, F2, and F3 do not pass through the relation filter since they do not have an id relation, whereas exemplar F7 does not have a boundary relation. Since, exemplar F4 is the query itself, the partial graph matching filter returns a full graph match and it has a similarity score of 1. Similarly, the query "q_thinwall_thick_less_0.5.stp", exemplars W7, W9, and W10 are retrieved after passing through the filters. Table 8.3 shows the target exemplars passing through each filter for all queries.

Table 8.3: Exemplars passing through each filter

Query	Entity Filter	Relation Filter	Attribute Filter	Partial Pattern Matching Filter
2_planes_with_distance	W1, W2, W3, F1, F3, W4, F5, F6, F7, W5, W7, W9, W10	W1, W2	W1, W2	W1, W2
gear_pinion_02	G1, G2, F1, G3, F2, F3, F5, F6, S4, S5, S6, G5	G1, G2, G3, F2, S4, S5, S6, G5	G1, G2, G3, F2, S4, S5, S6, G5	G1, G2, G3, F2, S4, S5, S6, G5
q_hole	F1, F2, F3, F4, F7	F4	F4	F4
q_thinwall_thick_less_0.5	W5, W7, W9	W5, W7, W9	W5, W7, W9	W5, W7, W9

As can be seen from Table 8.4, in case of the query "q_hole.stp", the structural filters only retrieve the query itself. The number of retrieved exemplars for the query "q_thinwall_thick_less_0.5.stp" is three whereas the number of retrieved exemplars for query "gear_pinion_02.stp" is eight. The query "2_planes_with_distance.stp" retrieves two exemplars. The accuracy of the structural similarity filters is further verified by the fact that all queries retrieve themselves. The similarity score is used to rank the retrieved exemplars. In this experiment, all retrieved exemplars with the same similarity score are considered equally similar to the query exemplar.

In case of the query “q_hole.stp”, the filters only retrieve the query itself. As well, in case of “q_thinwall_thick_less_0.5.stp” only two exemplars in addition to the query exemplar get retrieved. Similarly, in case of the query, “2_planes_with_distance.stp”, only one more exemplar in addition to the query exemplar gets retrieved. In all these cases, the retrieval results are accurate.

The structural filters cannot be considered to be entirely effective for this database, since the exemplar author is provided with only a few alternative configurations. One reason for the number of exemplars retrieved being small may be that the structural similarity filters may be too restrictive for this database and these queries. Having verified the accuracy of the structural similarity measures, the next objective should be to understand similarity with respect to graph-based models. This can be done by modifying or tuning the different structural similarity filters and studying the results of retrieval.

Table 8.4: Results of using all filters

Query	Retrieved Exemplars	Number of retrieved exemplars
2_planes_with_distance	2_planes_with_distance (1)	2
	2_lines_dist_2_planes (0.75)	
gear_pinion_02	gear_pinion_02 (1)	8
	q_radii_ratio (1)	
	q_radii_ratio_w_large_check (1)	
	q_tangent_circles (1)	
	spur_gears_02 (1)	
	gears_double_ratio_03 (1)	
	q_compound_5_gears (1)	
	q_connecting_rod_thick_enough (0.5)	
q_hole	q_hole (1)	1
q_thinwall_thick_less_0.5	q_thinwall_medium_with_boundary (1)	3
	q_thinwall_simple_boundary (1)	
	q_thinwall_thick_less_0.5 (1)	

The following experiments describe the experiments conducted by modifying the restrictiveness of the filters and the results obtained by doing so. These results are then evaluated in order to better understand similarity with respect to design exemplars.

8.3. Experiment 2: Modifying the restrictiveness of all filters together

In this set of experiments, the entity, relation, and attribute filters were made less restrictive. The main objective of doing so was to evaluate the impact that each filter has on the results of retrieval.

In the previous experiment, target exemplars that do not have at least as many entities, relations and attributes of each type as the query exemplar were filtered out from the database. This implies that the difference between the number of each type of entity, relation, and attribute of a target exemplar and the number of each type of entity, relation, and attribute of the query exemplar was more than or equal to zero. In this set of experiments, the difference was allowed to be less than zero. This implies that even those target exemplars that do not have at least the same number of entities, relations, and attributes of each type as the query exemplar were allowed to pass through the filters and subjected to the partial graph matching filter. This condition is expressed by equations 1, 2, and 3.

(number of entities in target – number of entities in query) ≥ difference – equation 1
(number of relations in target – number of relations in query) ≥ difference – equation 2
(number of attributes in target – number of attributes in query) ≥ difference – equation 3

The value that the variable "difference" can take would have to be less than zero, in order to let target exemplars having fewer entities, relations, and attributes than the query exemplar to pass through the structural similarity filters. The

algorithm for the entity filter is as illustrated in Figure 8.5.

```
Line 1: Load Query Exemplar;
Line 2: Process Query Exemplar;
Line 3: For each attribute type count number of attributes;
Line 4: for (all exemplars in database)
Line 5:     Load target exemplar from database;
Line 6:     For each entity type count number of entities;
Line 7:     Compare number of entities (num_entities) of each type with those in
            query exemplar (num_entities_query);
Line 8:     if((num_entities - num_entities_query) ≥ difference) for all entity types
            proceed to next filter;
```

Figure 8.5: Algorithm for making entity filter less restrictive

The main reason of making the entity, relation, and attribute filters less restrictive is that the database may contain some exemplars that may not pass the filters but may pass the partial graph match filter. For example, consider a query exemplar having two planes with parallel and distance relations between them. A target exemplar having two planes with just a parallel relation between them will not pass the relation filter. However the same exemplar will pass the partial graph match filter.

In the previous experiment, the value of "difference" was zero. In this case, the different values that "difference" could assume were -5, -3, and -1. This implies, that if the value of "difference" was -1, then those target exemplars that have one less entity of each type as compared to number of entities of each type in query exemplar were allowed

to proceed to the relation filter.

The primary objective of choosing these values is to increase the effectiveness of the structural similarity measures by letting more exemplars pass through the entity, relation, and attribute filters and subjecting them to the partial graph matching filter. However, it should be noted that by choosing higher values, there is a possibility that many exemplars from the database may be retrieved. Providing the exemplar author a large number of exemplars similar to the query exemplar may prove tedious to the exemplar author in evaluating which exemplar to adapt.

In the first experiment, the value of the variable "difference" was the same for the entity, relation, and attribute filters. However, the variable "difference" may assume different values for different filters. This is explored in further experiments. The number of exemplars retrieved obtained for this experiment for different values of "difference" are shown in Table 8.5. This table also lists the exemplars that were retrieved when none of the entity, relation, and attribute filters were in place and the target exemplars were subjected to the partial graph-matching filter directly. The actual exemplars retrieved in each case are listed in parentheses.

Table 8.5: Results for different levels of restrictiveness

Query	No Filters	Difference = -5	Difference= -3	Difference = -1
2_planes_with_distance	19 (63.3 %) (W1, W2, W3, F1, F2, G4, F3, W4, F4, F5, F6, F7, W5, W6, W7, W8, W9, W10, G5)	13 (43.3 %) (W1, W2, F1, F2, F3, F4, F5, F6, F7, W5, W7, W8, W9,)	13 (43.3 %) (W1, W2, F1, F2, F3, F4, F5, F6, F7, W5, W7, W8, W9,)	3 (10%) (W1, W2, W7)
gear_pinion_02	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	13 (43.3 %) (G1, G2, F1, G3, F2, F3, F5, F6, S4, S5, F7, S6, G5)	12 (40 %) (G1, G2, F1, G3, F2, F3, F5, F6, S4, S5, S6, G5)
q_hole	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)
q_thinwall_thick_less_0.5	5 (16.7 %) (W5, W7, W8, W9, W10)	5 (16.7 %) (W5, W7, W8, W9, W10)	4 (13.3 %) (W5, W7, W8, W9)	3 (10%) (W5, W7, W9)

The number of exemplars retrieved obtained for this experiment and experiments conducted for different values of "difference" are shown in Table 8.5. This table also lists the exemplars that were retrieved when none of the entity, relation, and attribute filters were in place and the target exemplars were subjected to the partial graph-matching filter

directly. The actual exemplars retrieved in each case are listed in parentheses.

As can be seen from Table 8.5, in case of the query "2_planes_with_distance.stp", a total of nineteen exemplars get retrieved when there are no filters, six more than when the value of the variable "difference" is – 5 and -3. However, when the value of "difference" is -1, there is a substantial reduction in the number of exemplars retrieved for the query "2_planes_with_distance". For the query "gear_pinion_02.stp", there is no difference in the number of exemplars retrieved when the value of the variable "difference" is -5 or -3. Eleven exemplars get retrieved when the value of "difference" is -1. Similarly, for the query "q_hole.stp", the same exemplars get retrieved irrespective of the value of "difference". For the query "q_thinwall_thick_less_0.5.stp", there is no significant difference in the number of exemplars getting retrieved for different values of "difference". These results, when compared to the results shown in Table 8.4, suggest that the number of exemplars retrieved increases or remains the same as the value of the variable "difference" changes from 0 to -5. However, for higher values of "difference", the number of retrieved exemplars does not change for all queries except for the query "2_planes_with_distance". This trend may suggest that making the structural filters less conservative has an impact on the number of exemplars retrieved only till a certain limit. These results suggest that this limit is -3. However, more experiments need to be conducted on each structural filter separately in order to make a conclusion. The next set of experiments was aimed at evaluating how restrictive should each of the structural filters be while retrieving exemplars. These experiments are described in the next section.

8.4. Experiment 3: Modifying the restrictiveness of each filter separately

This set of experiments was conducted in order to evaluate which filters were most influential in the retrieval process. This will provide an insight into the contribution of each filter in pruning the database and also evaluate how restrictive should each filter be. In order to do so, the restrictiveness of each filter was varied using the same values of the variable "difference" as in the previous experiment. This ensures that the results obtained from the two sets of experiments are comparable. For each value of the variable "difference", experiments were conducted with only one filter in place before being subjected to the partial graph matching filter.

8.4.1. Entity Filter

The results of the retrieval process with the entity filter are summarized in Table 8.6. A value of zero for the variable "difference" implies that those target exemplars not having at least the same number of entities of each type as the query exemplar do not pass the filter. These results show that when the value of the variable "difference" is -5 or -3 or -1, the results of retrieval are the same for all queries. In this case as well, the number of exemplars retrieved for the query "q_hole" does not change with the value of the variable "difference". As well, the number of exemplars retrieved for the query "2_planes_with_distance" does not change except when the value of the variable "difference" is 0. This implies that there is a difference in the number of retrieved exemplars only when all target exemplars that do not have at least as many entities of

each type as the query exemplar are filtered out from the database.

Table 8.6: Number of exemplars retrieved with only entity filter

Query	Diff = -5	Diff = -3	Diff = -1	Diff = 0
2_planes_with_distance	19 (63.3 %) (W1, W2, W3, F1, F2, G4, F3, W4, F4, F5, F6, F7, W5, W6, W7, W8, W9, W10, G5)	19 (63.3 %) (W1, W2, W3, F1, F2, G4, F3, W4, F4, F5, F6, F7, W5, W6, W7, W8, W9, W10, G5)	19 (63.3 %) (W1, W2, W3, F1, F2, G4, F3, W4, F4, F5, F6, F7, W5, W6, W7, W8, W9, W10, G5)	13 (43.3 %) (W1, W2, F1, F2, F3, F4, F5, F6, F7, W5, W7, W8, W9,)
gear_pinion_02	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	12 (40 %) (G1, G2, F1, G3, F2, F3, F5, F6, S4, S5, S6, G5)
q_hole	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)
q_thinwall_thick_less_0.5	5 (16.7 %) (W5, W7, W8, W9, W10)	5 (16.7 %) (W5, W7, W8, W9, W10)	5 (16.7 %) (W5, W7, W8, W9, W10)	4 (13.3 %) (W5, W7, W8, W9)

The results obtained from using the entity filter alone show that for the queries "2_planes_with_distance", and "gear_pinion_02", the entity filter prunes the database only when it is most restrictive. This implies that the number of exemplars retrieved for

all queries is satisfactory when the value of the variable "difference" is zero. When the value of "difference" is -5, -3 or -1, the number of retrieved exemplars increases, which is expected. However, for the queries "2_planes_with_distance.stp" and "gear_pinion_02.stp", the exemplar author may find it tedious to evaluate which of the retrieved exemplars is the most useful. These results suggest that for the entity filter to be useful, the value of the variable "difference" should be highly zero.

The results obtained from the entity filter may change if the hierarchy of the entities is taken into consideration. For example, "curve" is a type of entity. However, the entities "circle" and "line" are types of curves. Hence, if the entities "circle" and 'curve" are considered similar, then more exemplars will pass through the entity filter. However, more experiments need to be conducted in the future to verify this.

8.4.2. Relation filter

The results of the retrieval process with just the relation filter are summarized in Table 8.7. A value of zero for the variable "difference" implies that those target exemplars not having at least the same number of relations of each type as the query exemplar do not pass the filter.

Table 8.7: Number of exemplars retrieved with only relation filter

Query	Difference = -5	Difference = -3	Difference = -1	Difference = 0
2_planes_with_distance	19 (63.3 %) (W1, W2, W3, F1, F2, G4, F3, W4, F4, F5, F6, F7, W5, W6, W7, W8, W9, W10, G5)	19 (63.3 %) (W1, W2, W3, F1, F2, G4, F3, W4, F4, F5, F6, F7, W5, W6, W7, W8, W9, W10, G5)	5 (16.7 %) (W1, W2, G4, W8, W10)	3 (10 %) (W1, W2, W10)
gear_pinion_02	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	9 (30%) (G1, G2, G3, F2, M1, S4, S5, S6, G5)
q_hole	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)	1 (3.3 %) (F4)
q_thinwall_thick_less_0.5	5 (16.7 %) (W5, W7, W8, W9, W10)	5 (16.7 %) (W5, W7, W8, W9, W10)	5 (16.7 %) (W5, W7, W8, W9, W10))	4 (13.3 %) (W5, W7, W9, W10)

In this experiment, the number of exemplars retrieved for the query “q_hole” does not change with the value of the variable “difference” except when it is zero. For the queries “2_planes_with_distance” and “gear_pinion_02”, the exemplar author may find it tedious to evaluate which exemplars are most suitable to his needs the number of

exemplars retrieved when value of “difference” is -3 or -5, since nearly half the database is retrieved. As well, when the value of “difference” is zero, the exemplar author is provided with an appropriate number of exemplars from the database to consider. However, for the queries “2_planes_with_distance.stp”, “q_hole.stp”, and “q_thinwall_thick_less_0.5”, the number of exemplars retrieved is appropriate when the value of “difference” is -1.

On comparing the results for all queries for the different values of the variable “difference”, it can be inferred that a value of 0 makes the relation filter highly restrictive, whereas for values of -3 and -5, the relation filter is not restrictive enough. As well, comparing the results shown in Table 8.5 with the results shown in

Table 8.7, it is observed that the relation filter influences the results of retrieval more than the entity filter. This aspect is further explored by conducting experiments with both the entity and relation filters in place.

8.4.3. Entity and Relation Filters

The results of the retrieval process with both the entity and relation filters are summarized in Table 8.8. A value of zero for the variable “difference” implies that those target exemplars not having at least the same number of entities and relations of each type as the query exemplar do not pass the filter. In this experiment also, the number of exemplars retrieved for the query “q_hole.stp” does not change with the value of the variable “difference” except when it is zero. As well, the number of exemplars retrieved

for the query “2_planes_with_distance.stp” does not change when the value of “difference” is -5 or -3. . As well, when the value of “difference” is zero, the exemplar author is provided with an appropriate number of exemplars from the database to consider. However, for the queries “2_planes_with_distance.stp”, “q_hole.stp”, and “q_thinwall_thick_less_0.5”, the number of exemplars retrieved is appropriate when the value of “difference” is -1.

Table 8.8: Exemplars retrieved with entity and relation filters

Query	**Difference = -5**	**Difference = -3**	**Difference = -1**	**Difference = 0**
2_planes_with_distance	19 (W1, W2, W3, F1, F2, G4, F3, W4, F4, F5, F6, F7, W5, W6, W7, W8, W9, W10, G5)	19 (W1, W2, W3, F1, F2, G4, F3, W4, F4, F5, F6, F7, W5, W6, W7, W8, W9, W10, G5)	5 (16.7 %) (W1, W2, G4, W8, W10)	3 (10 %) (W1, W2, W10)
gear_pinion_02	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	14 (46.7 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	8 (G1, G2, G3, F2, S4, S5, S6, G5)
q_hole	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)	1 (F7)
q_thinwall_thick_less_0.5	5 (16.7 %) (W5, W7, W8, W9, W10)	5 (16.7 %) (W5, W7, W8, W9, W10)	5 (16.7 %) (W5, W7, W8, W9, W10)	4 (13.3 %) (W5, W7, W9, W10)

On comparing the results for all queries for the different values of the variable "difference", it can be inferred that a value of 0 makes the entity and relation filters highly restrictive, whereas for values of -3 and -5, the entity and relation filters are not restrictive enough. Comparing the results shown in Table 8.5, and Table 8.7 with the

results shown in Table 8.8, it can be seen that the relation filter influences the results of retrieval more than the entity filter.

These results seem to suggest that the relation filter should not be highly restrictive since a highly restrictive relation filter will filter out a lot of exemplars from the database. As well, the entity filter should be highly restrictive since, a less restrictive entity filter retrieves nearly 50 % of the exemplars in the database. This implies that the relation filter should be less restrictive as compared to the entity filter. Based on the number of exemplars retrieved, it can be inferred that the value of "difference" for the relation filter should be -1, whereas for the entity filter it should be 0. However, the results of retrieval may change for different levels of restrictiveness of the attribute filter. The experiments conducted for different levels of restrictiveness for the attribute filter are described in the next section.

8.4.4. Attribute Filter

The results of the retrieval process with the attribute filter alone are summarized in Table 8.9. A value of zero for the variable "difference" implies that those target exemplars not having at least the same number of attributes of each type as the query exemplar do not pass the filter. In this experiment, the number of exemplars retrieved for the query "q_hole" does not change with the value of the variable "difference". As well, the number of exemplars retrieved for the query "2_planes_with_distance" changes only when the value of "difference" is -5. For the query, "gear_pinion_02.stp", 47 % of the

exemplars in the database get retrieved when the value of “difference” is -5.

Table 8.9: Number of exemplars retrieved with attribute filter

Query	**Difference = -5**	**Difference = -3**	**Difference = -1**	**Difference = 0**
2_planes_with_distance	10 (33.34 %) (W1, W2, F1, F2, F3, F4, F5, F6, F7, W7)	10 (33.34 %) (W1, W2, F1, F2, F3, F4, F5, F6, F7, W7)	10 (33.34 %) (W1, W2, F1, F2, F3, F4, F5, F6, F7, W7)	10 (33.34 %) (W1, W2, F1, F2, F3, F4, F5, F6, F7, W7)
gear_pinion_02	14 (46.67 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	14 (46.67 %) (G1, G2, F1, G3, F2, F3, M1, F5, F6, S4, S5, F7, S6, G5)	13 (43.34 %) (G1, G2, F1, G3, F2, F3, F5, F6, S4, S5, F7, S6, G5)	13 (43.34 %) (G1, G2, F1, G3, F2, F3, F5, F6, S4, S5, F7, S6, G5)
q_hole	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)
q_thinwall_thick_less_0.5	5 (16.7 %) (W5, W7, W8, W9, W10)	4 (13.34 %) (W5, W7, W9, W10)	3 (10 %) (W5, W7, W9)	3 (10 %) (W5, W7, W9)

From the results shown in Table 8.9, it can be seen that the number of exemplars retrieved does not change significantly across all queries for different values of the variable “difference”. The query “gear_pinion_02.stp” retrieves 47% of the database when the value of “difference” is -5. As well, the fact that the query “q_thinwall_thick_less_0.5.stp” retrieves four exemplars when the value of “difference”

is -3, suggests that for the attribute filter, the value of the variable "difference" should be -3 or -5 for this database.

Based on the results obtained from conducting different experiments on the entity, relation, and attribute filters, it can be inferred that the restrictiveness of the relation filter has the most impact on the outcome of the retrieval process. This is demonstrated from the fact that, when the value of the variable "difference" is zero, the number of exemplars retrieved by the relation filter alone, is the less than the number of exemplars retrieved by entity or attribute filters. As well, the entity filter may have the least impact on the outcome of the retrieval process. This can be inferred from the fact that, when the value of the variable "difference" is zero, the number of exemplars retrieved by the entity filter alone, is the less than the number of exemplars retrieved by the relation or attribute filters. One possible explanation for this result may be that entities are more common in exemplars as compared to relations and attributes. The different types of entities found in exemplars are more than the different types of relations. As well, there is no significant difference in the number of exemplars retrieved as the attribute filter becomes less restrictive. This is demonstrated from the results obtained for different values of the variable "difference" when the attribute filter alone is in place.

These observations suggest that the value of the variable "difference" should be different for each filter. Specifically, the entity filter should be most restrictive and hence the value of the variable "difference" should be 0 for this filter. Since, a highly restrictive relation filter does not provide too many options for the exemplar author to adapt, the

value of "difference" for this filter should be either -1 or -3. Similarly, the results of retrieval with the attribute filter alone, suggest that the value of "difference" for this filter should be -3. Table 8.10 shows the results obtained for these values of entity, relation, and attribute filters.

Table 8.10: Number of exemplars retrieved for different combinations of filters.

Query	**difference(entity) = 0; difference(relation) = -1; difference (attribute) = -3;**	**difference(entity) = 0; difference(relation) = -3; difference (attribute) = -3;**
2_planes_with_distance	3 (10 %) (W1, W2, W7)	12 (40 %) (W1, W2, W3, F1, F3, F5, F6, F7, W5, W7, W9, W10)
gear_pinion_02	12 (40 %) (G1, G2, F1, G3, F2, F3, F5, F6, S4, S5, S6, G5)	12 (40 %) (G1, G2, F1, G3, F2, F3, F5, F6, S4, S5, S6, G5)
q_hole	5 (16.7 %) (F1, F2, F3, F4, F7)	5 (16.7 %) (F1, F2, F3, F4, F7)
q_thinwall_thick_less_0.5	3 (10 %) (W5, W7, W9)	3 (10 %) (W5, W7, W9)

From the results obtained, it can be seen that, as the value of the variable "difference" changes from -1 to -3 for the relation filter, nine more exemplars are retrieved for the query "2_planes_with_distance.stp". The number of exemplars obtained for other queries remain the same across different values of the variable "difference".

8.5. Summary

The primary objective of conducting these experiments was to develop a better understanding of similarity with respect to graph-based models and to provide an insight into the effect that each structural filter has on the outcome of the retrieval process. These experiments demonstrate that the structural similarity filters are accurate and perform as expected. It should be noted that the values for the variable "difference" are suggested specifically for the database used in these experiments. However, these results imply a trend on the results of retrieval that may be observed for different databases. The experiments and the results demonstrate that the structural filters can be modified according to the exemplar authors' needs and according to the size of the database. The results obtained suggest that the relation filter may be the more influential than the entity and the attribute filters. This implies that different levels of restrictiveness for each filter may yield appropriate results. Specifically, keeping the entity filter highly restrictive may provide an appropriate number of exemplars to the exemplar author.

Chapter 9

EXPERIMENTAL VERIFICATION OF SEMANTIC SIMILARITY MEASURES

Having implemented the semantic similarity measures as described in Chapter 7, different experiments were conducted in order to evaluate the effectiveness of these measures. As discussed in Chapter 8, if the exemplar author is provided with a large number of alternative configurations, it may be a relatively tedious task for the exemplar author to evaluate which of the retrieved exemplars satisfies most of his requirements. Similarly, if the similarity measures retrieve only a few alternative configurations, then there is a possibility that the exemplar author may not be able to adapt any of the retrieved exemplars to satisfy the new requirements. Hence, the semantic similarity measures may be considered effective if the exemplar author is provided with an appropriate number of alternative exemplar configurations. Studies in human factors suggest that a user should be provided 7 ± 2 alternative configurations [108]. These experiments were aimed at understanding the effect of different weighting schemes of the terms in the controlled vocabulary on the results of the exemplar retrieval process.

As mentioned in Chapter 7, for purposes of conducting these experiments, the database consisted of thirty exemplars. Semantic descriptions of four exemplars from the database were chosen as query exemplars for all the experiments. These four exemplars are the same four exemplars that were used as query exemplars for experimental

verification of structural similarity filters. This ensures that the results obtained from the structural and semantic similarity measures can be compared and evaluated.

As discussed in Chapter 5, the intent of the exemplar and the rationale for authoring it in a specific manner should be represented in order to retrieve exemplars semantically similar to the target exemplar. The semantic descriptions of the exemplar for finding the distance between two planes is shown in Figure 9.1. This file is named "2_planes_with_distance.des".

*"The intent of this exemplar is to find the **distance** between two **planes.** The exemplar consists of two **planes** with a **parallel** and a **distance relation** between the **planes** that have the **attribute extract** since they are not **explicitly present** in the model. The **parallel relation** ensures that the two **planes** are **parallel** to each other and the **distance relation extracts** the **distance** between the **two planes.** When the exemplar is applied to the model, the **distance** between the **planes** is displayed as well as the **planes** are **highlighted.**"*

Figure 9.1: Semantic description of exemplar "2_planes_with_distance"

As seen from Figure 9.1, the semantic description of the exemplar "2_planes_with_distance" contains a total of eighty nine words. Of these eighty nine words, twenty five words are taken from the controlled vocabulary. These words are highlighted in bold. It should be noted that certain words occur more than once. This is

important, since the number of occurrences of a specific word determines its importance in the description. The first sentence describes the intent of the exemplar. The remaining sentences are used to describe the rationale for using the specific entities and relations having their specific attributes.

The semantic description of the exemplar for finding a gear and a pinion in a model is shown in Figure 9.2. The file is named as "gear_pinion_02.des".

*This exemplar is intended to **find** a pair of **gears** in the model and determine which the pinion is and which the **gear** is. The exemplar consists of 2 **circles** that have the **attribute match.** One **circle** represents the **gear** and the other **circle** represents the pinion. The **equation relation** is applied to the **radii parameters** to **check** which the smaller **circle** is. The smaller **circle** is the pinion and the bigger **circle** is the **gear.***

Figure 9.2: Semantic description of exemplar "gear_pinion_02"

As seen from Figure 9.2, the semantic description of the exemplar "gear_pinion_02" contains seventy six words. Of these seventy six words, eighteen words are taken from the controlled vocabulary. These words are highlighted in bold. In this case as well, certain words occur more than once. The number of occurrences of a word determines its importance in the description.

The third query used in testing the semantic similarity measures is the exemplar for finding holes in a model. The semantic description of the query is named

"q_hole.des". The semantic description of the exemplar is shown in Figure 9.3.

*The intent of this exemplar is to **find holes** in a model. The exemplar consists of a **cylindrical surface** that is bound by two **circles** and two **planes**, each bound by one of the **circles**. These **entities form** the **structure** of the **hole**. An **id relation** is applied to the **cylindrical surface**. When this exemplar is applied to any model all **cylindrical holes** in the model will get **highlighted**.*

Figure 9.3: Semantic description of exemplar "q_hole"

As seen from Figure 9.3, the total number of words in this description is sixty nine, of which eighteen are taken from the controlled vocabulary. Similar to the descriptions of the queries "2_planes_with_distance" and "gear_pinion_02", certain words occur more than once.

The fourth query used for the experimental verification of the semantic similarity measures is the exemplar for finding thin walls in a model. The exemplar is named as "q_thinwall_thick_less_0.5.des". The semantic description of this exemplar is as shown in Figure 9.4. In this semantic description, the total number of words is sixty. Of these sixty words, sixteen words are taken from the controlled vocabulary.

*The intent of this exemplar is to **find thin walls.** The exemplar consists of a solid **manifold** bound by three **planes.** These **planes** form the **structure** of a **wall.** A **distance relation** is applied between two **planes** in order to **determine** the **thickness** of the **wall.** An **equation** is used to evaluate whether the **thickness** is less than 0.5 units.*

Figure 9.4: Semantic description of exemplar "q_thinwall_thick_less_0.5"

As described in Chapter 7, the first step in computing the semantic similarity between the query and a target exemplar in the database is to represent the semantic descriptions of both exemplars in the form of vectors. The size of the vector is equivalent to the size of the controlled vocabulary. The words in the description that are not part of the controlled vocabulary are not part of the vector. The importance of each term in a vector is dependent on the number of occurrences of the word in the document and the weight assigned to it in the controlled vocabulary. If a term in the vocabulary is assigned a weight of "p", and if it occurs "n" times in a description, its importance in the description is "{(p * n)/L}", where L is the number of words in the description. For example, consider a term in the controlled vocabulary that has assigned a weight of seven, and that it occurs four times in a text description having fifteen words. In this case, the value of "p" is 7, the value of "n" is 4 and the value of L is 15. The importance of this term in the semantic description is evaluated using the expression (7 * 4) / 15 which is calculated to be 1.87. Similarly, if the same term were assigned a weight of 3, the

importance of the term in the same semantic description is evaluated using the expression (3 * 4) / 15 which is calculated to be 0.8. The cosine angle between the two vectors provides a measure of the similarity between the two exemplars. Since the importance of a term in a semantic description changes with a change in the value of assigned weight to the term, the value of the cosine angle between two vectors will also change. Thus, the most influential factor in computing the similarity between two semantic descriptions is the weight assigned to a term in the controlled vocabulary. The edit distance on the other hand remains unchanged. Hence the experiments conducted to test the effectiveness of the semantic similarity measures involve the use of different weighing schemes for the terms in the controlled vocabulary. In the first experiment, words that occur frequently in text descriptions were assigned less weights compared to words that are not frequently used. The rationale for doing so is that words that are less common across semantic descriptions may be more important in distinguishing one semantic description from the other. In the second experiment, words that occur more frequently across semantic descriptions of exemplars were assigned more weights. The reason for doing so is that the semantic similarity measures will retrieve more exemplars using this weighing scheme and thus provide the exemplar author with more options. In the third experiment, all words in the vocabulary were assigned the same weights. The results obtained from these experiments are discussed in the following sections.

9.1. Experiment 1: Higher weights to less frequent words

In the first experiment, words that occur frequently in text descriptions were assigned less weights compared to words that are not frequently used. The rationale for doing so is that words that are less common across semantic descriptions may be more important in distinguishing one semantic description from the other. For example, consider the semantic descriptions of the queries "2_planes_with_distance", "gear_pinion_02", "q_hole", and "q_thinwall_thick_less_0.5" shown in Figure 9.1, Figure 9.2, Figure 9.3, and Figure 9.4 respectively. The word "gear" distinguishes the query "gear_pinion_02" from the other three queries. Similarly the word, "hole" distinguishes the query "q_hole" from the other exemplars. As well, the word "wall' distinguishes the exemplar "q_thinwall_thick_less_0.5" from the other exemplars. The word "planes" occurs in the semantic descriptions of "2_planes_with_distance", "q_thinwall_thick_less_0.5", and "q_hole". Thus, the word "planes" does not assist in distinguishing the three exemplars. Therefore, if more weights are assigned to less frequent words, the semantic similarity measures would be more effective.

Therefore, terms pertaining to design exemplar technology such as the different types of entities, relations and attributes were assigned lower weights, whereas terms describing features such as hole, boss, and slot were assigned a higher weight. As well, all terms in the vocabulary that belong to a domain are assigned the same weights. For example, all features such as boss, hole, and slot were assigned a weight of seven, whereas all different types of entities and relations were assigned a weight of

two. As well, singular and plural forms of the same word were assigned the same weight. A sample of the controlled vocabulary with this weighing scheme is listed in Table 9.1.

Table 9.1: Sample of vocabulary with higher weights for less frequent words

Word	**Weight**	**Word**	**Weight**
Distance	2	Hole	7
Parallel	2	Boss	7
Radius	2	Slot	7
Extracts	1.5	Pocket	7
Line	2	Gear	7
circle	2	Flute	7
Plane	2	Jig	7

As can be seen, terms pertaining to exemplar technology were assigned lower weights, whereas words describing features were assigned a higher weight. It is important to note that the word “extracts” is assigned less weight than the other words pertaining to the design exemplar. This is so, because the word “extracts” is more common among the semantic descriptions of exemplars than words describing entities and relations such as “line”, “circle”, and “radius”.

In order to retrieve those exemplars from the database that are semantically similar to the query exemplar the cosine angle between the query vector and each target vector was computed as per the algorithms described in Chapter 7. The cosine angle

between two vectors can vary between 0 and 1. A cosine angle of 1 implies an exact match, where as a cosine angle of 0 implies a completely dissimilar exemplar. As well, for purposes of restricting the number of retrieved exemplars, the minimum desired cosine value of the angle between a target exemplar and a query exemplar was 0.3. All target exemplars that had a cosine angle value of less than 0.3 with the query exemplar were not retrieved. The exemplars retrieved by the vector similarity filter were subjected to the edit distance similarity measure. The edit distance similarity measure takes into account all the words in the semantic descriptions that are not in the controlled vocabulary while computing the number of changes required converting the query exemplar to the target exemplar. The results of using this weighing scheme for each of the four queries are listed in Table 9.2.

Table 9.2: Results obtained with higher weights for features

Query	Exemplar	Cosine Angle	Edit Distance
2_planes_with_distance	2_lines_dist_2_planes	0.67	13
	2_planes_with_distance	1	0
	2_points_2_lines_with_dist	0.5	26
	2_ponts_with_distance	0.52	27
	planes_lines_points_distance	0.66	20
	q_dist_bounded_planes	0.94	21
	q_thinwall_parallel_planes	0.42	26
	q_thinwall_solid_bounded_planes	0.32	31
	q_thinwall_thick_less_0.5	0.3	32
gear_pinion_02	gears_double_ratio_03	0.76	23
	gear_pinion_02	1	0
	q_compound_5_gears	0.53	27
	q_radii_ratio	0.33	23
	q_radii_ratio_w_large_check	0.36	24
	spur_gears_02	0.52	23
q_hole	q_cylinder	0.387	20
	q_hole	1	0
	q_hole_depth_radius	0.711	9
	q_hole_radius_depth	0.735	13
	q_radius_cylindrical_hole	0.5043	28

q_thinwall_thick_less_0.5	2_planes_with_distance	0.3	26
	q_coplanar gears	0.399	22
	q_thinwall_medium_with_boundary	0.95	1
	q_thinwall_parallel_planes	0.76	27
	q_thinwall_simple_boundary	0.93	4
	q_thinwall_solid_bounded_planes	0.92	17
	q_thinwall_thick_less_0.5	1	0
	q_thinwall_twoloops_simple	0.93	4

From the results shown in Table 9.2, it can be seen that each query is most similar to itself. This is inferred from the fact the cosine value of the angle formed by each query with itself is 1. As well, the edit distance between each query exemplar and itself is zero. The number of exemplars retrieved for the query "2_planes_with_distance" is nine whereas the number of exemplars retrieved for the query "gear_pinion_02" is six. The semantic similarity measures retrieve five exemplars for the query "q_hole" and eight exemplars for the query "q_thinwall_thick_less_0.5".

For the query, "2_planes_with_distance", the query "q_dist_bounded_planes" is the most similar. On observing the semantic descriptions of the two exemplars, it is seen that the intent of both these exemplars is the same. The queries, "q_thinwall_parallel_planes", "q_thinwall_solid_bounded_planes", and "q_thinwall_thick_less_0.5" have an intent that is different from the intent of the query

exemplar. Therefore, the cosine value of the angle between these exemplars and the query exemplar is less than the cosine value of the angle formed by the other target exemplars. The query "planes_lines_points_distance" has the same intent as the query exemplar, but the rationale for authoring the exemplar is different from the query exemplar. Therefore, the cosine value of the angle between this exemplar and the target exemplar is 0.62.

Similarly, for the queries "gear_pinion_02", "q_hole", and "q_thinwall_thick_less_0.5", vectors formed by exemplars having similar intents have a higher cosine value. As well, the angle itself can be used to rank the retrieved exemplars. The edit distance measure does not contribute towards evaluating the similarity between the query exemplar and target exemplar, except when the query vector forms an angle with the same cosine value with two or more target vectors.

These results suggest that assigning higher weights to less frequent words retrieves exemplars that more similar to the query exemplar with respect to intent. This is expected, since less frequent words, such as words describing features, are used in describing the intent and more frequent words, such as words related to the design exemplar, are used to describe the rationale. It should be noted, that for this experiment, semantic descriptions for all exemplars in the database were written by the same author. Hence, there is a possibility that the style of writing semantic descriptions of exemplars may have an impact on the outcome of the semantic retrieval process.

9.2. Experiment 2: Higher weights for more frequent words

In the second experiment, words that occur frequently in text descriptions were assigned higher weights compared to words that are not frequently used. Thus for example, terms pertaining to design exemplar technology such as the different types of entities, relations and attributes were assigned higher weights, whereas terms describing features such as hole, boss, and slot were assigned a lower weight. As described in the previous section, less frequent words help in differentiating one semantic description from the other. When more frequent words are assigned higher weights, the vector representation of each semantic description will change Hence, there is a possibility that the dot product of a target vector and the query vector may increase which in turn increases the cosine value of the angle between them. This implies that more exemplars will get retrieved. This will provide more options to the exemplar author which may be helpful in expanding the exemplar author's search space.

Similar to the previous experiment, all terms in the vocabulary belonging to a specific domain are assigned the same weights and singular and plural forms of the same word are assigned the same weight. A sample of the controlled vocabulary with this weighing scheme is listed in Table 9.3.

Table 9.3: Sample of vocabulary with lower weights for features

Word	**Weight**	**Word**	**Weight**
Distance	7	Hole	1
Parallel	7	Boss	1
Radius	7	Slot	1
Extracts	6.5	Pocket	1
Line	7	Gear	1
circle	7	Flute	1
Plane	7	Jig	1

As seen from Table 9.3, words describing features were assigned the same weight. As well, it can be seen that the word “extracts” has a weight of 6.5, whereas the entities and relations have a weight of 7. In the previous experiment as well, the word “extracts” was assigned a smaller weight compared to the other words pertaining to the design exemplar. Hence, this distinction of weights is consistent with the weights in the previous experiment. Similar to the previous experiment, all target exemplars that had a cosine angle value of less than 0.3 with the query exemplar were not retrieved. The exemplars retrieved by the vector similarity filter were subjected to the edit distance similarity measure. The results of using this weighing scheme for each of the four queries are listed in Table 9.4.

Table 9.4: Exemplars retrieved by using higher weights for more frequent words

Query	Exemplar	Cosine Angle	Edit Distance
gear_pinion_02	gears_double_ratio_03	0.44	23
	gear_pinion_02	1	0
	q_boss_radius	0.34	27
	q_compound_5_gears	0.3	27
	q_double_radius	0.39	28
	q_hole_radius_depth	0.35	28
	q_radii_ratio	0.64	23
	q_radii_ratio_w_large_check	0.69	24
	spur_gears_02	0.3	23
2_planes_with_distance	2_lines_dist_2_planes	0.67	13
	2_planes_with_distance	1	0
	2_points_2_lines_with_dist	0.54	26
	2_ponts_with_distance	0.53	27
	gears_double_ratio	0.37	37
	planes_lines_points_distance	0.71	20
	q_boss_radius	0.41	27
	Q_cylinder	0.35	29
	Q_hole_depth_radius	0.35	23
	Q_hole_radius_depth	0.46	27
	q_dist_bounded_planes	0.92	21

	q_thinwall_medium_with_boundary	0.53	25
	q_thinwall_parallel_planes	0.67	26
	q_thinwall_simple_boundary	0.64	22
	q_thinwall_solid_bounded_planes	0.79	31
	q_thinwall_thick_less_0.5	0.691	32
	q_thinwall_twoloops_simple	0.649	22
q_hole	q_cylinder	0.387	20
	q_boss_radius	0.51	16
	q_connecting_rod_thick_enough	0.3	25
	q_coplanar_gears	0.3	36
	q_hole	1	0
	q_hole_depth_radius	0.56	9
	q_hole_radius_depth	0.62	13
	q_radius_cylindrical_hole	0.32	28
	q_tangent_circles	0.43	34
	q_thinwall_medium_with_boundary	0.32	25
	q_thinwall_parallel_planes	0.34	27
	q_thinwall_simple_boundary	0.43	25
	q_thinwall_solid_bounded_planes	0.34	28
	q_thinwall_thick_less_0.5	0.31	27
	q_thinwall_twoloops_simple	0.43	25

	spur_gears_02	0.47	26
q_thinwall_thick_less_0.5	2_planes_with_distance	0.69	26
	2_lines_dist_2_planes	0.4	24
	planes_lines_points_distance	0.49	25
	q_boss_radius	0.47	17
	q_coplanar gears	0.399	22
	q_connecting_rod_thick_enough	0.38	22
	q_cylinder	0.36	22
	q_dist_bounded_planes	0.64	18
	q_hole	0.31	25
	q_hole_depth_radius	0.34	21
	q_hole_radius_depth	0.47	20
	q_thinwall_medium_with_boundary	0.85	1
	q_thinwall_parallel_planes	0.65	27
	q_thinwall_simple_boundary	0.77	4
	q_thinwall_solid_bounded_planes	0.58	17
	q_thinwall_thick_less_0.5	1	0
	q_thinwall_twoloops_simple	0.77	4

As seen from Table 9.4, the edit distance measure remains the same in this case since the text descriptions themselves have not changed. As well, the cosine values of the

angles suggest that each query is most similar to itself. However, on comparing these results with the results obtained in the previous experiment (Table 9.4), it is seen that exemplars with the same intent have a smaller cosine value than the exemplars that have different intents but similar entities and relations. For example, the retrieval results show that the exemplars "q_radii_ratio" and "q_radii_ratio_w_large_check" are more semantically similar to the query "gear_pinion_02", compared to the exemplar "gear_double_ratio_03". One reason for this result may be the fact that the word "circle" has more weight than the word "gear". Similarly, for the query "q_thinwall_thick_less_0.5", the results suggest that the exemplar "2_planes_with_distance" is more semantically similar to the query than the exemplar "q_thinwall_parallel_planes". Hence, it may be inferred that this weighing scheme may assign a higher similarity score to exemplars that are more structurally similar to the query exemplar.

As well, the number of exemplars retrieved for the query "2_planes_with_distance" is seventeen whereas the number of exemplars retrieved for the query "gear_pinion_02" is nine. The semantic similarity measures retrieve sixteen exemplars for the query "q_hole" and seventeen exemplars for the query "q_thinwall_thick_less_0.5". Hence, the number of exemplars retrieved in this experiment is more than the number of exemplars retrieved in the previous experiment. The reason for these results may be the fact that structurally similar exemplars have similar entities and relations, and have similar vectors because of the weighing scheme

used in this case. As well, the number of exemplars retrieved in this experiment is more than the number of exemplars retrieved in the previous experiment.

The next experiment studies the effect of using equal weights for all terms in the controlled vocabulary, on the results of retrieval.

9.3. Experiment 3: Equal weights for all words

In this experiment, equal weights are assigned to all words in the controlled vocabulary. There is no distinction made between the words that occur frequently across semantic descriptions and the words that are not so frequent. A sample vocabulary is shown in Table 9.5.

Table 9.5: Sample of vocabulary with equal weights for all terms

Word	**Weight**	**Word**	**Weight**
Distance	4	Hole	4
Parallel	4	Boss	4
Radius	4	Slot	4
Extracts	4	Pocket	4
Line	4	Gear	4
circle	4	Flute	4
Plane	4	Jig	4

Similar to the previous experiment, all target exemplars that had a cosine angle

value of less than 0.3 with the query exemplar were not retrieved. The exemplars retrieved by the vector similarity filter were subjected to the edit distance similarity measure. The results of using this weighing scheme for each of the four queries are listed in Table 9.6.

Table 9.6: Exemplars retrieved with equal weights assigned to all terms

Query	Exemplar	Cosine Angle	Edit Distance
gear_pinion_02	gears_double_ratio_03	0.546	23
	gear_pinion_02	1	0
	q_compound_5_gears	0.38	27
	q_double_radius	0.36	28
	q_radii_ratio	0.57	23
	q_radii_ratio_w_large_check	0.61	24
	spur_gears_02	0.39	23
2_planes_with_distance	2_lines_dist_2_planes	0.676	13
	2_planes_with_distance	1	0
	2_points_2_lines_with_dist	0.55	26
	2_ponts_with_distance	0.54	27
	planes_lines_points_distance	0.71	20
	q_boss_radius	0.38	27
	Q_cylinder	0.36	29
	Q_hole_depth_radius	0.33	23

	Q_hole_radius_depth	0.40	27
	q_dist_bounded_planes	0.92	21
	q_thinwall_medium_with_boundary	0.48	25
	q_thinwall_parallel_planes	0.64	26
	q_thinwall_simple_boundary	0.56	22
	q_thinwall_solid_bounded_planes	0.62	31
	q_thinwall_thick_less_0.5	0.60	32
	q_thinwall_twoloops_simple	0.56	22
q_hole	q_cylinder	0.41	20
	q_boss_radius	0.42	16
	q_hole	1	0
	q_hole_depth_radius	0.62	9
	q_hole_radius_depth	0.67	13
	q_radius_cylindrical_hole	0.38	28
	q_tangent_circles	0.36	34
	q_thinwall_simple_boundary	0.36	25
	q_thinwall_solid_bounded_planes	0.33	28
	q_thinwall_thick_less_0.5	0.32	27
	q_thinwall_twoloops_simple	0.36	25
	spur_gears_02	0.32	26
q_thinwall_thick_less_0.5	2_planes_with_distance	0.35	26
	2_lines_dist_2_planes	0.6	24

	planes_lines_points_distance	0.43	25
	q_boss_radius	0.43	17
	q_connecting_rod_thick_enough	0.34	22
	q_cylinder	0.39	22
	q_dist_bounded_planes	0.571	18
	q_hole	0.321	25
	q_hole_depth_radius	0.33	21
	q_hole_radius_depth	0.43	20
	q_thinwall_medium_with_boundary	0.87	1
	q_thinwall_parallel_planes	0.66	27
	q_thinwall_simple_boundary	0.82	4
	q_thinwall_solid_bounded_planes	0.65	17
	q_thinwall_thick_less_0.5	1	0
	q_thinwall_twoloops_simple	0.824	4

As can be seen from the results, the cosine values of some of the retrieved exemplars are nearly the same. For example, from Table 9.6, it is observed that the queries “planes_lines_points_distance”, “q_boss_radius”, and “q_hole_radius_depth” are considered equally similar to the query exemplar “q_thinwall_thick_less_0.5” since the vector representations of these exemplars form the same angle with the query vector. Similarly, the exemplars “q_tangent_circles”, “q_thinwall_simple_boundary”, and

"q_thinwall_twoloops_simple" are considered equally similar semantically to the query "gear_pinion_02". These results imply that assigning equally weights to all the words in the controlled vocabulary does not help in differentiating the semantic descriptions from each other.

As well, the semantic similarity measures retrieve fifteen exemplars that are semantically similar to the query "q_thinwall_thick_less_0.5". The number of exemplars retrieved for the query "2_planes_with_distance" is sixteen whereas the number of exemplars retrieved for the query "gear_pinion_02" is seven. The semantic similarity measures retrieve twelve exemplars for the query "q_hole".

9.4. Summary

From all three experiments, it can be inferred that assigning higher weights to words that are less frequent across semantic descriptions, retrieves exemplars that are more semantically similar to the query exemplars. As well, the number of exemplars retrieved using this scheme is appropriate. One possible explanation for this may be that less frequent words such as "boss", "hole", "wall", and "gear" help differentiate one semantic description from the other. Therefore, assigning higher weights to such words helps retrieve exemplars that are similar with respect to intent.

Using a weighting scheme that assigns higher weights to more frequent words retrieves more exemplars, but the retrieved exemplars are more structurally similar to the query exemplar. Words such as "radius", "plane", "circle", and "distance" are more

frequent across all semantic descriptions. Therefore, assigning higher weights to such words retrieves exemplars that are structurally similar to the query exemplar.

Similarly, assigning equal weights to all words in the controlled vocabulary does not help in differentiating between semantic descriptions. Hence, it can be inferred that assigning higher weights to less frequent words make the semantic similarity measures relatively more effective.

Chapter 10

EXPERIMENTAL VALIDATION OF AGGREGATING SIMILARITY MEASURES.

The aggregation module combines the two similarity modules in order to retrieve exemplars that are both semantically and structurally similar to a query exemplar. As mentioned in Chapter 7, two ways of combining the similarity modules may be to either use them in series or use them in parallel. Three different experiments were conducted in order to study and evaluate the results obtained from different ways of combining the two similarity modules. In the first experiment, the two similarity modules were used in parallel on the same query. In the second experiment, the structural retrieval module is used to retrieve exemplars that are structurally similar to the query exemplar. Following this, the semantic similarity module is used to retrieve exemplars that are semantically similar to each of the structurally similar exemplars retrieved by the structural similarity module. In the third experiment, the semantic similarity module is used to retrieve exemplars that are semantically similar to the query exemplar following which the structural similarity module is used to retrieve exemplars that are structurally similar to each of the retrieved exemplars. This chapter discusses the results obtained from these experiments. The main objective of these experiments was to explore different ways that two views of similarity can be used together to evaluate the similarity between exemplars. As well, a secondary objective was to determine how the two similarity

modules can be combined in order to retrieve exemplars that are useful to the exemplar author.

For the different experiments described in this chapter, there are two variables that can be varied. The first variable is the cosine value of the angle formed between the query vector and the vector of a target exemplar in the database. For purposes of restricting the number of exemplars getting retrieved, it was decided that the lower limit on the value of the cosine angle should be either 0.3 or 0.5. The limit of 0.3 implies that target exemplars that had a cosine angle value of less than 0.3 with the query exemplar were not retrieved. The value of 0.5 makes the semantic similarity measures more restrictive. In this case, higher weights were assigned to words that occur less frequently across semantic descriptions.

The second variable is a measure of the structural similarity between a target exemplar and a query exemplar. This measure is calculated as a ratio of the number of entities and relations present in the query at the time of a successful match to the total number of entities and relations originally present in the query. A value of 1 implies an exact match whereas a value of 0 implies that the target exemplar is not similar to the query exemplar at all. If two target exemplars having the same similarity measure are retrieved, it implies that both exemplars are equally similar to the query exemplar. For purposes of this experiment, the minimum threshold value for this measure of structural similarity is set to either 0.5 or 0.75.

Hence, four experiments were conducted for each query exemplar since there are four possible combinations of these variables. These four experiments are tabulated in Table 10.1.

Table 10.1: Possible combination of experiment variables

	Cosine angle = 0.3	**Cosine Angle = 0.5**
Structural Similarity = 0.50	(0.3, 0.5)	(0.5, 0.75)
Structural Similarity = 0.75	(0.3, 0.75)	(0.5, 0.75)

10.1. Using the similarity modules in parallel

In the first set of experiments, the two modules were used in parallel to retrieve exemplars that are structurally and semantically similar exemplars. Using the two modules in parallel implies that the structural similarity module is used to retrieve exemplars that are structurally similar to a query exemplar while the semantic similarity module retrieves exemplars that are semantically similar to the same query exemplar. The value of the variable "difference", for the entity filter in this experiment was 0, for the relation filter, the value of "difference" was -3, and the value of "difference" was -3. For computing the semantic similarity between exemplars, higher weights were assigned to words that were less frequent across semantic descriptions. The user is provided with a list of structurally similar exemplars and a list of semantically similar exemplars. The objective of conducting these experiments is to observe whether the same

exemplars or different exemplars are retrieved using both modules. In this experiment, the four exemplars that were used to individually test the two modules were used as queries; "2_planes_with_distance", "gear_pinion_02", "q_hole", and "q_thinwall_thick_less_0.5".

The results of using the exemplar "2_planes_with_distance" are summarized in Table 10.2. In this case, the minimum value of the cosine angle between a target exemplar and the query is 0.3 and the minimum threshold of structural similarity measure is 0.5.

Table 10.2: Results of using structural and semantic similarity modules in parallel

2_planes_with_distance (cosine angle ≥ 0.3, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
2_lines_dist_2_planes (0.678)	2_lines_dist_2_planes (0.75)
2_planes_with_distance (1)	2_planes_with_distance (1)
2_points_2_lines_with_dist (0.5)	q_boss_radius (1)
2_ponts_with_distance (0.52)	q_cylinder (1)
planes_lines_points_distance (0.66)	q_hole_depth_radius (0.75)
q_dist_bounded_planes (0.94)	q_hole_radius_depth (1)
q_thinwall_parallel_planes (0.42)	q_radius_cylindrical_hole (0.75)
q_thinwall_solid_bounded_planes (0.34)	q_thinwall_simple_boundary (1)
q_thinwall_thick_less_0.5 (0.3)	q_thinwall_thick_less_0.5 (1)
	q_thinwall_medium_with_boundary (1)

As can be seen from the results from Table 10.2, only exemplars "2_lines_dist_2_planes", "2_planes_with_distance", and "q_thinwall_thick_less_0.5" are retrieved by both modules. The total number of unique exemplars that are retrieved by using the two retrieval modules in parallel is sixteen. Table 10.3 shows the number of exemplars that are retrieved for other combinations of the two experiment variables. The actual exemplars retrieved in these experiments are listed in Appendix C.

Table 10.3: Results obtained for different combinations of experiment variables

		2_planes_with_distance					
Cosine Angle	Struct similarity	Struct	Sem	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**0.50**	12	9	4	17	8	5
0.3	**0.75**	12	9	4	17	8	5
0.5	**0.50**	12	6	3	15	9	3
0.5	**0.75**	12	6	3	15	9	3

As can be seen from Table 10.3, when the lower limit on the cosine angle between exemplars is 0.3, both the modules retrieve nine exemplars each. However, the number of exemplars that are retrieved by both modules is four. This implies that the semantic retrieval module retrieves five exemplars that are not retrieved by the structural retrieval module. Similarly, the structural retrieval module retrieves eight exemplars that are not retrieved by the semantic retrieval module. The user is provided with a combined

list of seventeen exemplars.

When the lower limit of the cosine angle between a target exemplar and the query is 0.5, the semantic retrieval module retrieves six exemplars, of which three exemplars were not retrieved by the structural semantic module. The user is provided with a combined list of fifteen exemplars.

These results show that there is a possibility that the user may not retrieve relevant exemplars if only one similarity module is used. As well, the exemplars that are retrieved by both similarity modules may be considered more relevant to the query compared to the other exemplars. Different ways to combine the two similarity measures and compute an overall similarity measure for each of the retrieved exemplars are discussed later in the chapter. The results of using the two retrieval modules in parallel for the query "gear_pinion_02" are summarized in Table 10.4.

Table 10.4: Results obtained from using retrieval modules in parallel for "gear_pinion_02"

		gear_pinion_02					
Cosine Angle	% Struct similarity	Struct	Sem	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**0.50**	11	6	6	11	5	0
0.3	**0.75**	10	6	6	10	4	0
0.5	**0.50**	11	4	4	11	5	0
0.5	**0.75**	10	4	4	10	4	0

As can be seen from Table 10.4, when the lower limit on the cosine angle between exemplars is 0.3 and the structural similarity threshold is 0.5, the structural similarity module retrieves eleven exemplars and the semantic similarity module is six. However, all exemplars retrieved by the semantic similarity module are also retrieved by the structural similarity module. The user is provided with a combined list of eleven exemplars. However, when the threshold for the structural similarity measure is 0.75, the structural similarity module retrieves only ten exemplars, of which six are also retrieved by the semantic similarity module. In this case, the user is provided with a combined list of ten exemplars. When the lower limit of the cosine angle between a target exemplar and the query is 0.5, the semantic retrieval module retrieves four exemplars, all of which are also retrieved by the structural similarity module.

These results show that for the query, "gear_pinion_02", if the exemplar author uses only the structural similarity module, all the relevant exemplars will be retrieved. However, if the exemplar author uses only the semantic retrieval module, some similar exemplars may not get retrieved.

The results of using the two modules in parallel for the query "q_hole" are summarized in Table 10.5. For this query, both the structural and semantic similarity modules retrieve five exemplars each when the lower limit on the cosine angle value between a target exemplar and the query is 0.3. Of these five exemplars, one exemplar retrieved by the structural module is not retrieved by the semantic similarity module and vice versa. The same exemplars are retrieved by the structural similarity module for all

both threshold values of the structural similarity measure.

Table 10.5: Results obtained from using retrieval modules in parallel for "q_hole"

		q_hole					
Cosine Angle	% Struct similarity	Struct	Sem	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**50**	5	5	4	6	1	1
0.3	**75**	5	5	4	6	1	1
0.5	**50**	5	4	3	6	2	1
0.5	**75**	5	4	3	6	2	1

However, when the lower limit on the cosine value angle between a target exemplar and a query exemplar is 0.5, the semantic similarity module retrieves four exemplars, of which three are retrieved by the structural similarity module. Hence, for this query, if the exemplar author uses only one of the modules to retrieve exemplars at least one exemplar that may be similar to the exemplar being authored may not be retrieved.

Table 10.6 shows the results obtained by using the structural and semantic similarity module in parallel for the query "q_thinwall_thick_less_0.5". As can be observed from the results, when the lower limit on the cosine angle between exemplars is 0.3 and the structural similarity threshold is 0.5, the structural similarity module retrieves three exemplars and the semantic similarity module retrieves eight exemplars. All three

exemplars retrieved by the structural similarity module are also retrieved by the semantic similarity module. The user is provided with a combined list of eight exemplars.

Table 10.6: Results of using retrieval modules in parallel for "q_thinwall_thick_less_0.5"

		q_thickwall_thick_less_0.5					
Cosine Angle	% Struct similarity	Struct	Sem	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**50**	3	8	3	8	0	5
0.3	**75**	3	8	3	8	0	5
0.5	**50**	3	6	3	6	0	3
0.5	**75**	3	6	3	6	0	3

When the lower limit of the cosine angle between a target exemplar and the query is 0.5, the semantic similarity module retrieves six exemplars, of which three are also retrieved by the structural similarity module. In this case, the user is provided with a combined list of six exemplars.

These results show that for the query, "q_thinwall_thick_less_0.5", if the exemplar author uses only the semantic similarity module, all similar exemplars will be retrieved. However, if the exemplar author uses only the structural retrieval module, only three similar exemplars will be retrieved.

Based on the results of retrieval obtained by using the two similarity modules in parallel for the four query exemplars, it is observed that in all cases, there is a possibility,

that by using only one similarity module, the exemplar author may not retrieve some exemplars that may be useful to him. As well, using the two similarity modules in parallel may retrieve more exemplars than the number of exemplars retrieved by the two modules individually. For example, the number of exemplars retrieved for the query, "2_planes_with_distance" by using the two modules in parallel is fourteen when the lower limit on the cosine angle between two exemplars is 0.5 and the lower limit on the structural similarity score is 0.75, which is more than the number of exemplars retrieved by each module individually.

This implies that if exemplars are retrieved using only structural similarity measures or only semantic similarity measures individually, a number of relevant exemplars that may be useful to the exemplar author may not get retrieved.

10.2. Using the two modules in series

As discussed above, one way to combine both the structural and semantic retrieval modules is to use them in parallel on the same query. Another manner by which the two modules can be used in conjunction is to use them in series. Using the two modules in series implies using the two similarity modules one after the other. For example, if the structural retrieval module is used first to retrieve exemplars that are structurally similar to a query exemplar, then the semantic retrieval module is used to retrieve semantically similar exemplars to each of the structurally similar retrieved exemplars. Similarly, if the semantic retrieval module is used first to retrieve exemplars

that are semantically similar to a query exemplar, the structural similarity module is used to retrieve exemplars that are structurally similar to each of the semantically similar retrieved exemplars.

Different experiments were conducted in order to study the results of exemplar retrieval by using the two retrieval modules in series. In this experiment as well, the four query exemplars are "2_planes_with_distance", "gear_pinion_02", "q_hole", and "q_thinwall_thick_less_0.5". In these experiments as well, the variables that can be varied include the cosine value of the angle between vectors and the measure of structural similarity of a target exemplar to the query exemplar. As well, the order in which the retrieval modules are used is interchanged. This implies that in the first experiment, the structural similarity module is used to retrieve structurally similar exemplars to the query exemplar, following which the semantic similarity module is used on each of the structurally similar exemplars. In the second experiment, the semantic similarity module is used first to retrieve semantically similar exemplars to the query exemplar, following which, the structural similarity module is used with each of the semantically similar exemplars as queries. In both experiments, the weighing scheme used for the controlled vocabulary is the one in which terms occurring frequently across all text descriptions carry less weight whereas words that are not common carry higher weights. In order to evaluate the structural similarity between exemplars, all three filters are used before subjecting the exemplar to the pattern matching filter. The filters are set such that the value of the variable "difference" is 0 for the entity filter, -3 for the relation filter, and -3

for the attribute filter. The results of these experiments are discussed in detail in the following sections.

10.2.1. Using the structural similarity module first

In this experiment, the structural retrieval module is used to retrieve exemplars that are structurally similar to a query exemplar. The semantic retrieval module is then used to retrieve exemplars that are semantically similar to each of the structurally similar exemplars. Table 10.7 shows part of the results obtained from using the exemplar "2_planes_with_distance" as query. The entire list of retrieved exemplars is shown in Appendix D.

Table 10.7 has two columns. The first column has the names of exemplars that were retrieved by using the structured retrieval module. The numbers in parentheses next to each retrieved exemplar represents the structural similarity score of each exemplar. As explained before, this number is calculated as a ratio of the number of entities and relations present in the query at the time of a successful match to the total number of entities and relations originally present in the query.

Table 10.7: Partial Results of using the retrieval modules in series.

Query: 2_planes_with_distance.stp	
Structurally Similar Exemplars	**Semantically similar exemplars**
2_planes_with_distance (1)	2_lines_dist_2_planes (0.678)
	2_planes_with_distance (1)
	2_points_2_lines_with_dist (0.5)
	2_ponts_with_distance (0.52)
	planes_lines_points_distance (0.66)
	q_dist_bounded_planes (0.94)
	q_thinwall_parallel_planes (0.42)
	q_thinwall_solid_bounded_planes (0.32)
	q_thinwall_thick_less_0.5 (0.3)
q_boss_radius (1)	q_boss_radius (1)
	q_cylinder (0.47)
q_thinwall_medium_with_boundary (1)	q_cylinder (0.3)
	q_thinwall_medium_with_boundary (1)
	q_thinwall_parallel_planes (0.79)
	q_thinwall_simple_boundary (0.94)
	q_thinwall_solid_bounded_planes (0.87)
	q_thinwall_thick_less_0.5 (0.95)
	q_thinwall_twoloops_simple (0.94)

The second column has the names of the exemplars that are semantically similar to each of the structurally similar exemplars. The number in parentheses besides each of the names represents the cosine value of the angle between the query vector and the target vector. The total number of exemplars retrieved using the two modules in series are summarized in Table 10.8.

Table 10.8: Results from using structural module followed by semantic similarity module

		Query: 2_planes_with_distance					
Cosine Angle	Struct similarity	Struct	Sem	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**0.50**	12	21	12	21	0	9
0.3	**0.75**	12	21	12	21	0	9
0.5	**0.50**	12	18	12	18	0	6
0.5	**0.75**	12	18	12	18	0	6

As can be seen from Table 10.8, when the lower limit on the cosine angle between exemplars is 0.3, and the threshold for the structural similarity measure is 0.5, the semantic retrieval module retrieves twenty one exemplars, of which, nine exemplars are also retrieved by the structural similarity module. This implies that the semantic retrieval module retrieves twelve exemplars that are not retrieved by the structural retrieval module. It should be noted that the semantic retrieval module will always at least retrieve the same number of exemplars as retrieved by the structural retrieval

module. The reason for this is that the exemplars retrieved by the structural retrieval module serve as queries to the semantic retrieval module. The user is provided with a combined list of twenty one exemplars.

When the lower limit of the cosine angle between a target exemplar and the query is 0.5, the semantic retrieval module retrieves eighteen exemplars, of which nine exemplars were not retrieved by the structural semantic module. The user is provided with a combined list of eighteen exemplars.

These results show that there is a possibility that the user may not retrieve some exemplars that may be useful if only one similarity module is used. As well, the exemplars that are retrieved by both similarity modules may be considered more relevant to the query compared to the other exemplars. The number of exemplars retrieved by using the two similarity modules in series in more than the number of exemplars retrieved when the two similarity modules are used in parallel. Different ways to combine the two similarity measures and compute an overall similarity measure for each of the retrieved exemplars are discussed later in the chapter.

The results from using the two modules in series for the query "gear_pinion_02" are shown in Table 10.9. As can be seen from Table 10.9, when the lower limit on the cosine angle between exemplars is 0.3, and the threshold for the structural similarity measure is 0.5, the semantic retrieval module retrieves fifteen exemplars, of which, ten exemplars are also retrieved by the structural similarity module. This implies that the semantic retrieval module retrieves five exemplars that are not retrieved by

the structural retrieval module. In this case, the user is provided with a combined list of fifteen exemplars.

Table 10.9: Results from using the two modules in series for "gear_pinion_02.stp"

		Query: gear_pinion_02					
Cosine Angle	% Struct similarity	Struct	Sem	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**50**	10	15	10	15	0	5
0.3	**75**	10	15	10	15	0	5
0.5	**50**	10	13	10	13	0	3
0.5	**75**	10	13	10	13	0	3

When the lower limit of the cosine angle between a target exemplar and the query is 0.5, the semantic retrieval module retrieves thirteen exemplars, of which ten exemplars were not retrieved by the structural semantic module. In this case, the user is provided with a combined list of thirteen exemplars.

As can be seen from the results, the total number of exemplars that are retrieved using the two modules in series is more than the number of exemplars retrieved by the structural module alone or by using the two modules in parallel. The semantic retrieval module retrieves five exemplars in addition to the number of exemplars already retrieved by the structural retrieval module. This number is reduced to three as the semantic similarity threshold is increased from 0.3 to 0.5. However, the total number of exemplars

retrieved in this case is not as much as the number of exemplars retrieved for the query "2_planes_with_distance".

Table 10.10 shows the results of using the structural similarity module before the semantic similarity module for the query "q_hole.stp".

Table 10.10: Exemplars retrieved for query "q_hole" using structural similarity first

		Query: q_hole					
Cosine Angle	% Struct similarity	Struct	Sem	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**0.50**	5	12	5	12	0	7
0.3	**0.75**	5	12	5	12	0	7
0.5	**0.50**	5	6	5	6	0	1
0.5	**0.75**	5	6	5	6	0	1

From the results obtained for the query "q_hole.stp" it is seen that, when the lower limit on the cosine angle between exemplars is 0.3, and the threshold for the structural similarity measure is 0.5, the semantic retrieval module retrieves twelve exemplars, of which, five exemplars are also retrieved by the structural similarity module. This implies that the semantic retrieval module retrieves seven exemplars that are not retrieved by the structural retrieval module. However, when the lower limit of the cosine angle between a target exemplar and the query is 0.5, the semantic retrieval module retrieves six exemplars, of which five exemplars were already retrieved by the

structural semantic module. As well, the exemplar retrieved by the semantic similarity module is the same exemplar retrieved by the semantic module, when the two similarity modules are used in parallel. Hence, in this case, it is seen that using the two modules in series does not retrieve any more exemplars than when the two modules are used in parallel. Table 10.11 shows the results of using the structural similarity module before the semantic similarity module for the query "q_thinwall_thick_lesss_0.5.stp".

Table 10.11: Results for "q_thinwall_thick_less_0.5.stp" using the structural module first

		Query: q_thinwall_thick_less_0.5.stp					
Cosine Angle	% Struct similarity	Struct	Sem	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**0.50**	3	8	3	8	0	5
0.3	**0.75**	3	8	3	8	0	5
0.5	**0.50**	3	6	3	6	0	3
0.5	**0.75**	3	6	3	6	0	3

From the results obtained for the query "q_thinwall_thick_less_0.5.stp" it is seen that, when the lower limit on the cosine angle between exemplars is 0.3, and the threshold for the structural similarity measure is 0.5, the semantic retrieval module retrieves eight exemplars, of which, three exemplars are also retrieved by the structural similarity module. This implies that the semantic retrieval module retrieves five exemplars that are not retrieved by the structural retrieval module. However, when the lower limit of the

cosine angle between a target exemplar and the query is 0.5, the semantic retrieval module retrieves six exemplars, of which three exemplars were already retrieved by the structural semantic module. As well, the exemplars retrieved by the semantic similarity module are the same exemplars retrieved by the semantic module, when the two similarity modules are used in parallel. Hence, in this case, it is seen that using the two modules in series does not retrieve any more exemplars than when the two modules are used in parallel.

10.2.2. Using the semantic similarity module first

In this set of experiments, the semantic similarity module is first used to retrieve exemplars that are semantically similar to the query exemplar. The structural similarity module is then used to retrieve exemplars that are structurally similar to each of the semantically similar retrieved exemplars. The results obtained on using the exemplar "2_planes_with_distance" as a query are partially listed in Table 10.12. The complete list of retrieved exemplars is shown in Appendix E.

Table 10.12: Partial list of exemplars retrieved from using the semantic retrieval module first

2_planes_with_distance (cosine angle ≥ 0.3, structural similarity ≥ 50%)	
Semantically Similar Exemplars	**Structurally similar exemplars**
2_lines_dist_2_planes (0.678)	2_lines_dist_2_planes (1)
	q_thinwall_simple_boundary (0.75)
2_points_with_distance (0.52)	2_points_2_lines_with_dist (1)
	2_ponts_with_distance (1)
	planes_lines_points_distance (1)
	q_hole_depth_radius (1)
	q_thinwall_simple_boundary (1)
q_thinwall_thick_less_0.5 (0.3)	q_thinwall_medium_with_boundary (1)
	q_thinwall_simple_boundary (1)
	q_thinwall_thick_less_0.5 (1)

Table 10.12 has two columns. The first column has the names of exemplars that were retrieved by using the semantic retrieval module. The number in parentheses besides each of the names represents the cosine value of the angle between the query vector and the target vector. The second column has the names of the exemplars that are semantically similar to each of the structurally similar exemplars. The numbers in parentheses next to each retrieved exemplar represents the structural similarity measure of each exemplar. As explained before, this number is calculated as a ratio of the number of entities and relations present in the query at the time of a successful match to the total

number of entities and relations originally present in the query. The total number of exemplars retrieved using the two modules in series are summarized in Table 10.8.

Table 10.13: Number of exemplars retrieved by using semantic module first for "2_planes_with_dist.des"

		Query: 2_planes_with_distance.des					
Cosine Angle	Struct similarity	Sem	Struct	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**0.50**	9	18	9	18	9	0
0.3	**0.75**	9	18	9	18	9	0
0.5	**0.50**	6	16	6	16	10	0
0.5	**0.75**	6	16	6	16	10	0

As can be seen from Table 10.13, when the lower limit on the cosine angle between exemplars is 0.3, and the threshold for the structural similarity measure is 0.5, the structural retrieval module retrieves eighteen exemplars, of which, nine exemplars are also retrieved by the semantic similarity module. This implies that the structural retrieval module retrieves nine exemplars that are not retrieved by the semantic retrieval module. It should be noted that the structural retrieval module will always at least retrieve the same number of exemplars as retrieved by the structural retrieval module. The reason for this is that the exemplars retrieved by the semantic retrieval module serve as queries to the structural retrieval module. In this case, the user is provided with a combined list of eighteen exemplars. As well, the number of exemplars retrieved by using the structural

module first is more than the number of exemplars retrieved by the using the semantic module first.

When the lower limit of the cosine angle between a target exemplar and the query is 0.5, the structural retrieval module retrieves sixteen exemplars, of which six exemplars were retrieved by the structural semantic module. The user is provided with a combined list of sixteen exemplars. In this case, the number of exemplars retrieved by using the semantic module first is more than the number of exemplars retrieved by using the structural module first. Different ways to combine the two similarity measures and compute an overall similarity measure for each of the retrieved exemplars are discussed later in the chapter. Table 10.14 shows the number of exemplars retrieved for the query "gear_pinion_02.des".

Table 10.14: Results obtained by using the semantic module first for "gear_pinion_02.des"

		Query: gear_pinion_02.des					
Cosine Angle	% Struct similarity	Sem	Struct	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**50**	6	10	6	10	4	0
0.3	**75**	6	10	6	10	4	0
0.5	**50**	4	10	4	10	6	0
0.5	**75**	4	10	4	10	6	0

As can be seen from Table 10.14, when the lower limit on the cosine angle between exemplars is 0.3, and the threshold for the structural similarity measure is 0.5, the structural retrieval module retrieves ten exemplars, of which, six exemplars are also retrieved by the semantic similarity module. This implies that the structural retrieval module retrieves four exemplars that are not retrieved by the semantic retrieval module. In this case, the user is provided with a combined list of ten exemplars.

When the lower limit of the cosine angle between a target exemplar and the query is 0.5, the structural retrieval module retrieves ten exemplars, of which only four exemplars are retrieved by the semantic retrieval module. In this case as well, the user is provided with a combined list of ten exemplars.

As can be seen from the results, the total number of exemplars that are retrieved using the two modules in series is the same as than the number of exemplars retrieved by using the structural module first. However, the total number of exemplars retrieved in this case is not as much as the number of exemplars retrieved for the query "2_planes_with_distance".

Table 10.15 shows the number of exemplars retrieved for the query "q_hole.des". From the results obtained, it is seen that, when the lower limit on the cosine angle between exemplars is 0.3, and the threshold for the structural similarity measure is 0.5, the structural retrieval module retrieves six exemplars, of which, five exemplars are also retrieved by the semantic similarity module. This implies that the semantic retrieval module retrieves only one exemplar that is not retrieved by the structural

retrieval module.

Table 10.15: Results obtained by using the semantic module first for "q_hole.des"

		Query: q_hole.des					
Cosine Angle	% Struct similarity	Sem	Struct	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**50**	5	6	5	6	1	0
0.3	**75**	5	6	5	6	1	0
0.5	**50**	4	6	4	6	2	0
0.5	**75**	4	6	4	6	2	0

When the lower limit of the cosine angle between a target exemplar and the query is 0.5, the structural retrieval module retrieves six exemplars, of which four exemplars were already retrieved by the semantic module. As well, the exemplars retrieved are the same exemplars that are retrieved when the structural similarity module is used in first. Hence, in this case, it is seen that using the semantic similarity module first does not retrieve any more exemplars than the number of exemplars retrieved by using the structural similarity module.

Table 10.16 shows the number of exemplars retrieved for the query "q_thinwall_thick_less_0.5.des". From the results obtained, it is seen that, when the lower limit on the cosine angle between exemplars is 0.3, and the threshold for the structural similarity measure is 0.5, the structural retrieval module retrieves sixteen

exemplars, of which, eight exemplars are also retrieved by the semantic similarity module.

Table 10.16: Results obtained by using the semantic module first for "q_thinwall_thick_less_0.5.des"

		Query: q_thinwall_thick_less_0.5.des					
Cosine Angle	% Struct similarity	Sem	Struct	Struct ∩ Sem	Struct U Sem	Struct – Sem	Sem - Struct
0.3	**50**	8	16	8	16	8	0
0.3	**75**	8	16	8	16	8	0
0.5	**50**	6	16	6	16	10	0
0.5	**75**	6	16	6	16	10	0

When the lower limit of the cosine angle between a target exemplar and the query is 0.5, the structural retrieval module retrieves sixteen exemplars, of which six exemplars were already retrieved by the semantic module. As well, the number of exemplars retrieved is more than the number of exemplars retrieved when the structural similarity module is used in first.

10.3. Computation of a combined similarity measure

As described in the previous section, the different ways of incorporating two views of similarity while retrieving exemplars from a database that are similar to a query exemplar, are to either use the two similarity modules in parallel or in series. Having

conducted experiments to evaluate the different ways to conjoin the structural and semantic retrieval modules, the next step is to identify and evaluate different ways to develop an overall similarity score that takes into account both views of similarity. Both the structural and semantic retrieval modules assign a similarity measure to the retrieval modules. The semantic similarity measure is the value of the cosine angle between the vector representation of a target exemplar and the vector representation of a query exemplar. The structural similarity score of a target exemplar is the number of entities and relations left in a query exemplar at the time of a successful graph match to the number of entities and relations originally present in the query exemplar.

There may be different ways of combining these scores to assign an overall similarity score to the retrieved exemplar. These include adding the two scores together, taking an average of the two scores, multiplying the two scores, or using these scores to calculate the distance between the retrieved exemplar and the query exemplar.

Table 10.17 shows a list of exemplars for which both the structural similarity score and the semantically similar score is known. The objective is to evaluate different ways of combining both these scores in order to assign an overall similarity score to the retrieved exemplars. The values listed are for purposes of illustration only. The table also shows the overall similarity score assigned to each exemplar obtained by different methods listed above. Each of the listed methods is discussed with respect to the similarity scores shown in Table 10.17.

Table 10.17: Different ways to compute overall similarity score

Exemplars	Sem	Struct	Sum	Product	Average	Distance
A	**0.01**	**1**	1.01	0.01	0.505	0.99
B	**0.3**	**0.2**	0.5	0.06	0.25	1.06
C	**0.6**	**0.4**	1.0	0.24	0.5	0.72
D	**0.5**	**0.5**	1.0	0.25	0.5	0.707
E	**0.1**	**0.9**	1.0	0.09	0.45	0.905
F	**0.8**	**0.9**	1.7	0.72	0.85	0.22
G	**1**	**0.01**	1.01	0.01	0.505	0.99

10.3.1. Sum of semantic and similarity scores

One way of computing the overall similarity score of an exemplar is to add the structural and semantic similarity scores. For example, exemplar B in Table 10.17 has a structural similarity score of 0.2 and a semantic similarity score of 0.3. Hence the overall score that exemplar B has is 0.5. Similarly, exemplar E has a structural similarity score of 0.9 and a structural semantic similarity score of 0.1. Hence the overall similarity score of exemplar E is 1.

From Table 10.17 it can be seen that on summation of the similarity scores, exemplars that score high on one measure and low on the other measure have the same overall similarity score as those exemplars that score equally on both measures. This implies that these exemplars are equally similar to the query exemplar. The summation of

the semantic and structural similarity shows that the exemplars C, D, and E are equally similar to a query exemplar. Similarly, exemplars A and G are considered equally similar. The results imply that all these exemplars are almost equally similar to the query exemplar. Intuitively, exemplars that score high on both measures should have a high overall similarity score, whereas exemplars that score low on both measures should score a low overall similarity measure. Summation of the two similarity scores achieves this objective, as is observed in the overall scores obtained by exemplars B and F. However for exemplars that score equally on either the structural and semantic similarity measures, or score high on one of the measures and low on the other, this method is not effective in developing a clear distinguishing measure of similarity.

10.3.2. Product of Structural and Semantic similarity measures

A second way of computing the overall similarity between exemplars is to take a product of the two scores. In this case, the overall similarity score will almost always be less than each of the two measures for every exemplar since in this case, two fractions are being multiplied. In this case as well, exemplars that score high on both measures or score low on both measures have the highest and lowest overall similarity score. However, this method penalizes an exemplar for scoring low on either one of the measures. For example, exemplar A has a score of 1 on the structural similarity measure. However the product of the two measures assigns a score of 0.01 to this exemplar because it has a score of 0.01 on the semantic measure. This implies that a high score on

the structural similarity measure does not account for anything. This is not a correct interpretation and hence this method may not be a good way of assessing the overall similarity of two exemplars.

10.3.3. Average of Structural and Similarity measures

One way of computing the overall similarity measure of an exemplar is to take an average of the two similarity measures. Of all the different ways listed to compute the overall similarity between a query exemplar and a target exemplar, this method offers the most logical interpretation. In this method, the overall similarity score is computed by taking the average of the two similarity measures. This ensures that both the structural and semantic similarity measures are given equal importance in computing the final score. For example, the overall similarity score for exemplar A is 0.505, which is the same as exemplar D. This means that exemplars A and D are equally similar to the query exemplar. This interpretation seems logically correct since exemplar A is penalized for scoring low on the semantic measure and exemplar D is not given an undue advantage because it scores equally on both measures. As well, exemplars that score high on both measures also score high on the overall similarity measure. Exemplars that score low on both measures score low on the combined measure as well.

10.3.4. Computation of distance between target exemplar and query

In this case, the overall similarity is computed as the distance between each retrieved exemplar and the query exemplar. The structural and semantic similarity scores

of the query itself are both one since the query will be most similar to itself. A low overall similarity score obtained by this method implies that the target exemplar is close to the query exemplar and hence has a high measure of similarity. Similarly a high score obtained by this method indicates that the target exemplar lies further away from the query exemplar. For example, exemplar C has a semantic similarity measure of 0.6 and a structural similarity measure of 0.4. Hence the distance between this exemplar and the query exemplar is calculated as the square root of the term $((1-0.6)^2 + (1-0.4)^2)$. Thus the distance between exemplar C and the query is calculated to be 0.72.

From the results it is observed that exemplar F that scores high on both measures of similarity has the least distance from the query, whereas exemplar B which scores low on both measures of similarity lies furthest from the query exemplar. However, exemplar A and G score high on one measure whereas exemplar B scores low on both measures. Yet, as the overall similarity measure would suggest, these exemplars are equally similar to the query exemplar.

The different methods of computing an overall similarity score are applied to the query "q_thinwall_thick_less_0.5" with both modules of similarity applied in parallel. The results are shown in Table 10.18.

Table 10.18: Conjoined similarity measures for "q_thinwall_thick_less_0.5" (parallel)

Exemplars	Sem	Struct	Sum	Product	Average	Distance
A	**0.3**	**0.01**	0.31	0.003	0.155	1.21
B	**0.39**	**0.01**	0.4	0.039	0.2	1.16
C	**0.95**	**1**	1.95	0.95	0.975	0.05
D	**0.76**	**0.01**	0.77	0.076	0.385	1.01
E	**0.93**	**1**	1.93	0.93	0.965	0.07
F	**0.92**	**0.01**	0.93	0.092	0.465	0.99
G	**1**	**1**	2	1	1	0
H	**0.93**	**0.01**	0.94	0.093	0.47	0.99

As can be seen from the results, all exemplars that score high on both measures of similarity attain a high combined score of similarity through all four ways of computing the overall similarity score. Similarly, exemplars that score low on both measures get a low combined score. However, as noted before, the product of the two measures penalizes those exemplars that score a high on one measure and low on the other. The method of taking the average of the two measures ensures that both measures of similarity are given equal importance in computing the overall score. This gives a more accurate measure of the overall similarity as compared to other methods.

This discussion is not aimed at evaluating which is the best way to compute the overall similarity. Rather, this discussion is aimed at understanding similarity in graph-

based models. If the exemplar author is interested in retrieving exemplars that score high on both measures of similarity, then any of the four measures described above can be used to compute the overall similarity measure. Similarly, if the exemplar author is interested in retrieving exemplars that barely pass the structural and semantic similarity filters, any of the methods described above can be used to compute the overall similarity value. However, if the exemplar author is interested in retrieving exemplars that score high on one measure and low on the other, or exemplars that score relatively the same on both measures, it does not become immediately clear which method to use. However, the method that should be used would depend on factors such as user preferences, the context in which similarity is being evaluated, and the application involved. More experiments may need to be conducted in order to arrive at a conclusion.

10.4. Summary

The experiments described in this chapter are aimed at evaluating different ways of using the structural and semantic similarity measures in conjunction. Using the two similarity modules in conjunction implies incorporating two views of similarity while retrieving similar exemplars. The two similarity modules can either be used in parallel on the same query or in series.

It can be seen from the results that when the two modules are used in parallel, there is a possibility, that by using only one similarity module, the exemplar author may not retrieve some exemplars that may be useful to him. This is seen from the exemplars

retrieved for the queries “2_planes_with_distance”, and “q_hole”. However, in some cases, the structural similarity module may retrieve all exemplars retrieved by the semantic similarity module and vice versa. This is seen from the exemplars retrieved for the queries “gear_pinion_02” and “q_thinwall_thick_less_0.5”. Hence, it can be inferred that using the two similarity modules in conjunction is advantageous.

Apart from retrieving exemplars not retrieved by one similarity module, a secondary objective of using the two similarity modules in series is to increase the number of exemplars retrieved. This is observed from the results obtained for the queries, “2_planes_with_distance”, and “gear_pinion_02”. However, the number of exemplars retrieved for the queries “q_hole” and “q_thinwall_thick_less_0.5” is the same as the exemplars retrieved when the similarity modules are used in parallel. Hence, it may be inferred that for a database of thirty exemplars, using the two similarity modules in series may not be advantageous compared to using them in parallel. However, for large databases, using the similarity modules in series may retrieve more exemplars than using the two modules in parallel.

The structural and semantic similarity score may be combined in different ways to provide an overall similarity score to the retrieved exemplars. From

Table 10.17 it can be seen that taking an average of the two scores, or calculating the distance between each target exemplar and the query exemplar may be appropriate.

Chapter 11

ROBUSTNESS OF THE EXEMPLAR RETRIEVAL SYSTEM

The experiments described in chapters 8, 9, and 10 were aimed at evaluating the accuracy and effectiveness of the structural and semantic similarity measures. The results obtained from those experiments helped in understanding similarity from the design exemplar perspective. This chapter describes the experiments that were conducted in order to evaluate the 'goodness' or the 'robustness' of the exemplar retrieval system. These experiments are aimed at understanding the difference in the results obtained with different users using the exemplar search and retrieval tool.

As described earlier, the intent and rationale for authoring an exemplar in a specific manner is represented textually. Therefore, different exemplar authors may write the intent and rationale of an exemplar in different ways. Hence, it is important to verify whether the semantic similarity measures retrieve all the variations of a target exemplar. The objective of having multiple variations of the query exemplars is to observe whether similar results were obtained for each variation of a query. Having multiple variations of an exemplar in the database helps determine the robustness of the exemplar retrieval system by observing whether all variations of a target exemplar get retrieved. Further, the results obtained from this experiment help evaluate the effect that certain characteristics of a query or a target exemplar, such as size of a structured exemplar and length of a semantic description, may have on the results of retrieval.

In order to test the goodness of the exemplar retrieval system, multiple variations of some exemplars were included in the database. Four people were asked to author variations of exemplars in order to overcome any bias that may result from any one person author all variations. The expertise levels of all four people are summarized in Table 11.1.

Table 11.1: Expertise levels of four users of the design exemplar

User	**Expertise in design exemplar authoring**	**Availability**
1. Joshua D. Summers	Eight years	Associate Professor at Clemson University, SC
2. Sudhakar Teegavarapu	Two weeks	Graduate student at Clemson University
3. Shashidhar Putti	Three years	Graduated in 2007 from Clemson University, working at MN
4. Vikram Bapat	Four years	Graduated in 2008 from MTU, working at MI
5.Srinivasan Anandan (original author)	Four years	Graduate student at Clemson University

Specifically, these users were provided with structured representations of the four query exemplars, for which they wrote their own semantic descriptions using the controlled vocabulary. All four people were asked to write semantic descriptions of the query, “2_planes_with_distance”. Of the four users, three users were asked to write the

semantic descriptions of the exemplars “gear_pinion_02”, “q_hole”, and “q_thinwall_thick_less_0.5”. Two users were also asked to write semantic descriptions of the exemplars, “q_boss_radius”, “q_coplanar_gears”, and “q_hole_depth_radius”. All these variations were included in the database while running the experiments.

As mentioned earlier, different exemplar authors may write the semantic description of an exemplar in different manners. Hence, variability may be introduced in the retrieval results due to multiple semantic descriptions of the same exemplar. Therefore, it is highly desirable that the semantic similarity measures retrieve all variations of an exemplar. However, the structural similarity filters retrieve all target exemplars that are structurally similar to the query exemplar depending on the type of entities and relations present in the query. Hence, for purposes of evaluating the robustness of the structural similarity measures, the entire database is manually parsed in order to verify that the results are predictable. For this experiment, only one user was asked to author structural variations of the exemplars, “2_planes_with_distance”, “gear_pinion_02”, “q_hole”, and “q_thinwall_thick_less_0.5”.

In order to test the robustness of the exemplar retrieval tool, different variations of the exemplars, “2_planes_with_distance”, “gear_pinion_02”, “q_hole”, and “q_thinwall_thick_less_0.5” were used as queries. These are the same exemplars that were used as queries in the earlier experiments. Using the same queries ensured that results obtained in these experiments are comparable to the results obtained in the previous experiments.

The different semantic descriptions written by the users were included in the database for this experiment. Table 11.2 lists the names and numbers of these exemplars in the database.

Table 11.2: Names of exemplars authored by different users included in the database

Exemplar Number	**Name of Exemplar**
W2_1	2_planes_with_distance_summers.stp; 2_planes_with_distance_summers.des
W2_2	2_planes_with_distance_sudhakar.des
W2_3	2_planes_with_distance_shashi.des
W2_4	2_planes_with_distance_bapat.des
G1_1	gear_pinion_02_summers.stp; gear_pinion_02_summers.des
G1_2	gear_pinion_02_sudhakar.des
G1_3	gear_pinion_02_shashi.des
F1_2	q_boss_radius_sudhakar.des
F1_4	q_boss_radius_bapat.des
G4_2	q_copalanar_gears_sudhakar.des
G4_4	q_copalanar_gears_bapat.des
F4_1	q_hole_summers.stp; q_hole_summers.des
F4_2	q_hole_sudhakar.des
F4_3	q_hole_shashi.des
F5_2	q_hole_depth_radius_sudhakar.des
F5_4	q_hole_depth_radius_bapat.des

W9_1	q_thinwall_thick_less_0.5_summers.stp; q_thinwall_thick_less_0.5_summers.des;
W9_2	q_thinwall_thick_less_0.5_sudhakar.des;
W9_3	q_thinwall_thick_less_0.5_shashi.des;

The total number of variations of these exemplars including the original structured exemplars and their semantic descriptions is listed in Table 11.3. The semantic descriptions written by each user is listed in Appendices N, O, P, and Q.

Table 11.3: Number of variations of each exemplar

Exemplar	**Number of variations**	
	Semantic	**Structural**
2_planes_with_distance	5 (users 1, 2, 3, 4, 5)	2 (users 1, 5)
gear_pinion_02	4 (users 1, 2, 4, 5)	2 (users 1, 5)
q_hole	4 (users 1, 2, 4, 5)	2 (users 1, 5)
q_thinwall_thick_less_0.5	4 (users 1, 2, 4, 5)	2 (users 1, 5)
q_boss_radius	3 (users 3, 4, 5)	1 (user 1)
q_coplanar_gears	3 (users 3, 4, 5)	1 (user 1)
q_hole_depth_radius	3 (users 3, 4, 5)	1 (user 1)

Figure 11.1 shows the semantic description of the exemplar "q_hole" which was originally present in the database.

*The intent of this exemplar is to **find holes** in a model. The exemplar consists of a **cylindrical surface** that is bound by two **circles** and two **planes**, each bound by one of the **circles**. These **entities form** the **structure** of the **hole**. An **id relation** is applied to the **cylindrical surface**. When this exemplar is applied to any model all **cylindrical holes** in the model will get **highlighted**.*

Figure 11.1: Original semantic description of exemplar "q_hole"

As can be seen from Figure 11.1, the total number of words in the semantic description is sixty nine, of which eighteen are part of the controlled vocabulary. Figure 11.2 shows the semantic description of the same exemplar as written by user 4. In this case, the total number of words is forty eight, of which thirteen words are part of the controlled vocabulary.

*The intent of this exemplar is to locate all the **holes** in a given model. A **cylindrical surface** bound by **two circles** is located in the model. The two **circles** are **coincident** of two **planes** respectively. The **query extracts** the **cylindrical surface** and **highlights** it in the display.*

Figure 11.2: Semantic description of exemplar "q_hole" written by user 4.

Comparing the two semantic descriptions, it can be seen that the semantic description written by user 4 is shorter. As well, the words “holes”, “cylindrical”, “surface”, and “planes” are common in both descriptions.

Figure 11.3 shows the exemplar “q_hole.stp” originally present in the database, whereas Figure 11.4 shows the structured variation of the same exemplar authored by user 1. As can be seen from the two exemplars, exemplar “q_hole” has five entities and four relations, whereas the exemplar “q_hole_summers” has eight entities and seven relations. Both exemplars contain a cylindrical surface bound by two circles. However, exemplar “q_hole_summers.stp” has two surface normals that are opposite in direction to each other. All exemplars authored by user 1 are listed in Appendix F.

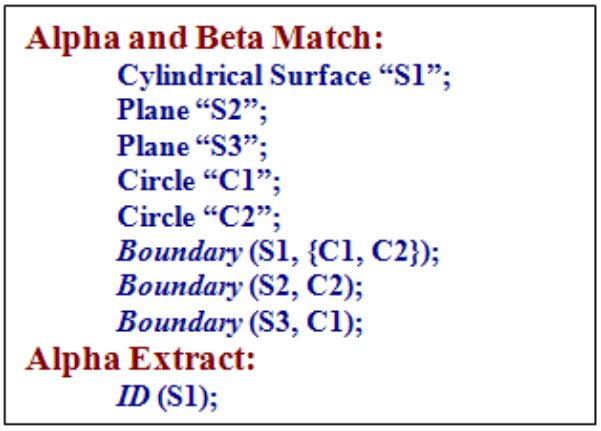


Figure 11.3: Exemplar “q_hole” originally present in the database

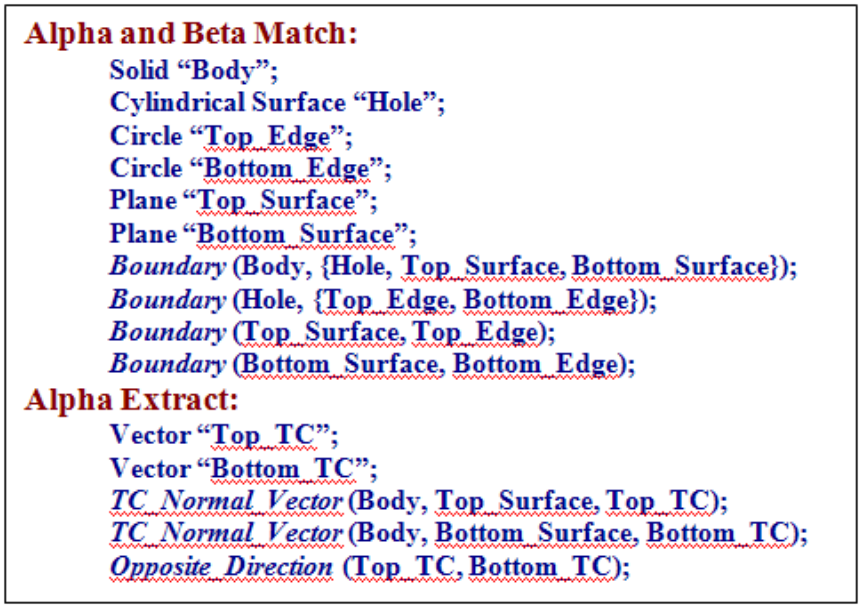


Figure 11.4: Exemplar "q_hole_summers" authored by user 1

11.1. Verifying the robustness of the semantic similarity measures

The semantic description of each exemplar written by each user was used as query in order to retrieve semantically similar exemplars. The results obtained for each description was compared to the results obtained for the original query. Table 11.4 shows the number of exemplars that were retrieved for each variation of the query "2_planes_with_distance", and the total number of unique exemplars that were retrieved.

Table 11.4: Exemplars retrieved for different variations of the query "2_planes_with_distance"

Query: 2_planes_with_distance (cosine angle ≥ 0.3)		
Query		**Number of Semantically similar exemplars**
Original query	2_planes_with_distance	**16** (W1, W2, W2_1, W2_2, W2_3, W2_4, S2, S3, W3, W4, W6, W8, W9, W9_1, W9_2, W9_3)
User 1	2_planes_with_distance_summers	**15** (W1, W2, W2_1, W2_2, W2_3, W2_4, S2, W3, W4, F4, F4_1, F4_2, W6, W9_2, W9_3)
User 2	2_planes_with_distance_sudhakar	**15** (W1, W2, W2_1, W2_2, W2_3, W2_4, S2, S3, W3, W4, W6, W8, W9_1, W9_2, W9_3)
User 3	2_planes_with_distance_shashi	**15** (W1, W2, W2_1, W2_2, W2_3, W2_4, S1, S2, S3, W3, W4, W6, W9_1, W9_2, W9_3)
User 4	2_planes_with_distance_bapat	**15** (W1, W2, W2_1, W2_2, W2_3, W2_4, S2, S3, W3, W4, W6, W8, W9_1, W9_2, W9_3)
(original query ∩ User 1 ∩ User 2 ∩ User 3 ∩ User 4)		**12** (W1, W2, W2_1, W2_2, W2_3, W2_4, S2, W3, S4, W9_1, W9_2, W9_3)
(original query U User 1 U User 2 U User 3 U User 4)		**20** (W1, W2, W2_1, W2_2, W2_3, W2_4, S1, S2, S3, W3, W4, F4, F4_1, F4_2, W6, W8, W9, W9_1, W9_2, W9_3)

It should be noted that the database now contains the variations authored by the different users in addition to the thirty exemplars already present in the database. This

implies that there are forty nine exemplars present in the database. The numbers in the parentheses indicate the exemplars that were retrieved by each query. As can be seen from Table 11.4, twelve exemplars were retrieved in common by the semantic descriptions of all users. A total of twenty unique exemplars were retrieved by the semantic descriptions of all the users. This implies that multiple semantic descriptions of the same query exemplar do not retrieve the same target exemplars. In order to understand the difference between the results of retrieval,

Table 11.5 compares the exemplars retrieved for each variation of the query to the number of exemplars retrieved by the original query. In this experiment, the minimum value of the cosine angle between the vector of a query exemplar and the vector of a target exemplar in order for an exemplar to be retrieved is 0.3.

The total number of unique exemplars retrieved by the original query and the query written by "user 1" is nineteen, of which twelve exemplars were retrieved by both queries. The total number of unique exemplars retrieved by the original query and the query written by "user 2" is sixteen, of which, fifteen are retrieved by both queries. Similarly, the total number of unique exemplars retrieved by the original query and the query written by "user 3" is seventeen, of which fourteen exemplars were retrieved by both queries. The number of exemplars that were retrieved by both the original query and the query written by "user 4" is fifteen. As well, the cosine values of the exemplars obtained for the semantic descriptions written by users 2, 3, and 4 are similar.

Table 11.5: Number of exemplars retrieved by each variation of the query in comparison to the original query

Query: 2_planes_with_distance (cosine angle ≥ 0.3)			
	Number of exemplars retrieved		**Number of exemplars retrieved**
(original query U User 1)	19	(original query ∩ User 1)	12
(original query U User 2)	16	(original query ∩ User 2)	15
(original query U User 3)	17	(original query ∩ User 3)	14
(original query U User 4)	16	(original query ∩ User 4)	15

Based on the results shown in Table 11.4 and Table 11.5, it can be seen that the semantic description written by user 1 retrieves substantially different exemplars as compared to the semantic descriptions written by other users. Figure 11.5 shows the semantic description written by user 1. As can be seen from the semantic description, the total number of words in the description is sixty seven, of which fifteen are part of the controlled vocabulary. As in all semantic descriptions, some words are repeated.

*This exemplar is used to **find** the **distance** between two **parallel planes.** Both **planes** are **highlighted** for the user and the **distance** value is shown the user. If no **planes** are found that are **parallel**, then no **matches** will be found. This exemplar may be used as a basis for finding **thin walls**, for finding the **heights** of b**osses**, or for finding the **depths** of blind **holes.***

Figure 11.5: Semantic description of the query "2_planes_with_distance" by User 1.

On comparison of the semantic description written by user 1 with the original description of the query (Figure 9.1), it is observed that both descriptions describe the intent clearly. However, in the original description the working of the exemplar in described in detail, whereas user 1 describes the rationale in a concise manner. As well, user 1 identifies alternate uses of the exemplar which results in exemplars 28, 34, and 35 getting retrieved. These exemplars are alternate variations of the exemplar to find holes.

As well, the expertise level of an exemplar author may influence the way the semantic descriptions are written. From Table 11.1, it is seen that user 1 has the highest expertise level compared to all other users. Hence, there is a possibility that user 1 can identify potential uses of an exemplar and hence may feel the need to include that information in the semantic description. The results obtained for the other queries may help identify a trend in the results of retrieval.

The exemplars retrieved for different variations of the query "gear_pinion_02" are listed in Table 11.6.

Table 11.6: Exemplars retrieved for different variations of the query "gear_pinion_02"

Query: gear_pinion_02 (cosine angle ≥ 0.3)		
Query		**Number of Semantically similar exemplars**
Original query	gear_pinion_02	10 (G1, G1_1, G1_2, G1_3, G2, G3, G4_2, S4, S5, G5)
User 1	gear_pinion_02_summers	8 (G1, G1_1, G1_2, G1_3, G2, G3, G4_2, S4, S5, G5)
User 2	gear_pinion_02_sudhakar	10 (G1, G1_1, G1_2, G1_3, G2, G3, G4, G4_2, G4_4, G5)
User 3	gear_pinion_02_shashi	10 (G1, G1_1, G1_2, G1_3, G2, G3, G4, G4_2, G4_4, G5)
(original query ∩ User 1 ∩ User 2 ∩ User 3)		8 (G1, G1_1, G1_2, G1_3, G2, G3, G4_2, S4, S5, G5)
(original query U User 1 U User 2 U User 3)		12 (G1, G1_1, G1_2, G1_3, G2, G3, G4, G4_2, G4_4, S4, S5, G5)

As can be seen from these results, twelve unique exemplars were retrieved by the different variations of the query, of which, eight common exemplars were retrieved by each variation. It can be seen from the results, it is seen that there is an 80% overlap in the exemplars retrieved by all semantic descriptions. This is illustrated by comparing the number of exemplars retrieved by each user with the number of exemplars retrieved by the original query (Table 11.7).

Table 11.7: Number of exemplars retrieved by each variation of the query "gear_pinion_02" in comparison to the original query

Query: gear_pinion_02 (cosine angle ≥ 0.3)			
	Number of exemplars retrieved		**Number of exemplars retrieved**
(original query U User 1)	10	(original query ∩ User 1)	8
(original query U User 2)	12	(original query ∩ User 2)	8
(original query U User 3)	12	(original query ∩ User 3)	8

It is observed that exemplars 36 and 37 are only retrieved by the original semantic description. These exemplars are not retrieved by any of the other semantic descriptions. As well, the semantic descriptions written by users 2 and 3 retrieve the same exemplars. Figure 11.6 and Figure 11.7 show the semantic descriptions written by users 2 and 3 respectively.

*The intent of this exemplar is to identify the **gear** and pinion among two meshing **gears**. The **query** locates two **circles** of different **radii** in a model and compares them using an **equation**. The **equation** imposes a **constraint** that the **radius** of the **gear** is greater than the **radius** of the pinion. However, it is not clear how the information is **highlighted** to the user as there are no **IDs**.*

Figure 11.6: Semantic description of "gear_pinion_02" written by user 2

*This exemplar may be used to identify and tag the **gear** and pinion in a **gear** pair. When a model is queried against this exemplar, it returns a **match** if the model has two **gears**, represented as **circles** C1, C2 in the **alpha match** portion of the exemplar and if the **radius** of one **gear** is greater than the other, represented by the **Equation** "eq1"(gear>pinion) **relation** in the **extract** portion of the exemplar.*

Figure 11.7: Semantic description of "gear_pinion_02" written by user 3

The length of the semantic description written by user 2 is seventy, whereas the length of the description written by user 3 is seventy three. As well, the words that are part of the controlled vocabulary in each description are the same. Hence, the exemplars retrieved for both semantic descriptions are the same. Figure 11.8 shows the semantic description written by user 1. As can be seen, the length of the description is fifty one and of these fifty one words, four words are part of the controlled vocabulary.

*This exemplar extracts the **radii** of two **circles**, comparing the values. If one is larger than the second, it can be assumed to be the **gear** with the smaller one potentially the pinion. This exemplar is a simple one that can be expanded for use in **gear** train design and sizing.*

Figure 11.8: Semantic description of "gear_pinion_02" written by user 1

Hence, even for this query, it is seen that users 2 and 3 describe the rationale in more detail compared to user 1. In this case as well, since user 1 is more experienced than the other users, the user identifies potential uses of the same exemplar. As well, only four words in the semantic description are part of the controlled vocabulary. This may be a possible explanation for the difference in the results of retrieval.

Table 11.8 shows the total number of exemplars obtained from using different variations of the exemplar "q_hole" as queries.

Table 11.8: Number of exemplars obtained for different variations of the query "q_hole"

Query: q_hole (cosine angle ≥ 0.3)		
Query		**Number of Semantically similar exemplars**
Original query	q_hole	11 (W2_1, F1_4, F3, F4, F4_1, F4_2, F4_3, F5, F5_2, F5_4, F6)
User 1	q_hole_summers	4 (W2_1, F4, F4_1, F4_2)
User 2	q_hole_sudhakar	9 (S1, G4_2, F4, F4_1, F4_2, F5, F5_2, F5_4, F6)
User 3	q_hole_shashi	9 (W2_1, F1_4, F4, F4_3, F5, F5_2, F5_4, F6, F7)
(Original query ∩ User 1 ∩ User 2 ∩ User 3)		1 (F4)
(Original query U User 1 U User 2 U User 3)		14 (W2_1, S1, F1_4, G4_2, F3, F4, F4_1, F4_2, F4_3, F5, F5_2, F5_4, F6, F7)

As can be seen from the results, only one exemplar is retrieved in common by all the different queries. For this query it is seen that if user 1 is not considered, then the number of common exemplars retrieved by all other users increases from one to five.

Table 11.9 compares the results obtained from each variation to the results obtained by using the original query.

Table 11.9: Number of exemplars retrieved by each variation of the query "q_hole" in comparison to the original query

Query: q_hole (cosine angle ≥ 0.3)			
	Number of exemplars retrieved		**Number of exemplars retrieved**
(Original query U User 1)	11	(User 0 ∩ User 1)	3
(Original query U User 2)	10	(User 0 ∩ User 2)	8
(Original query U User 3)	12	(User 0 ∩ User 3)	7

Figure 11.9 shows the semantic description written by user 1. The length of the semantic description is thirty eight, of which, nine words are part of the controlled vocabulary.

This exemplar is used to find potential ***holes*** *or* ***bosses.*** *It matches* ***cylindrical surfaces*** *that are in turn bounded by two* ***circles,*** *who in turn bound two* ***planes.*** *The* ***cylindrical surface*** *is then returned to the user* ***(highlighted).***

Figure 11.9: Semantic description of exemplar "q_hole" written by user 1

Figure 11.10 shows the semantic descriptions for the query "q_hole" written by

user 2 and user 3. The length of the description written by user 2 is forty eight words, of which eleven words are part of the vocabulary.

*The intent of this exemplar is to locate all the **holes** in a given model. A **cylindrical surface** bound by two **circles** is located in the model. The two **circles** are **coincident** of two **planes** respectively. The query **extracts** the **cylindrical surface** and **highlights** it in the display.*

Figure 11.10: Semantic description of exemplar "q_hole" written by user 2

Comparing the semantic descriptions it can be seen that user 1 has a more concise description. As well, user 1 does not identify the intent of this specific exemplar clearly. In fact, he identifies multiple uses of the same exemplar. This may be a reason for the variation in the results obtained.

Similarly, Table 11.10 shows the total number of exemplars obtained from using different variations of the exemplar "q_thinwall_thick_less_0.5" as queries.

Table 11.10: Number of exemplars obtained for different variations of the query "q_thinwall_thick_less_0.5"

Query: q_thinwall_thick_less_0.5 (cosine angle ≥ 0.3)		
Query		**Number of Semantically similar exemplars**
Original Query	q_thinwall_thick_less_0.5	9 (W2, F3, W5, W6, W7, W8, W9, W9_2, W10)
User 1	Q_thinwall_thick_less_0.5_summers	8 (W1, W2, W2_2, W2_4, S2, S3, W3, W9_1)
User 2	q_thinwall_thick_less_0.5_sudhakar	14 (W2, W2_1, W2_2, W2_3, W2_4, W4, W5, W6, W7, W8, W9, W9_2, W9_3, W10)
User 3	q_thinwall_thick_less_0.5_shashi	13 (W1, W2, W2_1, W2_2, W2_3, W2_4, S2, W3, W4, W6, W8, W9_2, W9_3)
(Original Query ∩ User 1 ∩ User 2 ∩ User 3)		1 (W2)
(Original Query U User 1 U User 2 U User 3)		20 (W1, W2, W2_1, W2_2, W2_3, W2_4, S2, S3, W3, F3, W4, W5, W6, W7, W8, W9, W9_1, W9_2, W9_3, W10)

A total of twenty exemplars were retrieved by the different users. In this case, only one exemplar was retrieved by all users and that was the original query. Table 11.11 compares the results obtained from each variation to the results obtained by using the original query.

Table 11.11: Number of exemplars retrieved for each variation of the query "q_thinwall_thick_less_0.5" in comparison to the original query

Query: q_thinwall_thick_less_0.5 (cosine angle ≥ 0.3)			
	Number of exemplars retrieved		**Number of exemplars retrieved**
(Original query U User 1)	16	(Original query ∩ User 1)	1
(Original query U User 2)	15	(Original Query ∩ User 2)	8
(Original query U User 3)	18	(Original query ∩ User 3)	4

For this query as well, it is seen that if user 1 is not considered, the number of common exemplars that are retrieved by all users increases from one to five. As well, users 2 and 3 retrieve more exemplars than the exemplars retrieved by user 1 and the original query. Figure 11.11 shows the semantic description for the query "q_thinwall_thick_less_0.5", written by user 2. The description contains seventy seven words, of which seventeen words are part of the controlled vocabulary. Figure 11.12 shows the semantic description written by user 3. The semantic description consists of hundred words, of which, nineteen words are part of the controlled vocabulary. As in all semantic descriptions, some of these words occur more than once.

*The intent of this exemplar is to **find thin walls** in a given model. The **query** identifies a solid **manifold** bounded by three **planes.** It is checked if any two of the three **planes** are **parallel** to each other. The **query** then returns the **distance** between these two **parallel planes**, which is the **thickness** of the **wall.** It is verified if the **thickness** is less than 0.5 units, in order to ensure it is a **thin wall.***

Figure 11.11: Semantic description of "q_thinwall_thick_less_0.5" written by user 2

*This exemplar may be used to identify **walls** with **thickness** less than 0.5. When a model is queried against this exemplar, it returns a **match** if the model has a **solid manifold** bound by three **planes**, represented by the **alpha match** portion of the exemplar. While at least two of these **planes** are expected to be **parallel**, represented by the **Parallel** (S1, S3) **relation** in the **alpha extract** portion of the exemplar , the **distance** between them should be less than 0.5 , represented by the **Equation** "eq1" (thickness < 0.5) **relation** in the **alpha extract** portion of the exemplar.*

Figure 11.12: Semantic description of "q_thinwall_thick_less_0.5" written by user 3

Figure 11.13 shows the semantic description written by user 1. The description has seventy words, of which only nine words are part of the controlled vocabulary.

*This exemplar is used as a beginning point for building thinwall **feature** recognition exemplars. First, three **planes** that are all bounding the same **solid** are found. Then, the **distance** between two of the **parallel planes** are extracted. Finally, the **distance** is checked against a thinwall threshold value of 0.5. If the **distance** is less than 0.5, then the exemplar holds to be true and a potential thinwall **feature** is found.*

Figure 11.13: Semantic description of "q_thinwall_thick_less_0.5" written by user 1

The semantic description written by user 1 shows that, although the user states the intent of the exemplar, the word "thinwall" is not part of the controlled vocabulary. As well, words in the description that are part of the controlled vocabulary carry less weight since all of the words are part of the design exemplar terminology and frequently occur across all semantic descriptions in the database. This may be the reason why user 1 does not retrieve any other thin wall exemplars.

From the results obtained for all the experiments described in this chapter, it is observed that, among all the users, the difference between the results obtained for the original query and the results obtained for the query written by "user 1" is significantly more for all four queries. One possible explanation for this trend may be the length of the semantic description. The reason why the length of the description may be a factor is that the probability of using the same words as another user decreases as the length of description decreases. In other words, the probability of the same words appearing in two

descriptions is higher as compared to two descriptions of significantly different lengths. One more possible explanation of the difference in results may be the expertise levels of the users. The expertise level of "user 1" is far greater than the other users in this experiment. Hence, the descriptions written by "user 1" may be more precise and concise as compared to the other users. Users 3 and 4 (Table 11.1), as well as the author who wrote the original semantic descriptions have relatively the same expertise level in design exemplar technology. User 2 is a novice in the use of design exemplar technology. This may explain as to why the results obtained from using the variations written by "user 2" and "user 3" as queries, are similar to the results obtained for the original query.

The results obtained from these experiments may be improved by modifying the manner in which a semantic description is represented in the form of a vector. Presently, the vector space model does not account for synonyms. Hence, if two users write semantic descriptions of the same exemplar using synonyms, the dot product of the vector representations of the two semantic descriptions would be zero. Hence, the cosine value of the angle between these vectors would be zero. Therefore, if synonyms are accounted for, the vector representations of the two semantic descriptions would be similar. As well, ontology can be developed in order to better represent the controlled vocabulary that has been developed. For example, "circle" is a type of "curve". Hence, ontology will assist in recognizing this kind of hierarchy and help in accurately representing a semantic description in the form of a vector.

11.2. Verifying the robustness of the structural similarity measures

In order to verify the robustness of the structural similarity measures, user 1 was asked to author variations of four structured exemplars, "2_planes_with_distance", "gear_pinion_02", "q_hole", and "q_thinwall_thick_less_0.5". These four variations are listed in Appendix F. Table 11.12 shows the number of entities and relations present in each query authored by user 1.

Table 11.12: Entities and relations of exemplars authored by user 1

Exemplar	Entities	Relations	Attributes
2_planes_with_distance_summers	2 planes, 1 solid manifold	1 boundary, 1 distance, 1 parallel, 3 id	4 alpha match, 5 alpha extract
gear_pinion_02_summers	2 circles	1 tangent, 2 radius, 2 id, 1 equation	2 alpha match, 6 alpha extract
q_hole_summers	1 solid manifold, 1 cylindrical surface, 2 circles, 2 planes, 2 vectors	4 boundary, 2 surface normal, 1 opposite direction	10 alpha match, 5 alpha extract
q_thinwall_thick_less_0.5_summers	1 solid manifold, 1 cylindrical surface, 3 planes, 2 lines, 2 vectors, 1 parameter	4 boundary, 2 surface normal, 1 opposite direction, 1 parallel, 1 distance, 1 equation, 3 id	9 alpha match, 12 alpha extract

Table 11.13 shows the exemplars retrieved for each query authored by "user 1" whereas Table 11.14 shows the number of exemplars retrieved by the original queries.

Table 11.13: Number Exemplars retrieved for each query authored by "user 1"

Query	Number of Retrieved Exemplars
2_planes_with_distance_summers	**5** (W2_1, W5, W6, W9, W9_1)
gear_pinion_02_summers	**7** (G1_2, F1, F3, F5, F6, S6, G5)
q_hole_summers	**1** (F4_1)
q_thinwall_thick_less_0.5_summers	**2** (W7, W9_1)

Table 11.14: Number Exemplars retrieved for each original query

Query	Number of Retrieved Exemplars
2_planes_with_distance	**14** (W1, W2, W2_1, W3, F1, F3, F5, F6, F7, W5, W7, W9, W9_1, W10)
gear_pinion_02	**12** (G1, G1_2, G2, F1, G3, F2, F5, F6, S4, S5, S6, G5)
q_hole	**6** (F1, F3, F4, F4_1, F5, F6)
q_thinwall_thick_less_0.5	**4** (F3, W7, W9, W9_1)

From these results, it can be seen that the number of exemplars retrieved by the original queries is more than the number of exemplars retrieved by user 1. As well, all queries originally present in the database retrieve the variations of exemplars that are authored by user 1. On comparing the queries it is found that the exemplars authored by

user 1 have more entities and relations. Hence, it may be inferred, that as the query exemplar becomes more complex, the number of exemplars getting retrieved may be less due to the structural similarity filters. In this experiment, target exemplars that did not have at least many entities of each type as the query exemplar were filtered. As well, the limits on the difference allowed in the number of relations and attributes of each type in a target exemplar and a query exemplar was set to be more than or equal to negative three. Therefore, one way to increase the number of exemplars retrieved by user 1 may be to make the structural filters less restrictive.

11.3. Summary

The experiments conducted to verify the robustness of the structural and semantic similarity measures were discussed in this chapter. The robustness of the semantic similarity measures was verified by evaluating the differences in the results of retrieval due to multiple users writing semantic descriptions of the same exemplar. From the results, it can be inferred that the number of words in the semantic description that are part of the controlled vocabulary may be an important factor while retrieving semantically similar exemplars. This can be seen from the results obtained for the queries "q_hole" and "q_thinwall_thick_less_0.5". As well, the expertise level of a user may influence the manner in which the semantic description of an exemplar is written. An expert exemplar author may write concise descriptions and identify potential uses of the same exemplar, which may influence the results of retrieval.

Similarly, the robustness of the structural similarity measures was verified by evaluating the differences in the results of retrieval due to different users authoring multiple structured exemplars meant for the same purpose. From the results, it can be seen that the structural similarity measures perform as expected. The complexity of the query exemplar with respect to the number of entities and relations may be an important factor while retrieving structurally similar exemplars. As the query exemplar becomes more complex, the structural filters may be made less restrictive, in order for more exemplars to get retrieved.

Chapter 12

CONCLUSIONS, CONTRIBUTIONS, AND FUTURE WORK

The objective of this research was to understand similarity with respect to graph based models in engineering design. The design exemplar was used as a case study to achieve this objective. Providing assistance to the exemplar author in authoring relatively complex exemplars served as a motivation for this research. Structural and semantic similarity measures between exemplars were defined in order to evaluate the similarity between exemplars. This research also helped in understanding the concept of semantics in engineering design which helped in identifying the different types of semantic information in engineering design. Of the different types of semantic information, design intent and rationale were identified as the types that need to be represented in order to evaluate the semantic similarity between exemplars. Having implemented the structural and semantic similarity measures, different experiments were conducted in order to develop an understanding of the idea of incorporating two views of similarity while retrieving similar exemplars. This chapter discusses the conclusions and observations of this research. It also discusses the contributions made by the outcomes of this research to other areas of engineering design.

12.1. Conclusions

The main objective of this research is to understand and explore the idea of incorporating different views of similarity to facilitate divergent search and retrieval of

design models. This objective is achieved by developing and evaluating different approaches to define similarity between design exemplars. The research questions that were answered as part of this research are discussed below.

RQ1: Can similarity measures be defined for graph-based models?

As discussed in chapter 3, the first question that this research tries to answer is whether or not similarity measures can be defined for exemplars and on what basis can exemplars be considered similar. Based on the similarity strategies and graph-matching concepts discussed in Chapter 4, exemplars can be considered structurally similar based on the types of common entities and relations present in both the query exemplar and a target exemplar and the way the entities are related to each other.

These similarity measures were implemented and experiments were conducted to verify their accuracy and effectiveness. The database of exemplars was manually parsed in order to verify that the structural similarity measures retrieved exemplars as expected. The results of these experiments are shown in Table 8.4. As well, the restrictiveness of the structural similarity measures were modified in order to evaluate as to which structural similarity metric was the most influential in the retrieval process. The results of these experiments are shown in Chapter 8. From the results it can be inferred that the structural similarity filters can be tuned according to the exemplar author's needs which influences the results of the exemplar retrieval process. For the database used in this research, it is suggested that the entity filter should be more conservative than the relation and the

attribute filter.

Since, there may exist different ways to author exemplars meant for the same purpose, structural similarity measures alone may not retrieve all exemplars that may of of interest to the exemplar author. Hence, exemplars can be considered semantically similar with respect to intent and rationale for authoring them in a specific manner. Chapter 9 discusses the different experiments that were conducted in order to study the usefulness of representing semantics. The results shown in

Table 9.2, Table 9.4, and Table 9.6 show that the semantic similarity measures retrieve exemplars that are not retrieved by the structural similarity measures.

Using the two views of similarity in conjunction to retrieve exemplars helps the exemplar author to discover different uses for an existing exemplar. It also helps the exemplar author in discovering different ways of authoring an exemplar meant for a specific purpose. This argument is supported by the results obtained by using the two similarity modules in conjunction. The two views of similarity can be conjoined by using the structural similarity measures and semantic similarity measures at the same time in parallel, or one after the other in series. Different experiments were conducted in order to explore the different ways of combining the structural and semantic views of similarity. The results obtained from the experiments discussed in Chapter 10 suggest that conjoining the two approaches of similarity is beneficial, since the number of exemplars retrieved may be more than the number of exemplars retrieved by either one approach individually. As well, for large databases, using the two similarity modules in

series may retrieve more exemplars than using them in parallel.

RQ2: What kind of semantic information is needed to define semantic similarity measures?

In order to answer this question, an understanding of the concept of semantics in engineering design needs to developed. In order to do so, different views of semantics in engineering design were studied. These views are presented in Chapter 5. Based on these different views, a definition of semantics in engineering design has been proposed as part of this research. Design semantics has been defined as what is understood from what is being said or represented. It may not be possible to represent "understanding" in any format. However an understanding of the design can be facilitated by representing design knowledge precisely. Different design documents were studied in order to classify the different types of knowledge found in engineering design that facilitate understanding of design. These different types of knowledge include information about a design product, and information about a design process. In order to understand an exemplar, it is necessary to know the intent of authoring the exemplar. Since there are multiple ways of authoring an exemplar for a specific purpose, it is also necessary to know the rationale for authoring the exemplar in a specific manner. Hence in order to retrieve exemplars that are semantically similar to a query exemplar, the intent and rationale for authoring the exemplar in a specific manner is represented.

RQ3: How can semantic information be associated with design exemplars?

Different ways to represent design knowledge have been described in chapter 6. The focus of this research is not to evaluate the best way to represent design knowledge. Rather the objective is to use a knowledge representation scheme in order to incorporate the semantic view of similarity. Hence, for purposes of retrieving semantically similar exemplars, the intent and rationale for authoring exemplars in a specific manner are represented textually. Text similarity measures are then used to evaluate the similarity between the query exemplar and target exemplars. First, the cosine value of the angle is between the vector representations of a target exemplar and a query exemplar is computed. Second, the edit distance measure is used to compute the number of changes required to convert a target exemplar into a query exemplar.

The weights assigned to the terms in the controlled vocabulary may influence the cosine value of the angle between the query vector and a target vector. The results of varying the weights of the terms in the controlled vocabulary are presented in Chapter 9. As observed from the results, when the terms pertaining to the design exemplar technology were assigned higher weights, the results of retrieval were closer to the exemplars retrieved by the structural similarity measures. The measures of semantic similarity are more effective when words that are relatively less common across all text descriptions are assigned more weights. Examples of such words include words describing features such as holes, pockets, and ribs. As well, the edit distance measure proves to be relatively less useful than the cosine angle measure in evaluating

the semantic similarity between exemplars.

12.2. Contributions

The outcomes of this research will help in contributing towards the areas of model reuse, semantics in engineering design, and similarity in engineering design. It will also contribute towards exemplar technology.

Exemplar Technology: As mentioned earlier, authoring exemplars for large design problems can be tedious and time consuming. The exemplar retrieval tool will provide assistance to the exemplar author in authoring complex exemplars by retrieving are structurally similar as well as semantically similar to the exemplar being authored. This will facilitate authoring exemplars that are precise and less time consuming. Exemplar authors can query using the exemplar that they have authored thus far or they can retrieve exemplars using text.

Engineering Model Reuse: The main contribution of this research is the dual approach of evaluating similarity. This approach can be used to evaluate the similarity between two designs by incorporating different views of similarity in the same similarity measure. For example, two products may have different functions but are manufactured in the same manner. These products may be considered similar with respect to their manufacturing processes. As well, two products may have similar functions but may be manufactured using different processes. These products may be considered similar with respect to the desired functionality. The dual approach proposed

in this research may be used to evaluate the overall similarity between these products incorporating both views of similarity.

The measures of similarity developed as part of this research can be used for graph-based representations in general. For example, function structures, process plans, etc. are usually represented graphically. The similarity metrics used for retrieving exemplars can be used to evaluate the similarities between function structures, process plans etc.

For example, a sprinkler manufacturing firm that designs different types of sprinklers may decide to design a different type of sprinkler. The manufacturing process followed while manufacturing the new sprinkler can be represented in terms of a graph. This graph can then be compared with the graph-based representations of manufacturing process plans of other sprinklers produced by the company. This will help in identifying a sprinkler such that least number of changes needs to be made to its process plan in order to manufacture the new sprinkler. Similarly function decomposition diagrams of two products can be compared in order to determine similar or common functions performed by both products. This may further help in generating more concepts, cost estimation, etc.

As mentioned in chapter 2, there exist different strategies for search and retrieval of CAD models. Some approaches use graphs as a descriptor of 3D-models and graph-matching algorithms are used in order to evaluate the similarity between the CAD models. The proposed algorithm for computing the structural similarity between graphs in this research can be extended to other graph-based representations

mentioned in literature.

For example, a new shape descriptor for 3D-models based on a dilation based skeleton (DBS) has been developed [109]. This descriptor is composed of a group of adjacent skeletal faces that captures the essence of a 3D-model and ignores the insignificant features. The DBS graph captures both the geometric and topological information of the 3D-model. The first step in generating a DBS graph is determination of dilation units. It is observed that the DBS of a solid model varies strongly depending upon the dilation units chosen. The proposed strategy involves extracting the dilation units and generating the initial DBS from the information got from the boundary representation and then executing a detection and refinement process to generate the final DBS. The second step is to voxelize the input CAD model. Voxelization involves converting each 3D-model into a set of voxels that represent the model. The third step is to perform dilation on the volumetric space of the 3D CAD model and thereby generate the DBS representation. The DBS is formed by putting together all the dilated skeletal faces (DSFs). A DSF is defined as a set of points in the dilation space, each of which has the same distance to two given dilation units. The next step is to generate a DBS graph from the DBS representation. Each node of the DBS graph represents one DSF surface. Each node has five attributes attached to it. These include “type”, “area”, “average distance”, and “average separation angle”. Figure 12.1 shows the four 3D-models along with the corresponding DBS graphs generated by the procedure described above.

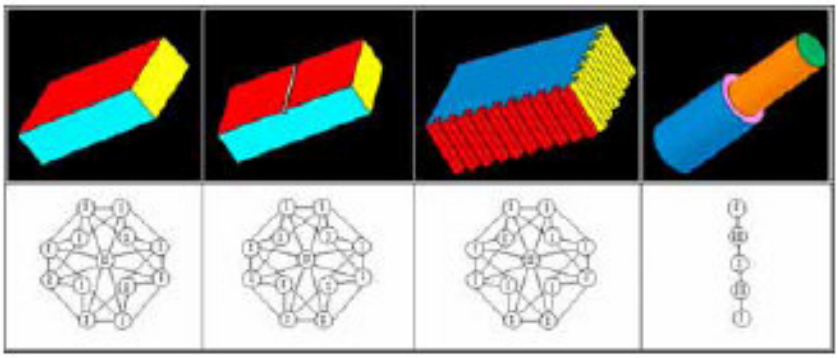


Figure 12.1: Four 3D CAD models and their DBS graphs [109]

The algorithm developed as part of this research can be used for this type of graph-matching. In this case also structural similarity can be evaluated by the use of filters. Instead of entity and attribute similarity filters, all graphs in the database that do not have nodes of the same type can be filtered. In the second step, all those graphs that have nodes of the same type but different attributes can be filtered. Having filtered graphs with respect to the type and attributes of filters, the structural similarity can be computed as explained in chapter 4.

A method for automatic generation of a manufacturing sequence for MEMS components using manufacturing rules stored in the form of graph-grammar has been suggested [110]. Figure 12.2 illustrates how the different layers of a MEMS device may be represented using a graph.

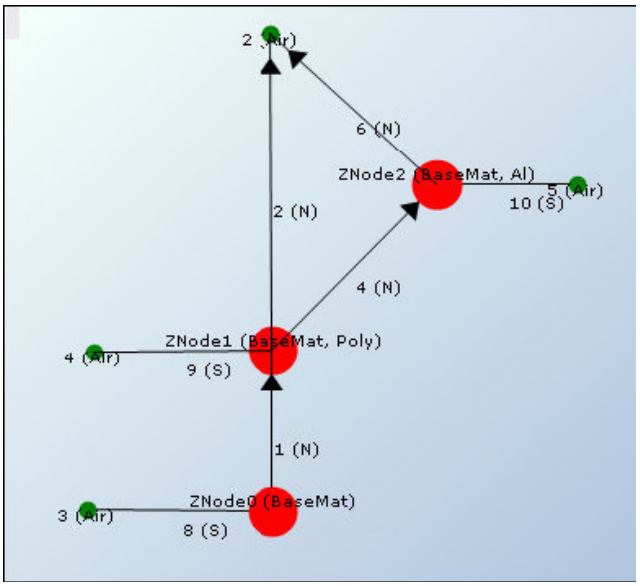


Figure 12.2: An example graph depicting layers within a MEMS device [110]

Given the final design of a MEMS component, the proposed tool will devise a fabrication sequence. The grammar rules that capture the constraints on the MEMS manufacturing process are applied to the final design in such a manner that the final component is a blank silicon wafer. Hence the rules are applied in reverse order since the starting point of MEMS manufacturing process is a blank silicon wafer. Since the output obtained is a graph, the measures of similarity proposed in this research can be used to determine which MEMS components are similar with respect to their manufacturing process.

Semantics in Engineering Design: The concept of semantics is not well understood in engineering design. This is clear from the different views of semantics that

exist in literature as presented in chapter 5. As part of this research a new definition of semantics in engineering design based on design knowledge has been synthesized. A classification scheme for the different types of design knowledge that can be considered semantic has been proposed. Different knowledge representation schemes in order to represent the different types of design knowledge have been surveyed. Design rationale and intended use have been identified as relevant types of semantic information for retrieving exemplars. Both design rationale and intended use are represented textually. Similarly, depending on the requirements, appropriate design knowledge may be represented using an appropriate knowledge representation scheme in order to incorporate semantics in design automation.

Similarity in Engineering Design: This research formed the motivation for exploring the area of similarity in engineering design. Different definitions of similarity for the various stages of design have been studied and compared. As mentioned earlier, a useful way to generate solutions to engineering design problems is to compare the solutions of design problems similar to the one at hand and validate the solutions to satisfy the new design requirements. This process involves evaluating the similarity between the design problem at hand and the various design problems in the repository. This research provided the motivation for exploring the area of similarity in engineering design.

Various theories of similarity have been proposed in literature. For example, the spatial model approach states that similarity data can be perceived as

proximity data. This implies that the more similar object A is to object B, the more proximate these objects are in the psychological space. It has been suggested that using the techniques of multi-dimensional scaling, a spatial representation can be developed, in which the distances between objects correspond as closely as possible to the similarity between objects [111, 112]. The vector-space approach suggests representing objects as vectors of features in multi-dimensional space. The similarity of objects is measured in terms of the Euclidean distance between them. The number of dimensions is based on the number of unique or distinguishing features of an object. Template models were developed primarily for the purpose of object recognition [113]. These models assume that the representations have much more detailed information of the object. Hence similarity between objects is measured as the degree of one-to-one comparisons of the representations being compared. The information theoretic approach suggests that the overall similarity between two objects is the average similarity computed based on different perspectives. The reader is referred to [114] for a detailed explanation of intuitions and assumptions made by this model. Based on the assumptions a similarity theorem is proposed, which is given by equation below.

$$\text{sim}(A,B) = \frac{\log P(\text{common}(A,B))}{\log P(\text{description}(A,B))}$$

The various similarity models and theories discussed above can be applied to different stages of engineering design. For example, a complete definition of the design problem may involve development of the product design specification, which lists the

basic purpose of the product along with the constraints and criteria. In that case, the product design specifications of different products can be compared in order to evaluate the similarity of the two design problems. The vector space approach could be applied to find the similarities between specifications of designs. During functional decomposition, the overall function of the design is broken down into various sub-functions and usually represented in the form of a graph or a tree. Hence to evaluate the similarity between function structures, the edit distance approach can be employed.

In order to generate concepts for a design, the designer may start by searching for similar concepts already existing in a database. Edit distance approach can be used in finding similar concepts, provided the concepts are represented in the form of a graph or tree, such as a working structure [12]. A process-plan based similarity explicitly states that two designs are similar if and only if their process plans are similar [115]. There are several similarity measures that are used to classify or group designs together with respect to the manufacturing information associated with them [116]. Some of the measures are discussed below. Group Technology (GT) is one of the oldest ways to classify designs together [116]. The idea is to make parts of similar shape in specially grouped machines. This in turn presents a need to classify designs and products as per their manufacturing requirements. Usually GT code consists of two positions. In one case, the position could be a global property of the design such as size, weight, color etc. These properties are independent of the values in these positions. In another case, a position represents some characteristics particular to a design. However, it is argued that

using GT code is a crude way to classify designs. As well, GT code was meant to be interpreted by humans, which creates difficulties in automating the code generation process.

Case-based Reasoning: The search and retrieval method discussed in this dissertation may be useful in discovering new uses for the retrieved exemplars. This approach may be considered as an extension to case-based reasoning. Case-based reasoning is referred to the technique of adapting solutions to previous design problems that are similar to the design problem at hand to satisfy the new design requirements. The dual approach of search and retrieval can be used to find solutions that are similar to the retrieved solutions. Thus, the retrieved solutions can be adapted to satisfy the requirements of the corresponding problems of the similar solutions.

12.3. Future Work

The main objective of this research is to understand and explore the idea of incorporating both the structural and semantic views of similarity to facilitate divergent search and retrieval of design models. This dual approach can be easily extended for incorporating more than two views of similarity between graph-based models. It can also serve as a testbed for conducting future research in analogy based design and case based reasoning.

In order to retrieve semantically similar exemplars, a textual representation was used to represent the intent and rationale for authoring an exemplar due to ease of

implementation. However, the other knowledge representation schemes described in Chapter 6 may be equally effective in evaluating the semantic similarity between exemplars. These representation schemes should be evaluated against the requirements of knowledge representation for exemplar retrieval in order to conclude which representation scheme is the best suited for purposes of exemplar retrieval.

The experiments described in Chapter 11 show that there is variation in the results obtained for multiple semantic descriptions of the same exemplar. However there are multiple ways of modifying the manner in which the semantic descriptions of exemplars can be written in order to obtain more overlap in the results obtained for multiple variations of the same query exemplar. One way of doing so, is to account for synonyms and hierarchies of words. For example, “circle” is a type of “curve”. Hence, partial credit can be given if the word “circle” is used in place of the word “curve”. As well, a constraint could be imposed on the percentage of words in a description that are part of the controlled vocabulary. For example, constraint could be imposed such that at least 40% of the words in the description should be part of the controlled vocabulary. One more way of modifying the representation could be to have the user answer a set of questions. The answers to the questions could be used to formulate the intent and rationale of the exemplar. As well, while writing the intent and rationale, the user can be asked to select words from a pull down menu while writing the semantic description. This could be one way of enforcing the use of words that are part of the controlled vocabulary.

As mentioned earlier, this research formed the motivation for exploring the area of similarity in engineering design. There exist different views of similarity in engineering design. A clear understanding of similarity in engineering design should be developed. This will help in developing measures of similarity that are more accurate, more effective, and more efficient than the measures of similarity proposed as part of this research.

The focus of conducting different experiments was to verify the accuracy and effectiveness of the proposed similarity measures. However, if the size of the database is relatively large then it is equally important to retrieve exemplars efficiently. A trade off may be needed between accuracy and speed in cases where the size of the database is sufficiently large.

APPENDICES

Appendix A contains the list of all exemplars in the database. Appendix B shows the controlled vocabulary used to write semantic descriptions of exemplars. Appendices C, D, and E show the results of using the structural and semantic similarity measures in parallel and series. For all the experiments, the exemplars "2_planes_with_distance", "gear_pinion_02", "q_hole", and "q_thinwall_thick_less_0.5" were used as queries. Appendices F, G, H, I, and J show the structured and semantic variations authored by different users.

Appendix A: List of Exemplars in database

Appendix A contains all the text representations of all structural exemplars in the database. As well, each structural exemplar is followed by the semantic description of the exemplar. The exemplars are listed in the same order as listed in Table 8.1.

Alpha Match:
Plane "S1";
Plane "S2";
Line "C1";
Line "C2";
Alpha Extract:
Parameter "d";
Distance **({C1, C2}, d);**
Parallel **(C1, C2);**
Incident **(C1, S1);**
Incident **(C2, S2)\;**
Id **(d);**
Id **(C1);**
Id **(C2);**

W1: 2_lines_dist_2_planes.stp

The intent of this exemplar is to match two planes in the model and find the distance between 2 lines incident on these planes. The exemplar consists of two planes and two lines that have the match attribute since they are explicitly present in the model. There exists a distance and a parallel relation between the two lines. These relations extract the distance between the two lines and store it in the parameter d. The parameter d and the distance and parallel relations have the attribute extract. Each line is related to one plane by an incident relation which is extract.

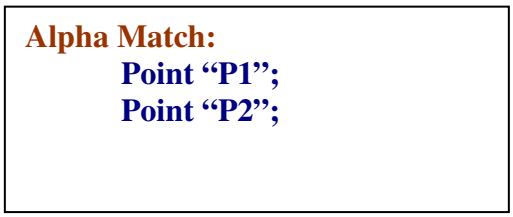
W2: 2_lines_dist_2_planes.des

Alpha Match:
Plane "S1";
Plane "S2";
Alpha Extract:
Parameter "dist";
***Id* (dist);**
***Id* (S1);**
***Id* (S2);**
***Distance* ({S1, S2}, dist);**
***Parallel* (S1, S2);**

W2: 2_planes_with_distance.stp

The intent of this exemplar is to find the distance between two planes. The exemplar consists of two planes with a parallel and a distance relation between the planes that have the attribute extract since they are not explicitly present in the model. The parallel relation ensures that the two planes are parallel to each other and the distance relation extracts the distance between the two planes. When the exemplar is applied to the model, the distance between the planes is displayed as well as the planes are highlighted.

W2: 2_planes_with_distance.des



S1: 2_points.stp

The intent of this exemplar is to find two points in a model. The exemplar consists of two points that have the match attribute.

S1: 2_points.des

Alpha Match:
Line “C1”;
Line “C2”;
Point “P1”;
Point “P2”;
Alpha Extract:
Parameter “d”;
Id (d);
Distance ({C1, C2}, d);
Parallel (C1, C2);

S2: 2_points_2_lines_with_dist.stp

The intent of this exemplar is to find the distance between two lines and also match two points in the model. The exemplar consists of two lines that have the match attribute with a parallel and a distance relation between them. The distance relation extracts the value of the distance between the lines and stores it in the distance parameter. The value of the distance is displayed because of the id relation attached to the parameter.

S2: 2_points_2_lines_with_dist.des

Alpha Match:
Point "P1";
Point "P2";

Alpha Extract:
Distance **({P1, P2}, d1);**
Parameter (d1);
Id **(d1);**

S3: 2_points_with_distance.stp

The intent of this exemplar is to find the distance between two points. The two points have the attribute match since they are explicit in the model. The two points are related with a distance relation that extracts the distance between them. The value of the distance is stored in the parameter d1 which is displayed on application of the exemplar to a model.

S3: 2_points_with_distance.des

Alpha Match:
Circle "C1"
Circle "C2"
Alpha Extract:
***Radius* ({C1}, gear);**
***Radius* ({C2}, pinion);**
Parameter "gear";
Parameter "pinion";
***Equation* "eq1" (gear > pinion);**

G1: gear_pinion_02.stp

This exemplar is intended to find a pair of gears in the model and determine which the pinion is and which the gear is. The exemplar consists of 2 circles that have the attribute match. One circle represents the gear and the other circle represents the pinion. The equation relation is applied to the radii parameters to check which the smaller circle is. The smaller circle is the pinion and the bigger circle is the gear.

G1: gear_pinion_02.des

Alpha Match:

Circle “C1”
Circle “C2”
Circle “C3”
Circle “C4”
Circle “C5”

Alpha Extract:

Parameter “r1”;
Parameter “r2”;
Parameter “r3”;
Parameter “r4”;
Parameter “r5”
Parameter “distance”
Parameter “distance2”
Parameter “distance3”
Parameter “ratio”
Radius ({C1}, r1);
Radius ({C2}, r2);
Radius ({C3}, r3);
Radius ({C4}, r4);
Radius ({C5}, r5);
Equation “eq1” (ratio = r1/r2 * r4/r5);
Equation “eq2” (distance = r1 + r2);
Equation “eq3” (distance2 = r4 + r5);
Equation “eq4” (distance3 = 2 * r3);
Equation “eq4” (distance + distance3 = distance2);

G2: gears_double_ratio.stp

This exemplar is about finding the gear ratio of a gear train consisting of 5 gears. One of the four gears is an idler gear. The exemplar consists of 5 circles that represent the gears. Radius relations are applied to each circle and their values are stored in parameters r1, r2, r4 and r5. The center to center distance between gears 1, 2 and the gears 4, 5 is calculated. The reduction ratio is calculated as a product of the ratio r1 to r2 and ratio r4 and r5. An equation is applied to distance and distance2 such that distance2 is equal to the sum of distance and diameter of the idler gear. ID relations are applied to each circle and the parameters in order to display the parameters and highlight the circles upon a successful match.

G2: gears_double_ratio.des

Alpha Match:

Line “C1”;
Line “C2”;
Plane “S1”;
Plane “S2”;
Point “P1”;
Point “P2”;

Alpha Extract:

Line “C3”;
Parameter “d1”;
***Incident* (S1, C1);**
***Incident* (S2, C2);**
***Incident* (C1, P1);**
***Incident* (C2, P2);**
***Incident* (C3, P1);**
***Incident* (C3, P2);**
***Right Angle* (C3, S2);**
***Id* (d1);**
***Distance* ({P1, P2}, d1);**
***Parallel* (C1, C2);**

W3: planes_lines_points_distance.stp

The intent of this exemplar is to find the distance between two planes. The distance relation applied between the two points extracts the distance between the two planes since the points are incident on one line each which in turn are incident on one plane each. The two points are connected through a line that has the extract attribute. This line is made perpendicular to one plane in order to ensure that the planes are parallel to each other.

W3: planes_lines_points_distance.des

Alpha Match:

Line "C1";
Line "C2";

Alpha Extract:

Parameter "distance";
Parameter "input";
Parameter "output";
Parameter "ratio";
Distance **(C1, C2);**
Parallel **(C2, C2);**
Equation "eq1" **(output < distance/2);**
Equation "eq2" **(distance/2 < 3 *(output + input);**
Equation "eq3" **(output/input = ratio);**

B1:q_belt_radii_.stp

The intent of this exemplar is to size a transmission belt. The exemplar consists of two lines which represent the shafts of a belt system. The distance between the centers is calculated using the distance relation between the two lines. Id relations are used to identify the input shaft and the output shaft. The exemplar has a parameter which has a fixed value. Three equations are used to fully constrain the belt size.

B1: q_belt_radii.des

Alpha Match:

Cylindrical surface "S1"
Plane "S2";
Plane "S3";
Circle "C1";
Circle "C2";
Boundary (S1 {C1, C2});
Boundary (S3, C2);
Boundary (S3, C1);

Alpha Extract:

***Distance* (S2, S3);**
***Parallel* (S2, S3)**
***Radius* (C1);**
Parameter "hght_of_cylinder";
Parameter "radius";
***Equation* "eq1" (radius < 1);**

F1: q_boss_radius.stp

The intent of this exemplar is to find a cylindrical boss in a model such that its radius is less than one unit and also determine its height. The cylindrical surface that is bound by two circles and the two planes, each bound by one of the circles form the structure of the boss. A radius relation is applied to one circle and an equation applied to the radius parameter checks whether its value is less than one unit. As well, a distance relation is applied between the planes to determine the height of the boss.

F1: q_boss_radius.des

Alpha Match:

Circle "C1"
Circle "C2";
Circle "C3";
Circle "C5";
Circle "C5";

Alpha Extract:

Parameter "R1";
Parameter "R2";
Parameter "R3";
Parameter "R4";
Parameter "R5";
Parameter "ratio";
Radius **({C1}, R1);**
Radius **({C2}, R2);**
Radius **({C3}, R3);**
Radius **({C4}, R4);**
Radius **({C5}, R5);**
Id (R1);
Id (R2);
Id (R3);
Id (R4);
Id (R5);
Equation **"eq1" (ratio = R1 / R3);**
Equation **"eq2" (R4 = R5);**
Equation **"eq3" (R1 + R3 + 2*R2 = R4 + R5);**

G3: q_compound_5_gears.stp

The intent of this exemplar is to match a compound gear train such that they satisfy a set of conditions. The exemplar consists of five circles that represent the gears. A set of equations are applied to the parameters in order to verify whether the gears satisfy certain conditions. The first equation evaluates the ratio of the radii of two gears. The second equation checks whether the two other gears have the same radii. The final equation ensures that the sum of the radii of the two gears of the same size is equal to sum of the radii of two other gears and twice the radii of the last gear.

G3: q_compound_5_gears.des

Alpha Match:

Cylindrical Surface "S1"
Cylindrical Surface "S3"
Circle "C1"
Circle "C2";
Circle "C3";
Circle "C5";
Circle "C5";

Alpha Extract:

Plane "S3";
Parameter "thickness" (thickness = 1);
Parameter "r_big";
Parameter "r_small";
Coincident (S1, S2);
Parallel (S1, S2);
Incident (S1, C1);
Incident (S2, C2);
Incident (S3, C1);
Incident (S3, C2);
Radius ({C1}, r_big);
Radius ({C2}, r_small);
Equation "eq1" (r_big > r_small);
Equation "eq2" (r_big-r_small > thickness);

F2: q_connecting_rod_thick_enough.stp

The intent of this exemplar is to determine if the connecting rod is thick enough. The exemplar consists of two cylinders that are explicit representing the inner and outer walls of the connecting rod. A radius relation is applied to each cylindrical surface and the values are stored in parameters r1 and r2. The exemplar consists of a numerical parameter named thickness which stores the minimum value of the desired thickness. The thickness of the connecting rod is determined by subtracting the radius of the outer cylindrical surface from the inner cylindrical surface. An equation is used to check whether the difference between the radii is more than the parameter thickness.

F2: q_connecting_rod_thick_enough.des

Alpha Match:
Circle "C1"
Circle "C2"
Alpha Extract:
Plane "S1"
***Coincident* (C1, S1);**
***Coincident* (C2, S1);**
***Id* (C1);**
***Id* (C2);**

G4: q_coplanar_gears.stp

The intent of this exemplar is to find gears that are coplanar. The exemplar consists of two circles that are coincident with a plane. The circles represent the two gears. The coincident relations help to ensure that the two gears are coplanar. The plane and the coincident relations have the attribute extract.

G4: q_coplanar_gears.des

Alpha Match:

Cylindrical surface "S1"
Plane "S2";
Plane "S3";
Circle "C1";
Circle "C2";
Boundary (S1 {C1, C2});
Boundary (S3, C2);
Boundary (S3, C1);

Alpha Extract:

***Distance* (S2, S3);**
***Parallel* (S2, S3)**
***Radius* (C1);**
Parameter "hght_of_cylinder";
Parameter "radius";

F3: q_cylinder.stp

The intent of this exemplar is to find a cylinder in a model and determine its radius and height. The cylindrical surface bound by two circles and two planes, each bound by one of the circles form the structure of the cylinder. A radius relation applied to one circle determines the radius of the cylinder. As well, a distance relation is applied between the planes to determine the height of the cylinder.

F3: q_cylinder.des

Alpha Match:

Solid_Manifold_Brep "R1"
Plane "S1";
Plane "S2";
Boundary (R1 {S1, S2});

Alpha Extract:

***Distance* (S1, S2);**
***Parallel* (S1, S2)**
Parameter "distance";
Id (distance);

W4: q_dist_bounded_planes.stp

The intent of this exemplar is to find the distance between two planes of a manifold. The exemplar consists of a solid manifold that is bound by two planes. A distance and a parallel relation is applied between the two planes in order to determine the distance between these two planes. The distance and parallel relations are extract. An id relation is applied to the distance parameter in order to display the value.

W4: q_dist_bounded_planes.des

Alpha Beta Match:
Circle "C1";

Alpha Beta Extract:
Parameter "temp";

Alpha Extract:
Parameter "r_a";
***Radius* (C1);**
***Equation* "eq_a" (temp = r_a*2);**

Beta Extract:
Parameter "r_b"
***Radius* (C1);**
***Equation* "eq_b" (r_b = temp);**

M1: q_double_radius.stp

The intent of this exemplar is to modify a circle by doubling its radius. The exemplar consists of a circle that has the attribute alphabeta match because it is present in the model both before and after modification. It consists of a radius relation and a radius parameter that stores the value of the radius before modification. This value is store in a numeric parameter temp that had the attribute aplhabeta extract since it is implicit in the model both before and after modification. The beta state consists of another radius relation and radius parameter applied to the circle. An equation relation is applied between the second parameter and temp that doubles the value stored in temp and applies it to the model.

M1: q_double_radius.des

Alpha Match:

Cylindrical Surface "S1";
Plane "S2";
Plane "S3";
Circle "C1";
Circle "C2";
***Boundary* (S1 {C1, C2});**
***Boundary* (S3, C2);**
***Boundary* (S3, C1);**

Alpha Extract:

***Id* (S1);**

F4: q_hole.stp

The intent of this exemplar is to find holes in a model. The exemplar consists of a cylindrical surface that is bound by two circles and two planes, each bound by one of the circles. These entities form the structure of the hole. An id relation is applied to the cylindrical surface. When this exemplar is applied to any model all cylindrical holes in the model will get highlighted.

F4: q_hole.des

Alpha Match:

Cylindrical surface "S1"
Plane "S2";
Plane "S3";
Circle "C1";
Circle "C2";
***Boundary* (S1 {C1, C2});**
***Boundary* (S2, C2);**
***Boundary* (S3, C1);**

Alpha Extract:

Line "C3";
Point "P3";
Point "P4";
***Incident* (P3, S2);**
***Incident* (P4, S3);**
***Coincident* (S1, C3);**
***Boundary* (C3 {P3, P4});**
***Distance* ({P3, P4}, depth);**
***Radius* ({C1}, radius);**
Parameter "depth";
Parameter "radius";

F5: q_hole_depth_radius.stp

The intent of this exemplar is to find holes in a model and determine its radius and depth. The exemplar consists of a cylindrical surface that is bound by two circles and two planes, each bound by one of the circles. These entities form the structure of the hole. A radius relation is applied to one circle gives the radius of the hole. The exemplar also consists of a line that is bound by two points such that the two points are incident on the two planes and the line is coincident with the cylindrical surface. This line forms the center line of the hole A distance relation is applied between the two points since that would give the depth of the hole.

F5: q_hole_depth_radius.des

Alpha Match:

Cylindrical surface "S1"
Plane "S2";
Plane "S3";
Circle "C1";
Circle "C2";
***Boundary* (S1 {C1, C2});**
***Boundary* (S2, C2);**
***Boundary* (S3, C1);**

Alpha Extract:

***Distance* ({S2, S3}, depth);**
***Radius* ({C1}, radius);**
Parameter "depth";
Parameter "radius";

F6: q_hole_radius_depth.stp

The intent of this exemplar is to find holes in a model and determine its radius and depth. The exemplar consists of a cylindrical surface that is bound by two circles. The exemplar also consists of two planes, each bound by one of the circles. These entities form the structure of the hole. A radius relation is applied to one circle in order to determine the radius of the hole. A distance relation is applied between the two planes since that would give the depth of the hole.

F6: q_hole_radius_depth.des

Alpha Match:
Circle "C1"
Circle "C2";
Alpha Extract:
Parameter "r1";
Parameter "r2";
Parameter "ratio";
***Radius* ({C1}, r1);**
***Radius* ({C2}, r2);**
***Id* (r1);**
***Id* (r2);**
***Id* (ratio);**
***Equation* "eq1" (r1/r2 = ratio);**

S4: q_radii_ratio.stp

The intent of this exemplar is to find the ratio of the radii of two circles. The exemplar consists of two circles that are explicitly present in the model. A radius relation is attached to each circle. The value of the radius of each circle is stored in a parameter each. An equation relation is applied to both parameters which determines the ratio of the two radii and stores the value in another parameter. Id relations attached to all parameters help display the value of the radius of each circle and the value of the ratio.

S4: q_radii_ratio.des

Alpha Match:
Circle "C1"
Circle "C2";
Alpha Extract:
Parameter "r1";
Parameter "r2";
Parameter "ratio";
Radius ({C1}, r1);
Radius ({C2}, r2);
Id (r1);
Id (r2);
Id (ratio);
Equation "eq1" (r1/r2 = ratio);
Equation "eq2" (r1 > r2);

S5: q_radii_ratio_w_large_check.stp

The intent of this exemplar is to find the ratio of the radii of two circles and determine the larger circle. A radius relation is attached to each circle in order to determine its radius. An equation relation is applied to each radius parameter which determines the value of the ratio of the two radii and stores the value in another parameter. Another equation applied to the two radius parameters helps determine the larger circle. Id relations attached to all parameters help display the value of the radius of each circle and the value of the ratio.

S5: q_radii_ratio_w_large_check.des

Alpha Match:
Cylindrical Surface "S1";
Plane "S2";
Plane "S3";
Circle "C1"
Circle "C2";
Alpha Extract:
Parameter "rad";
***Incident* (C1, S1);**
***Incident* (C1, S2);**
***Incident* (C2, S1);**
***Incident* (C2, S3);**
***Radius* ({S1}, rad);**
***Id* (rad);**

F7: q_radius_cylindrical_hole.stp

The intent of this exemplar is to find the radius of a hole. The exemplar consists of a cylindrical surface and 2 plane surfaces. The exemplar also consists of 2 edge curves and a line. The 2 plane surfaces are incident on one curve each while the curves are incident on the cylindrical surface. Thus the curves and the surfaces together form the structure of a hole. A radius relation applied to the cylindrical surface gives the radius of the hole.

F7: q_radius_cylindrical_hole.des

Alpha Match:

Circle "C1"
Circle "C2";

Alpha Extract:

Parameter "rad1";
Parameter "rad2";
***Radius* ({C1}, rad1);**
***Radius* ({C2}, rad2);**
***Tangent* (C1, C2);**
***Id* (C1);**
***Id* (C2);**
Equation "eq1" (rad1 > rad2)

S6: q_tangent_circles.stp

The intent of this exemplar is to find circles that are tangent to each other. A tangent relation applied to both the circles helps determine whether the two circles are tangent to each other. The tangent relation has the extract attribute.

S6: q_tangent_circles.des

Alpha Match:

Solid_Manifold_Brep "R1"
Plane "S1";
Plane "S2";
Plane "S3";
Boundary (R1 {S1, S2, S3});

Alpha Extract:

Vector *"direction1";*
Vector *"direction2";*
Parameter *"thickness"*;
Parameter *"angle"*;
TC_Normal_Vector (Body, S1, direction1);
TC_Normal_Vector (Body, S2, direction2);
Angle (direction1, direction2);
***Distance* (S1, S2);**
***Parallel* (S1, S2)**
***Equation* "eq1" (thickness > 0.5);**
***Equation* "eq2" (angle > -0.5);**
***Equation* "eq3" (angle < 0.5);**

W5: q_thinwall_medium_with_boundary.stp

The intent of this exemplar is to find thin walls. The exemplar consists of a solid manifold bound by three planes. These planes form the structure of a wall. The equation relations applied to the angle parameter between the two surface normals are used to ensure that the two planes are on the same side of the third plane. A distance relation is applied between the same two planes in order to determine the thickness of the wall. In this case a wall considered as thin as long as the thickness is less than 0.5 units. An equation applied to the thickness parameter checks whether this condition is satisfied.

W5: q_thinwall_medium_with_boundary.des

Alpha Match:

Plane "S1";

Plane "S2";

Alpha Extract:

***Parallel* (S1, S2);**

W6: q_thinwall_parallel_planes.stp

The intent of this exemplar is to find thin walls in a model. The exemplar consists of two planes representing the two sides of the wall with a parallel relation between them. Id relations are applied to the planes so that on application of the exemplar to a model, the thinwalls in the model will be highlighted.

W6: q_thinwall_parallel_planes.des

Alpha Match:

Solid_Manifold_Brep "R1"
Plane "S1";
Plane "S2";
Plane "S3";
Boundary (R1 {S1, S2, S3});

Alpha Extract:

Vector *"direction1";*
Vector *"direction2";*
Parameter *"thickness"*;
Parameter *"limit"*;
***Fixed* (limit= 0.5);**
TC_Normal_Vector (Body, S1, direction1);
TC_Normal_Vector (Body, S2, direction2);
Opposite_Direction (direction1, direction2);
***Distance* (S1, S2);**
***Parallel* (S1, S2)**
***Equation* "eq1" (thickness < limit);**
***Equation* "eq2" (thickness > 0);**

W7: q_thinwall_medium_with_boundary.stp

The intent of this exemplar is to find thin walls. The exemplar consists of a solid manifold bound by three planes. These planes form the structure of a wall. Two surface normal relations are applied to two planes and have the extract attribute. As well, an opposite direction relation is applied between the surface normals. This ensures that both sides of the wall are on the opposite sides of the third plane A distance relation is applied between the same two planes in order to determine the thickness of the wall. In this case a wall is considered to be thin if its thickness is less than 0.5 units. Two equations are applied to the distance parameter to check whether this condition is satisfied.

W7: q_thinwall_medium_with_boundary.des

Alpha Match:

Solid_Manifold_Brep "R1"
Plane "S1";
Plane "S2";
Plane "S3";
Boundary (R1 {S1, S2, S3});

Alpha Extract:

***Parallel* (S1, S2)**

W8: q_thinwall_solid_bound_parallel_planes.stp

The intent of this exemplar is to find thin walls. The exemplar consists of a solid manifold bound by three planes. These planes form the structure of a wall. 2 planes are made parallel to each other representing two sides of the wall. All entities have the attribute match and the parallel relation have the extract attribute.

W8: q_thinwall_solid_bound_parallel_planes.des

Alpha Match:

Solid_Manifold_Brep "R1"
Plane "S1";
Plane "S2";
Plane "S3";
Boundary (R1 {S1, S2, S3});

Alpha Extract:

Parameter *"thickness";*
***Parallel* (S1, S2)**
***Distance* (S1, S2);**
***Equation* "eq1" (thickness < 0.5);**

W9: q_thinwall_solid_bound_parallel_planes.stp

The intent of this exemplar is to find thin walls. The exemplar consists of a solid manifold bound by three planes. These planes form the structure of a wall. A distance relation is applied between two planes in order to determine the thickness of the wall. An equation is used to evaluate whether the thickness is less than 0.5 units.

W9: q_thinwall_solid_bound_parallel_planes.des

Alpha Match:

Solid_Manifold_Brep "R1"
Plane "S1";
Plane "S2";
Plane "S3";
Boundary (R1 {S1, S2, S3});

Alpha Extract:

Vector *"direction1";*
Vector *"direction2";*
Parameter *"thickness"*;
Parameter *"limit"*;
***Fixed* (limit= 0.5);**
TC_Normal_Vector (Body, S1, direction1);
TC_Normal_Vector (Body, S2, direction2);
Opposite_Direction (direction1, direction2);
***Distance* (S1, S2);**
***Parallel* (S1, S2)**
***Equation* "eq1" (thickness < limit);**
***Equation* "eq2" (thickness > 0);**

W10: q_thinwall_medium_with_boundary.stp

The intent of this exemplar is to find thin walls. The exemplar consists of a solid manifold bound by three planes. These planes form the structure of a wall. Two surface normal relations are applied to two planes and have the extract attribute. As well, a same direction relation is applied between the surface normals. This ensures that both sides of the wall are on the same sides of the third plane A distance relation is applied between the same two planes in order to determine the thickness of the wall. In this case a wall is considered to be thin if its thickness is less than 0.5 units. Two equations are applied to the distance parameter to check whether this condition is satisfied.

W10: q_thinwall_medium_with_boundary.des

Alpha Match:

Circle "C1"
Circle "C2";

Alpha Extract:

Parameter "gear";
Parameter "pinion";
Plane "S1";
Radius **({C1}, gear);**
Radius **({C2}, pinion);**
Tangent **(C1, C2);**
Incident **(C1, S1);**
Incident **(C2, S1);**
Equation "eq1" (gear > pinion)

G5: spur_gears_02.stp

The intent of this exemplar is to find a pair of spur gears in a model. The exemplar consists of two circles that have the attribute match and represent the two gears. An equation is applied to the radius parameters to ensure that the pinion is the smaller of the two gears. The circles are made tangent to each other by applying the tangent relation. Both the circles are made incident on a plane and hence are coplanar. When this exemplar is applied to a model, the pinion and gear are highlighted and their radii are displayed.

G5: spur_gears_02.des

Appendix B: Controlled Vocabulary

boss	Sweep	determine
bosses	Symmetric	determines
Chamfer	symmetry	Fix
concave	taper	Highlight
convex	Thread	highlights
Coradial	Tjunction	incident
Datum	undercut	Logical
depression	web	connective
depressions	webs	connectives
Dimple	alpha	Match
dimples	Angle	matches
structure	Area	modifies
draft	beta	Modify
Fillet	Change	block
Flange	changes	blocks
flat	check	Parallel
grill	checks	common
gusset	Circle	side
gussets	circles	sides
wall	Coincident	front
Hole	equation	Pattern

holes	equations	Perpendicular
Neck	Concentric	plane
Pattern	Constraint	planes
patterns	constrain	line
Pipe	curve	lines
pipes	curves	Present
rib	surface	Query
ribs	surfaces	queries
round	Cylinder	Relation
Scaling	cylinders	relations
Shaft	Delete	attribute
shafts	deletes	attributes
corner	edge	relates
corners	edges	Remove
sharp	Entity	removes
shell	entities	graph
Sketch	explicit	graphs
Slot	explicitly	Normal
slots	Expose	normals
Spiral	extract	Tangent
step	extracts	transformation
feature	Find	transform

features	finds	validate
bending	elastic	safety
bend	fatigue	factor
brittle	fit	shear
buckling	fracture	spalling
validates	cams	revolves
validation	Camshaft	Rotation
verify	chain	rotate
verification	chains	Topology
verifies	coupling	Translation
vertex	Crankshaft	translate
vertices	fastener	translates
Volume	Gear	boring
Horizontal	gears	bore
Intersect	Hopper	broaching
intersection	key	drilling
intersects	keys	milling
midpoint	Piston	punching
Parameter	vessel	punch
parameters	roller	sawing
Radial	screw	saw
angle	screws	tapping

angles	spline	tap
area	Valve	thrust
circumference	valves	turning
height	head	turn
length	bushing	shrinkage
distance	jaws	shrink
perimeter	chucks	core
radius	drillbit	time
radii	fixture	drag
slenderness	flute	gate
slender	form	mold
thickness	forming	cavity
thick	jig	closure
thin	mill	parting
volume	shank	pattern
width	shanks	riser
wide	spindle	cope
Pocket	threading	core
pockets	thread	cure
bearing	Bezier	ejector
belt	Bspline	gate
belts	Extrude	pin

blade	extrusion	pins
blades	extrudes	runner
brake	Hermite	sprue
brakes	Loop	thermoplastic
Cam	Revolve	thermoset
torsion	buckle	spall
wear	corrosion	strain
yield	creep	stress
fretting	plastic	rupture

Appendix C: Using structural and semantic similarity measures in parallel

TableC1 lists the names of exemplars that are retrieved for the query, "2_planes_with_distance". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.3, and the level of structural similarity is more than or equal to 0.5. The number in parentheses besides each exemplar name in the list of semantically similar exemplars represents the value of the cosine angle between that exemplar and the query exemplar. . The number in parentheses besides each exemplar name in the list of structurally similar exemplars represents the measure of structural similarity between that exemplar and the query exemplar. . The cells marked in yellow represent exemplars retrieved by both modules.

Table C1: Using structural and similarity modules in parallel for "2_planes_with_distance"

2_planes_with_distance (cosine angle ≥ 0.3, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
2_lines_dist_2_planes (0.678)	2_lines_dist_2_planes (0.75)
2_planes_with_distance (1)	2_planes_with_distance (1)
2_points_2_lines_with_dist (0.5)	q_boss_radius (1)
2_ponts_with_distance (0.52)	q_cylinder (1)
planes_lines_points_distance (0.66)	q_hole_depth_radius (0.75)
q_dist_bounded_planes (0.94)	q_hole_radius_depth (1)
q_thinwall_parallel_planes (0.42)	q_radius_cylindrical_hole (0.75)
q_thinwall_solid_bounded_planes (0.34)	q_thinwall_simple_boundary (1)
q_thinwall_thick_less_0.5 (0.3)	q_thinwall_thick_less_0.5 (1)
	q_thinwall_medium_with_boundary (1)

TableC2 lists the names of exemplars that are retrieved for the query, "2_planes_with_distance". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.3, and the level of structural similarity is more than or equal to 0.75.

Table C2: Using structural and similarity modules in parallel for "2_planes_with_distance"

2_planes_with_distance (cosine angle ≥ 0.3, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
2_lines_dist_2_planes (0.678)	2_lines_dist_2_planes (0.75)
2_planes_with_distance (1)	2_planes_with_distance (1)
2_points_2_lines_with_dist (0.5)	q_boss_radius (1)
2_ponts_with_distance (0.52)	q_cylinder (1)
planes_lines_points_distance (0.66)	q_hole_depth_radius (0.75)
q_dist_bounded_planes (0.94)	q_hole_radius_depth (1)
q_thinwall_parallel_planes (0.42)	q_radius_cylindrical_hole (0.75)
q_thinwall_solid_bounded_planes (0.34)	q_thinwall_simple_boundary (1)
q_thinwall_thick_less_0.5 (0.3)	q_thinwall_thick_less_0.5 (1)
	q_thinwall_medium_with_boundary (1)

Table C1 lists the names of exemplars that are retrieved for the query, "2_planes_with_distance". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.5, and the level of structural similarity is more than or equal to 0.5.

Table C1: Using structural and similarity modules in parallel for "2_planes_with_distance"

2_planes_with_distance (cosine angle ≥ 0.3, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
2_lines_dist_2_planes (0.678)	2_lines_dist_2_planes (0.75)
2_planes_with_distance (1)	2_planes_with_distance (1)
2_points_2_lines_with_dist (0.5)	q_boss_radius (1)
2_ponts_with_distance (0.52)	q_cylinder (1)
planes_lines_points_distance (0.66)	q_hole_depth_radius (0.75)
q_dist_bounded_planes (0.94)	q_hole_radius_depth (1)
	q_radius_cylindrical_hole (0.75)
	q_thinwall_simple_boundary (1)
	q_thinwall_thick_less_0.5 (1)
	q_thinwall_medium_with_boundary (1)

Table C2 lists the names of exemplars that are retrieved for the query, "2_planes_with_distance". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.5, and the level of structural similarity is more than or equal to 0.75.

Table C2: Using structural and similarity modules in parallel for "2_planes_with_distance"

2_planes_with_distance (cosine angle ≥ 0.3, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
2_lines_dist_2_planes (0.678)	2_lines_dist_2_planes (0.75)
2_planes_with_distance (1)	2_planes_with_distance (1)
2_points_2_lines_with_dist (0.5)	q_boss_radius (1)
2_ponts_with_distance (0.52)	q_cylinder (1)
planes_lines_points_distance (0.66)	q_hole_depth_radius (0.75)
q_dist_bounded_planes (0.94)	q_hole_radius_depth (1)
q_thinwall_parallel_planes (0.42)	q_radius_cylindrical_hole (0.75)
q_thinwall_solid_bounded_planes (0.34)	q_thinwall_simple_boundary (1)
q_thinwall_thick_less_0.5 (0.3)	q_thinwall_thick_less_0.5 (1)
	q_thinwall_medium_with_boundary (1)

Table C3 lists the names of exemplars that are retrieved for the query, "gear_pinion_02". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.3, and the level of structural similarity is more than or equal to 0.5. The number in parentheses besides each exemplar name in the list of semantically similar exemplars represents the value of the cosine angle between that exemplar and the query exemplar. . The number in parentheses besides each exemplar name in the list of structurally similar exemplars represents the measure of structural similarity between that exemplar and the query exemplar. The cells marked in yellow represent exemplars retrieved by both modules.

Table C3: Using structural and similarity modules in parallel for "gear_pinion_02"

gear_pinion_02 (cosine angle ≥ 0.3, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
gears_double_ratio_03 (0.76)	gears_double_ratio_03 (1)
gear_pinion_02 (1)	gear_pinion_02 (1)
q_compound_5_gears (0.53)	q_compound_5_gears (1)
q_radii_ratio (0.33)	q_radii_ratio (1)
q_radii_ratio_w_large_check (0.36)	q_radii_ratio_w_large_check (1)
spur_gears_02 (0.52)	spur_gears_02 (1)
	q_boss_radius (1)
	q_hole_depth_radius (1)
	q_hole_radius_depth (1)
	q_tangent_circles (1)

Table C4 lists the names of exemplars that are retrieved for the query, "gear_pinion_02". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.3, and the level of structural similarity is more than or equal to 0.75.

Table C4: Using structural and similarity modules in parallel for "gear_pinion_02"

gear_pinion_02 (cosine angle ≥ 0.3, structural similarity ≥ 0.75)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
gears_double_ratio_03 (0.76)	gears_double_ratio_03 (1)
gear_pinion_02 (1)	gear_pinion_02 (1)
q_compound_5_gears (0.53)	q_compound_5_gears (1)
q_radii_ratio (0.33)	q_radii_ratio (1)
q_radii_ratio_w_large_check (0.36)	q_radii_ratio_w_large_check (1)
spur_gears_02 (0.52)	spur_gears_02 (1)
	q_boss_radius (1)
	q_hole_depth_radius (1)
	q_hole_radius_depth (1)
	q_tangent_circles (1)

Table C5 lists the names of exemplars that are retrieved for the query, "gear_pinion_02". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.5, and the level of structural similarity is more than or equal to 0.5.

Table C5: Using structural and similarity modules in parallel for "gear_pinion_02"

gear_pinion_02 (cosine angle ≥ 0.5, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
gears_double_ratio_03 (0.76)	gears_double_ratio_03 (1)
gear_pinion_02 (1)	gear_pinion_02 (1)
q_compound_5_gears (0.53)	q_compound_5_gears (1)
spur_gears_02 (0.52)	q_radii_ratio (1)
	q_radii_ratio_w_large_check (1)
	spur_gears_02 (1)
	q_boss_radius (1)
	q_hole_depth_radius (1)
	q_hole_radius_depth (1)
	q_tangent_circles (1)

Table C6 lists the names of exemplars that are retrieved for the query, "gear_pinion_02". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.5, and the level of structural similarity is more than or equal to 0.75.

Table C6: Using structural and similarity modules in parallel for "gear_pinion_02"

gear_pinion_02 (cosine angle ≥ 0.5, structural similarity ≥ 0.75)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
gears_double_ratio_03 (0.76)	gears_double_ratio_03 (1)
gear_pinion_02 (1)	gear_pinion_02 (1)
q_compound_5_gears (0.53)	q_compound_5_gears (1)
spur_gears_02 (0.52)	q_radii_ratio (1)
	q_radii_ratio_w_large_check (1)
	spur_gears_02 (1)
	q_boss_radius (1)
	q_hole_depth_radius (1)
	q_hole_radius_depth (1)
	q_tangent_circles (1)

Table C7 lists the names of exemplars that are retrieved for the query, "q_hole". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.3, and the level of structural similarity is more than or equal to 0.5. The number in parentheses besides each exemplar name in the list of semantically similar exemplars represents the value of the cosine angle between that exemplar and the query exemplar. . The number in parentheses besides each exemplar name in the list of structurally similar exemplars represents the measure of structural similarity between that exemplar and the query exemplar. The cells marked in yellow represent exemplars retrieved by both modules.

Table C7: Using structural and similarity modules in parallel for "q_hole"

q_hole (cosine angle ≥ 0.3, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
q_cylinder (0.387)	q_cylinder (1)
q_hole (1)	q_hole (1)
q_hole_depth_radius (0.711)	q_hole_depth_radius (1)
q_hole_radius_depth (0.735)	q_hole_radius_depth (1)
q_radius_cylindrical_hole (0.504)	q_boss_radius (1)

Table C8 lists the names of exemplars that are retrieved for the query, “q_hole”. The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.3, and the level of structural similarity is more than or equal to 0.75.

Table C8: Using structural and similarity modules in parallel for “q_hole”

q_hole (cosine angle ≥ 0.3, structural similarity ≥ 0.75)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
q_cylinder (0.387)	q_cylinder (1)
q_hole (1)	q_hole (1)
q_hole_depth_radius (0.711)	q_hole_depth_radius (1)
q_hole_radius_depth (0.735)	q_hole_radius_depth (1)
q_radius_cylindrical_hole (0.504)	q_boss_radius (1)

Table C9 lists the names of exemplars that are retrieved for the query, “q_hole”. The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.5, and the level of structural similarity is more than or equal to 0.5.

Table C9: Using structural and similarity modules in parallel for “q_hole”

q_hole (cosine angle ≥ 0.5, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
q_hole (1)	q_hole (1)
q_hole_depth_radius (0.711)	q_hole_depth_radius (1)
q_hole_radius_depth (0.735)	q_hole_radius_depth (1)
q_radius_cylindrical_hole (0.504)	q_boss_radius (1)
	q_cylinder (1)

Table C10 lists the names of exemplars that are retrieved for the query, “q_hole”. The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.5, and the level of structural similarity is more than or equal to 0.75.

Table C10: Using structural and similarity modules in parallel for “q_hole”

q_hole (cosine angle ≥ 0.5, structural similarity ≥ 0.75)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
q_radius_cylindrical_hole (0.504)	q_cylinder (1)
q_hole (1)	q_hole (1)
q_hole_depth_radius (0.711)	q_hole_depth_radius (1)
q_hole_radius_depth (0.735)	q_hole_radius_depth (1)
	q_boss_radius (1)

Table C11 lists the names of exemplars that are retrieved for the query, "q_thinwall_thick_less_0.5". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.3, and the level of structural similarity is more than or equal to 0.5. The number in parentheses besides each exemplar name in the list of semantically similar exemplars represents the value of the cosine angle between that exemplar and the query exemplar. . The number in parentheses besides each exemplar name in the list of structurally similar exemplars represents the measure of structural similarity between that exemplar and the query exemplar. The cells marked in yellow represent exemplars retrieved by both modules.

Table C11: Using structural and similarity modules in parallel for "q_thinwall_thick_less_0.5"

q_thinwall_thick_less_0.5 (cosine angle ≥ 0.3, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
q_thinwall_thick_less_0.5 (1)	q_thinwall_thick_less_0.5 (1)
q_thinwall_medium_with_boundary (0.95)	q_thinwall_medium_with_boundary (1)
q_thinwall_simple_boundary (0.93)	q_thinwall_simple_boundary (1)
q_thinwall_parallel_planes (0.76)	
2_planes_with_distance (0.3)	
q_thinwall_solid_bound_parallel_planes (0.92)	
q_cylinder (0.39)	
q_thinwall_twoloops_simple (0.93)	

Table C12 lists the names of exemplars that are retrieved for the query, "q_thinwall_thick_less_0.5". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.3, and the level of structural similarity is more than or equal to 0.75.

Table C12: Using structural and similarity modules in parallel for "q_thinwall_thick_less_0.5"

q_thinwall_thick_less_0.5 (cosine angle ≥ 0.3, structural similarity ≥ 0.75)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
q_thinwall_thick_less_0.5 (1)	q_thinwall_thick_less_0.5 (1)
q_thinwall_medium_with_boundary (0.95)	q_thinwall_medium_with_boundary (1)
q_thinwall_simple_boundary (0.93)	q_thinwall_simple_boundary (1)
q_thinwall_parallel_planes (0.76)	
2_planes_with_distance (0.3)	
q_thinwall_solid_bound_parallel_planes (0.92)	
q_cylinder (0.39)	
q_thinwall_twoloops_simple (0.93)	

Table C13 lists the names of exemplars that are retrieved for the query, "q_thinwall_thick_less_0.5". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.5, and the level of structural similarity is more than or equal to 0.75.

Table C13: Using structural and similarity modules in parallel for "q_thinwall_thick_less_0.5"

q_thinwall_thick_less_0.5 (cosine angle ≥ 0.5, structural similarity ≥ 0.5)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
q_thinwall_thick_less_0.5 (1)	q_thinwall_thick_less_0.5 (1)
q_thinwall_medium_with_boundary (0.95)	q_thinwall_medium_with_boundary (1)
q_thinwall_simple_boundary (0.93)	q_thinwall_simple_boundary (1)
q_thinwall_parallel_planes (0.76)	
q_thinwall_twoloops_simple (0.93)	
q_thinwall_solid_bound_parallel_planes (0.92)	

Table C14 lists the names of exemplars that are retrieved for the query, "q_thinwall_thick_less_0.5". The minimum value of the cosine angle between the query vector and a target vector is more than or equal to 0.5, and the level of structural similarity is more than or equal to 0.75.

Table C14: Using structural and similarity modules in parallel for "q_thinwall_thick_less_0.5"

q_thinwall_thick_less_0.5 (cosine angle ≥ 0.5, structural similarity ≥ 0.75)	
Semantically Similar Exemplars	**Structurally Similar exemplars**
q_thinwall_thick_less_0.5 (1)	q_thinwall_thick_less_0.5 (1)
q_thinwall_medium_with_boundary (0.95)	q_thinwall_medium_with_boundary (1)
q_thinwall_simple_boundary (0.93)	q_thinwall_simple_boundary (1)
q_thinwall_parallel_planes (0.76)	
q_thinwall_twoloops_simple (0.93)	
q_thinwall_solid_bound_parallel_planes (0.92)	

Appendix D: Using the structural similarity measures first followed by semantic similarity measures.

This appendix lists the names of all exemplars that were retrieved when the two similarity modules were used in series. In this case, the structural similarity module was used first in order to retrieve exemplars that are structurally similar to the query exemplar. The semantic retrieval module is then used to retrieve exemplars that are semantically similar to each of the structurally similar exemplars. . The number in parentheses besides each exemplar name in the list of semantically similar exemplars represents the value of the cosine angle between that exemplar and the query exemplar. The minimum permissible value that the cosine angle can have for an exemplar to get retrieved is 0.3. Similarly, the number in parentheses besides each exemplar name in the list of structurally similar exemplars represents the measure of structural similarity between that exemplar and the query exemplar. The minimum permissible measure of structural similarity that a target exemplar may have in order to get retrieved is 0.5.

Table D15: Exemplars retrieved by using the structural retrieval module first

Query: 2_planes_with_distance.stp	
Structurally Similar Exemplars	**Semantically similar exemplars**
2_planes_with_distance	2_lines_dist_2_planes (0.678)
	2_planes_with_distance (1)
	2_points_2_lines_with_dist (0.5)
	2_ponts_with_distance (0.52)
	planes_lines_points_distance (0.66)
	q_dist_bounded_planes (0.94)
	q_thinwall_parallel_planes (0.42)
	q_thinwall_solid_bounded_planes (0.32)
	q_thinwall_thick_less_0.5 (0.3)
q_boss_radius	q_boss_radius (1)
	q_cylinder (0.47)
q_thinwall_medium_with_boundary	q_cylinder (0.3)
	q_thinwall_medium_with_boundary (1)
	q_thinwall_parallel_planes (0.79)
	q_thinwall_simple_boundary (0.94)
	q_thinwall_solid_bounded_planes (0.87)
	q_thinwall_thick_less_0.5 (0.95)

	q_thinwall_twoloops_simple (0.94)
q_cylinder	q_boss_radius (0.42)
	q_cyliner (1)
	q_double_radius (0.329)
	q_hole (0.387)
	q_hole_depth_radius (0.402)
	q_hole_radius_depth (0.425)
	q_radii_ratio (0.354)
	q_radii_ratio_w_large_check (0.369)
	q_radius_cylindrical_hole (0.315)
	q_thinwall_medium_with_boundary (0.308)
	q_thinwall_solid_bound_parallel_planes (0.371)
	q_thinwall_thick_less_0.5 (0.399)
q_hole_radius_depth	q_cylinder (0.402)
	q_hole (0.711)
	q_hole_depth_radius (1)
	q_hole_radius_depth (0.946)
	q_radius_cylindrical_hole (0.8411)
q_hole_depth_radius	q_cylinder (0.42)
	q_hole (0.735)
	q_hole_depth_radius (0.946)

	q_hole_radius_depth (1)
	q_radius_cylindrical_hole (0.918)
q_radius_cylindrical_hole	q_cylinder (0.315)
	q_hole (0.504)
	q_hole_depth_radius (0.841)
	q_hole_radius_depth (0.836)
	q_radius_cylindrical_hole (0.918)
q_thinwall_simple_boundary	q_thinwall_medium_with_boundary (0.94)
	q_thinwall_parallel_planes (0.85)
	q_thinwall_simple_boundary (1)
	q_thinwall_solid_bound_parallel_planes (0.9)
	q_thinwall_thick_less_0.5 (0.93)
	q_thinwall_twoloops_simple (1)
q_thinwall_thick_less_0.5	q_thinwall_medium_with_boundary (0.95)
	q_thinwall_parallel_planes (0.76)
	q_thinwall_simple_boundary (0.93)
	q_thinwall_solid_bound_parallel_planes (0.92)
	q_thinwall_thick_less_0.5 (1)
	q_thinwall_twoloops_simple (0.93)
	2_planes_with_distance (0.3)

	q_cylinder (0.399)

Table D16: Exemplars retrieved by using the structural retrieval module first

Query: gear_pinion_02.stp	
Structurally Similar Exemplars	**Semantically similar exemplars**
q_boss_radius (1)	q_boss_radius (1)
	q_cylinder (0.47)
q_hole_radius_depth	q_cylinder (0.402)
	q_hole (0.711)
	q_hole_depth_radius (1)
	q_hole_radius_depth (0.946)
	q_radius_cylindrical_hole (0.8411)
q_hole_depth_radius	q_cylinder (0.42)
	q_hole (0.735)
	q_hole_depth_radius (0.946)
	q_hole_radius_depth (1)
	q_radius_cylindrical_hole (0.918)
gears_double_ratio_03	gears_double_ratio_03 (1)
	gear_pinion_02 (0.769)
	q_compound_5_gears (0.88)
	q_coplanar_gears (0.69)
	spur_gears_02 (0.849)
gear_pinion_02	gears_double_ratio_03 (0.76)

	gear_pinion_02 (1)
	q_compound_5_gears (0.53)
	q_radii_ratio (0.33)
	q_radii_ratio_w_large_check (0.36)
	spur_gears_02 (0.52)
q_compound_5_gears	gears_double_ratio_03 (0.88)
	gear_pinion_02 (0.53)
	q_compound_5_gears (1)
	q_coplanar_gears (0.82)
	spur_gears_02 (0.91)
q_radii_ratio	gear_pinion_02 (0.33)
	q_connecting_rod_thick_enough (0.39)
	q_cylinder (0.35)
	q_double_radius (0.75)
	q_radii_ratio (1)
	q_radii_ratio_w_large_check (0.93)
q_radii_ratio_w_large_check	gear_pinion_02 (0.36)
	q_connecting_rod_thick_enough (0.42)
	q_cylinder (0.37)
	q_double_radius (0.79)
	q_radii_ratio (0.93)
	q_radii_ratio_w_large_check (1)

q_tangent_circles	q_tangent_circles (1)
	spur_gears_02 (0.32)
spur_gears_02	gears_double_ratio_03 (0.84)
	gear_pinion_02 (0.52)
	q_compound_5_gears (0.91)
	q_coplanar_gears (0.85)
	q_tangent_circles (0.32)
	spur_gears_02 (1)

Table D17: Exemplars retrieved by using the structural retrieval module first

Query: q_hole.stp	
Structurally Similar Exemplars	**Semantically similar exemplars**
q_boss_radius (1)	q_boss_radius (1)
	q_cylinder (0.47)
q_hole_radius_depth (1)	q_cylinder (0.402)
	q_hole (0.711)
	q_hole_depth_radius (1)
	q_hole_radius_depth (0.946)
	q_radius_cylindrical_hole (0.8411)
q_hole_depth_radius (1)	q_cylinder (0.42)
	q_hole (0.735)
	q_hole_depth_radius (0.946)
	q_hole_radius_depth (1)
	q_radius_cylindrical_hole (0.918)
q_cylinder (1)	q_boss_radius (0.42)
	q_cyliner (1)
	q_double_radius (0.329)
	q_hole (0.387)
	q_hole_depth_radius (0.402)
	q_hole_radius_depth (0.425)

	q_radii_ratio (0.354)
	q_radii_ratio_w_large_check (0.369)
	q_radius_cylindrical_hole (0.315)
	q_thinwall_medium_with_boundary (0.308)
	q_thinwall_solid_bound_parallel_planes (0.371)
	q_thinwall_thick_less_0.5 (0.399)
q_hole (1)	q_cylinder (0.38)
	q_hole (1)
	q_hole_depth_radius (71)
	q_hole_radius_depth (0.73)
	q_radius_cylindrical_hole (0.50)

Table D18: Exemplars retrieved by using the structural retrieval module first

Query: q_thinwall_thick_less_0.5.stp	
Structurally Similar Exemplars	**Semantically similar exemplars**
q_thinwall_medium_with_boundary (1)	q_cylinder (0.3)
	q_thinwall_medium_with_boundary (1)
	q_thinwall_parallel_planes (0.79)
	q_thinwall_simple_boundary (0.94)
	q_thinwall_solid_bounded_planes (0.87)
	q_thinwall_thick_less_0.5 (0.95)
	q_thinwall_twoloops_simple (0.94)
q_thinwall_simple_boundary	q_thinwall_medium_with_boundary (0.94)
	q_thinwall_parallel_planes (0.85)
	q_thinwall_simple_boundary (1)
	q_thinwall_solid_bound_parallel_planes (0.9)
	q_thinwall_thick_less_0.5 (0.93)
	q_thinwall_twoloops_simple (1)
q_thinwall_thick_less_0.5	q_thinwall_medium_with_boundary (0.95)
	q_thinwall_parallel_planes (0.76)
	q_thinwall_simple_boundary (0.93)

	q_thinwall_solid_bound_parallel_planes (0.92)
	q_thinwall_thick_less_0.5 (1)
	q_thinwall_twoloops_simple (0.93)

Appendix E: Using semantic similarity measures followed by structural similarity measures.

This appendix lists the exemplars retrieved for different queries by using the two modules of similarity in series. In this case, the semantic similarity module is used first to retrieve a list of exemplars that are semantically similar to the query exemplar. The structural similarity module is used to retrieve exemplars that are structurally similar to each of the semantically similar exemplars. The number in parentheses besides each exemplar name in the list of semantically similar exemplars represents the value of the cosine angle between that exemplar and the query exemplar. The minimum permissible value that the cosine angle can have for an exemplar to get retrieved is 0.3. Similarly, the number in parentheses besides each exemplar name in the list of structurally similar exemplars represents the measure of structural similarity between that exemplar and the query exemplar. The minimum permissible measure of structural similarity that a target exemplar may have in order to get retrieved is 0.5.

Table E1: Exemplars retrieved by using the semantic retrieval module first

Query: 2_planes_with_distance	
Semantically Similar Exemplars	**Stucturally similar exemplars**
2_lines_dist_2_planes (0.678)	2_lines_dist_2_planes (1)
	q_thinwall_simple_boundary (0.75)
2_planes_with_distance (1)	2_planes_with_distance≤ (1)
	q_boss_radius (1)
	q_cylinder (1)
	q_hole_depth_radius (0.75)
	q_hole_radius_depth (1)
	q_radius_cylindrical_hole (0.75)
	q_thinwall_medium_with_boundary (1)
	q_thinwall_simple_boundary (1)
	q_thinwall_thick_less_0.5 (1)
2_points_2_lines_with_dist (0.5)	2_points_2_lines_with_dist (1)
	q_thinwall_simple_boundary (1)
2_points_with_distance (0.52)	2_points_2_lines_with_dist (1)
	2_ponts_with_distance (1)
	planes_lines_points_distance (1)
	q_hole_depth_radius (1)
	q_thinwall_simple_boundary (1)

planes_lines_points_distance (0.66)	planes_lines_points_distance (1)
q_dist_bounded_planes (0.94)	q_boss_radius (1)
	q_cylinder (1)
	q_dist_bounded_planes (1)
	q_hole_radius_depth (1)
	q_thinwall_medium_w_boundary (1)
	q_thinwall_simple_w_boundary (1)
	q_thinwall_twoloops_simple (1)
q_thinwall_parallel_planes (0.42)	2_lines_dist_2_planes (1)
	2_planes_with_distance(1)
	planes_lines_points_distance (1)
	q_boss_radius (1)
	q_cylinder (1)
	q_dist_bounded_planes (1)
	q_hole (1)
	q_hole_depth_radius (1)
	q_hole_radius_depth (1)
	q_radius_cylindrical_hole (1)
	q_thinwall_medium_with_boundary (1)
	q_thinwall_parallel_planes (1)
	q_thinwall_simple_boundary (1)
	q_thinwall_solid_bounded_planes (1)

	q_thinwall_thick_less_0.5 (1)
	q_thinwall_twoloops_simple (1)
q_thinwall_solid_bound_parallel_planes (0.32)	q_thinwall_medium_with_boundary (1)
	q_thinwall_simple_boundary (1)
	q_thinwall_solid_bounded_planes (1)
	q_thinwall_thick_less_0.5 1(1)
q_thinwall_thick_less_0.5 (0.3)	q_thinwall_medium_with_boundary (1)
	q_thinwall_simple_boundary (1)
	q_thinwall_thick_less_0.5 (1)

Table E19: Exemplars retrieved by using the semantic retrieval module first

Query: gear_pinion_02	
Semantically Similar Exemplars	**Structurally similar exemplars**
gears_double_ratio_03 (0.76)	gears_double_ratio_03 (1)
gear_pinion_02 (1)	q_boss_radius (1)
	q_hole_depth_radius (1)
	q_hole_radius_depth (1)
	gears_double_ratio_03 (1)
	gear_pinion_02 (1)
	q_compound_5_gears (1)
	q_radii_ratio (1)
	q_radii_ratio_w_large_check (1)
	q_tangent_circles (1)
	spur_gears_02 (1)
q_compound_5_gears (0.53)	gears_double_ratio_03 (1)
	q_compound_5_gears (1)
q_radii_ratio (0.33)	gears_double_ratio_03 (1)
	q_compound_5_gears (1)
	q_radii_ratio (1)
	q_radii_ratio_w_large_check (1)
q_radii_ratio_w_large_check (0.36)	gears_double_ratio_03 (1)

	q_compound_5_gears (1)
	q_radii_ratio (1)
	q_radii_ratio_w_large_check (1)
spur_gears_02 (0.52)	q_boss_radius (0.7)
	q_hole_depth_radius (0.6)
	spur_gears_02 (1)

Table E20: Exemplars retrieved by using the semantic retrieval module first

Query: q_hole	
Semantically Similar Exemplars	**Structurally similar exemplars**
q_cylinder (0.387)	q_boss_radius (1)
	q_cylinder (1)
	q_hole_depth_radius (1)
	q_hole_radius_depth (1)
q_hole (1)	q_boss_radius (1)
	q_cylinder (1)
	q_hole (1)
	q_hole_depth_radius (1)
	q_hole_radius_depth (1)
q_hole_depth_radius (0.71)	q_hole_depth_radius (1)
q_hole_radius_depth (0.73)	q_boss_radius (1)
	q_cylinder (1)
	q_hole_depth_radius (0.85)
	q_hole_radius_depth (1)
q_radius_cylindrical_hole (0.5)	q_hole_depth_radius (0.6)
	q_radius_cylindrical_hole (1)

Table E21: Exemplars retrieved by using the semantic retrieval module first

Query: q_thinwall_thick_less_0.5	
Semantically Similar Exemplars	**Stucturally similar exemplars**
2_planes_with_distance (0.3)	2_planes_with_distance (1)
	q_boss_radius (1)
	q_cylinder (1)
	q_hole_depth_radius (0.75)
	q_hole_radius_depth (1)
	q_radius_cylindrical_hole (0.75)
	q_thinwall_medium_with_boundary (1)
	q_thinwall_simple_boundary (1)
	q_thinwall_thick_less_0.5 (1)
q_cylinder (0.399)	q_boss_radius (1)
	q_cylinder (1)
	q_hole_depth_radius (0.57)
	q_hole_radius_depth (1)
q_thinwall_medium_with_boundary (0.95)	q_thinwall_medium_with_boundary (1)
	q_thinwall_simple_with_boundary (0.75)
q_thinwall_solid_bound_parallel_planes (0.9256)	q_thinwall_medium_with_boundary (1)
	q_thinwall_simple_boundary (1)

	q_thinwall_solid_bound_parallel_planes (1)
	q_thinwall_thick_less_0.5 (1)
q_thinwall_thick_less_0.5 (1)	q_thinwall_medium_with_boundary (1)
	q_thinwall_simple_boundary (1)
	q_thinwall_thick_less_0.5 (1)
q_thinwall_simple_boundary (0.93)	q_thinwall_simple_boundary (1)

Appendix F: Structured exemplars authored by user 1.

This appendix lists the structural exemplars authored by user 1. These exemplars are numbered the same way they are numbered in Table 11.2.

Alpha Match:
Plane "S1";
Plane "S2";
Line "C1";
Line "C2";
Alpha Extract:
Distance "dist" (Plane_1, Plane_2, dist);
Parallel (Plane_1, Plane_2);
ID "Plane_1";
ID "Plane_2";
ID "dist";

W2_1: 2_planes_with_distance_01.stp

Alpha Match:

Circle “Pinion”;

Circle “Gear”;

Alpha Extract:

Tangent (Pinion, Gear);

Radius “pinion” (Pinion, pinion);

Radius “gear” (Gear, gear);

ID “pinion”;

ID “Pinion”;

Equation “eq1” (gear > pinion);

G1_1: gear_pinion_02_01.stp

Alpha Match:

Solid "Body";
Cylindrical Surface "Hole";
Circle "Top_Edge";
Circle "Bottom_Edge";
Plane "Top_Surface";
Plane "Bottom_Surface";
Boundary (Body, {Hole, Top_Surface, Bottom_Surface});
Boundary (Hole, {Top_Edge, Bottom_Edge});
Boundary (Top_Surface, Top_Edge);
Boundary (Bottom_Surface, Bottom_Edge);

Alpha Extract:

Vector "Top_TC";
Vector "Bottom_TC";
TC_Normal_Vector (Body, Top_Surface, Top_TC);
TC_Normal_Vector (Body, Bottom_Surface, Bottom_TC);
Opposite_Direction (Top_TC, Bottom_TC);

F4_1: q_hole_01.stp

Alpha Match:

Solid "Body";
Plane "Side 1";
Plane "Side 2";
Plane "Top Surface";
Line "Edge 1";
Line "Edge 2";
Boundary (Body, {Side 1, Side 2, Top Surface});
Boundary (Top Surface, {Edge 1, Edge 2});
Boundary (Side 1, Edge 1);
Boundary (Side 2, Edge 2);

Alpha Extract:

Parallel (Side 1, Side 2);
Distance (Side 1, Side 2, distance);
Parameter "distance";
Vector "Side_1_TC";
Vector "Side_2_TC";
TC_Normal_Vector (Body, Side 1, Side_1_TC);
TC_Normal_Vector (Body, Side 2, Side_2_TC);
Opposite_Direction (Side_1_TC, Side_2_TC);
Equation "bool 1" (distance < 0.5);
ID (Side 1);
ID (Side 2);
ID (Top);

W9_1: q_thinwall_thick_less_0.5_01.stp

Appendix G: Semantic descriptions written by User 1 (Dr. Joshua Summers).

This exemplar is used to find the distance between two parallel planes. Both planes are highlighted for the user and the distance value is shown the user. If no planes are found that are parallel, then no matches will be found. This exemplar may be used as a basis for finding thin walls, for finding the heights of bosses, or for finding the depths of blind holes. **W2_1: 2_planes_with_distance_01.des**
This exemplar extracts the radii of two circles, comparing the values. If one is larger than the second, it can be assumed to be the gear with the smaller one potentially the pinion. This exemplar is a simple one that can be expanded for use in gear train design and sizing. **G1_1: gear_pinion_02_1.des**
This exemplar is used to find potential holes or bosses. It matches cylindrical surfaces that are in turn bounded by two circles, who in turn bound two planes. The cylindrical surface is then returned to the user (highlighted). **F4_1: q_hole_1.des**
This exemplar is used as a beginning point for building thinwall feature recognition exemplars. First, three planes that are all bounding the same solid are found. Then, the distance between two of the parallel planes are extracted. Finally, the distance is checked against a thinwall threshold value of 0.5. If the distance is less than 0.5, then the exemplar holds to be true and a potential thinwall feature is found. **W9_1: q_thinwall_thick_less_0.5_1.des**

Appendix H: Semantic descriptions written by user 2 (Sudhakar Teegavarapu)

The intent of this exemplar is to find the distance between a set of two parallel planes. The selected set of planes is highlighted. If the parallel constraint holds true for this highlighted set of planes, the distance between them is extracted and displayed. **W2_2: 2_planes_with_distance.des**
The intent of this exemplar is to identify the gear and pinion among two meshing gears. The query locates two circles of different radii in a model and compares them using an equation. The equation imposes a constraint that the radius of the gear is greater than the radius of the pinion. However, it is not clear how the information is highlighted to the user as there are no IDs. **G1_2: gear_pinion_02.des**
The intent of this exemplar is to find the height of a boss in the given model whose radius is less than 1 unit. Two circles each coincident on the bounding planes, and the cylindrical surface between the circles constitute the boss. The two bounding surfaces are checked by the parallel constraint. In case the surfaces are parallel, the distance between them is extracted as the height of the boss. Also, radius of one of the circles is extracted as the radius of the boss. Since, it is not verified if the radius of the two circles is equal, tapered bosses could also be extracted using this query. However, in case of a tapered boss, radius of one of the circles cannot be taken as the radius of the boss. **F1_2: q_boss_radius.des**

The intent of this exemplar is to locate coplanar gears in a given model. A specific plane in the model is selected. A circle in the model, which represents the pitch circle of a gear, is checked for coincidence on the selected plane. Similarly, a second distinctive circle in the model, which represents the pitch circle of another gear, is also checked for coincidence on the same plane. Incase both the circles are coincident on the selected plane; the circles are highlighted in the display. It is not considered important to check if the circles are tangential to each other, as the intent is to find only coplanar gears, not 'meshing' gears. However, this query could also extract coplanar circles that represent holes or bosses.

G4_2: q_coplanar_gears.des

The intent of this exemplar is to locate all the holes in a given model. A cylindrical surface bound by two circles is located in the model. The two circles are coincident of two planes respectively. The query extracts the cylindrical surface and highlights it in the display.

F4_2: q_hole.des

The intent of this exemplar is to find the depth and radius of a hole in the given model. Two circles each coincident on the bounding planes, and the cylindrical surface between the circles constitute the hole. The distance between two points that are coincident on the bounding planes, connected by a line coincident on the cylindrical surface, is extracted as the depth of the hole. Since, it is not verified if the radius of the two circles is equal, tapered holes could also be extracted using this query. However, in case of a tapered hole, radius of one of the circles cannot be taken as the radius of the hole.

F5_2: q_hole_depth_radius.des

The intent of this exemplar is to find thin walls in a given model. The query identifies a solid manifold bounded by three planes. It is checked if any two of the three planes are parallel to each other. The query then returns the distance between these two parallel planes, which is the thickness of the wall. It is verified if the thickness is less than 0.5 units, in order to ensure it is a thin wall.

W9_2: q_thinwall_thick_less_0.5_2

Appendix I: Semantic descriptions written by user 3 (Shashidhar Putti)

This exemplar may be used to identify two planes parallel to each other and can be used to find the distance between them. The alpha portion of the exemplar represents the entities explicitly present in the model while the exemplar represents the relations that are implicit. Here, when a model is queried against this match it returns a match if the model has two planes, represented by S1, S2 in the exemplar, that are parallel to each other, represented by Parallel (S1, S2) relation in the alpha extract. And, when a match is found, the exemplar returns the distance between the two planes using the distance relation included in the exemplar, Distance ({S1, S2}, dist) relation in the alpha extract. **W2_3: 2_planes_with_distance.des**
This exemplar may be used to identify and tag the gear and pinion in a gear pair. When a model is queried against this exemplar, it returns a match if the model has two gears, presented as circles C1, C2 in the alpha match portion of the exemplar and if the radius of one gear is greater than the other, represented by the Equation “eq1”(gear>pinion) elation in the extract portion of the exemplar. **G1_3: gear_pinion_02.des**
This exemplar may be used to find a cylindrical hole in a model. When a model is queried against this exemplar, it returns a match if a cylindrical surface, S1 in the exemplar, bound by two circles, C1, C2 in the exemplar, is found. Also, each of these circles is required to be bound by a plane each, S2, S3 in the exemplar. When a match is found, the exemplar highlights the model represented by the ID(S1) constraint in the alpha extract portion of the exemplar. **F4_3: q_hole.des**

This exemplar may be used to identify walls with thickness less than 0.5. When a model is queried against this exemplar, it returns a match if the model has a solid manifold bound by three planes, represented by the alpha match portion of the exemplar. While at least two of these planes are expected to be parallel, represented by the Parallel (S1, S3) relation in the alpha extract portion of the exemplar , the distance between them should be less than 0.5 , represented by the Equation “eq1” (thickness < 0.5) relation in the alpha extract portion of the exemplar.

W9_3: q_thinwall_thick_less_0.5.des

Appendix J: Semantic descriptions written by user 4 (Vikram Bapat)

The intent of this exemplar is to find distance between planes in the model which are parallel. The model could have planes which were not explicitly designed to be parallel. The exemplar algorithm will match to find pairs of planes in the model which are parallel, id them and then evaluate the distance between the two planes. **W2_4: 2_planes_distance.des**
The intent of this exemplar is to find bosses in the model with radius less than 1 unit. The exemplar defines the structure of a hole to match with the model. This structure is defined as a cylindrical surface bound by two circles and two planes, the two planes are bound entirely by the two circles bounding the cylindrical surface. Once matches are found in the model, the algorithm evaluates the radius of the hole as the radius of a bounding circle. It checks the evaluated radius to see if it is less than one unit. **F1_4: q_boss_radius.des**
The intent of this exemplar is to find circles which are coplanar. The exemplar is matched with the model to find pairs of circle and evaluated to check whether they are coincident with the same plane. **G4_4: q_coplanar_gears.des**
The intent of this exemplar is to detect the holes in the model and to evaluate their depth and radius. The exemplar defines the structure of a hole to match with the model. This structure is defined as a cylindrical surface bound by two circles and two planes, the two planes are bound entirely by the two circles bounding the cylindrical surface. Once matches are found in the model, the algorithm evaluates the radius of the hole as the radius of a bounding circle. It also evaluates a line coincident with cylindrical surface and bound by points coincident on planes bounding the cylindrical surface. For the matches found it calculates the distance between two points as the depth of the cylinder. **F5_4: q_hole_depth_radius.des**

REFERENCES

1. Cicirello, V., Regli, W., *An approach to a feature-based comparison of solid models of machined parts.* Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2002. 16: p. 385-399.
2. Jiantao, P., Jayanti, S., Hou, S., Ramani, K., *Similar 3D Model Retrieval Based on Multiple Level of Detail*, in *The 14th Pacific Conference on Computer Graphics and Applications*. 2006: Taipei, Taiwan.
3. Akbar, S., Kung, J., Wagner, R, *Multi-Feature based 3D Model Similarity Retrieval*, in *1st International Conference on Digital Information management*. 2006.
4. Anandan, S., Srirangam, M., Summers, J.D., *A Case Study in the use of Design Exemplar as a Search and Retrieval Tool*, in *ASME IDETC / DTM*. 2008: New York, NY.
5. Jiantao, P., Ramani, K., *Priority-Based Geometric Constraint Satisfaction.* Transactions of the ASME Journal of Computing and Information Science in Engineering, 2007. **7**: p. 322-329.
6. Heather L. Rea, D.E.R.C., Jonathan R. Corney, Nick K. Taylor *A Surface Partitioning Spectrum (SPS) for Retrieval and Indexing of 3D CAD Models*, in *3rd International Symposium on 3D Data Processing, Visualization, and Transmission, 3DPVT, .* 2004: Thessaloniki, Greece.
7. Kayyar, M., Summers, J.D., Ameri, F., Biggers, S. , *A Case Study of Design Process and Development of a Design Enabling Tool for Wright Metal Products*, in *ASME DETC*. 2007: Las Vegas, NV.
8. Summers, J., Shah, J., *The Design Exemplar: A New Data Structure for Design Automation.* Journal of Mechanical Design, 2004. 126(5): p. 775-87.
9. *http://www.m-w.com*. [cited.
10. Summers, J., Lacroix, Z., Shah, J., *Case-Based Design Facilitated by the Design Exemplar*, in *Seventh International Conference on Artificial Intelligence in Design, .* 2002: Netherlands.
11. *http://office.microsoft.com/*. [cited.
12. Pahl, G., Beitz, W., *Engineering Design: A Systematic Approach*. 1996: Springer.
13. Bettig, B., Shah, J., Summers, J., *Domain Independent Characterization of Parametric and Geometric Problems in Embodiment Design*, in *Proceedings of the ASME DETC* 2000: Baltimore, M.D. .

14. Chartrand, G., *Introductory Graph Theory*. 1977, New York, NY.: Dover Publications Inc.

15. Summers, J., Divekar, A., Anandan, S., *Towards Estabishing the Design Exemplar as a CAD Query Language.* Computer-Aided Design & Applications, 2006. **3**(1-4): p. 523-534.

16. Putti, S., Summers, J. *Dynamic Networks: Towards a Mechanical Design Visual Programming Knowledge*. in *Proceedings of the IDETC / CIE ASME Design Engineering Technical Conferences*. 2006. Philadelphia, PA.

17. Harary, F., *Graph Theory*. Addison-Wesley in mathematics. 1969: Addison-Wesley Pub. Co.

18. Gross, J., Yellen, Jay., ed. *Handbook of Graph Theory*. Discrete mathematics and its applications. 2004, CRC Press.

19. Zhang, C., Chen, T. *Efficient Feature Extraction for 2D/3D objects in mesh representation*. in *Proceedings of Inernational Conference on Image Processing*. 2001. Thessaloniki, Greece.

20. Paquet, E., et al., *Description of shape information for 2D and 3D objects.* Signal Processing: Image Communication, 2000. 16: p. 103-122.

21. Corney, J., et al., *Coarse Filters for Shape Matching.* IEEE Computer Graphics and Applications, 2002**.** 22(3): p. 65-74.

22. Osada, R., et al, *Shape Distributions.* ACM Transactions on Graphics, 2002. 21(4): p. 807-832.

23. Ohbuchi, R., Otagiri, T., Ibato, M., Takei, T. *Shape-Similarity Search of Three-Dimensional Models using Parameterized Statistics*. in *Proceedings of 10th Pacific Graphics*. 2002. Beijing, China.

24. Funkhouser, T., et al., *A Search Engine for 3D Models.* ACM Transactions on Graphics, 2003. 22(1).

25. Shum, H., et al. *On 3d Shape Similarity*. in *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition*. 1996.

26. Kortgen, M., et al. *3D Shape Matching with 3D shape contexts*. in *The 7th Central European Seminar on Computer Graphics*. April 2003.

27. Iyer, N., et al. *Dynamic Early Design Advice Using Shape Retrieval*. in *International CIRP Design Seminar, CIRP*. 2003. Grenoble, France.

28. Bunke, H., *Graph Matching: Theoretical Foundations, Algorithms, and Applications.* Vision Interface 2000, 2000: p. 82-8.

29. Bunke, H., Jiang, X., Kandel, A.,, *On the Minimum Common Supergraph of Two Graphs.* Computing, 2000. 65(1).

30. Papadopoulos, A., Manolopoulos, Y., *Structure-Based Similarity Search with Graph Histograms.* DEX/IWOSS International Workshop on Similarity Search, IEEE Computer Society, 1999: p. 174-178.

31. Tversky, A., *Preference, Belief and Similarity: Selected Writings.* The MIT Press, 2003.

32. Bunke, H., *Error Correcting Graph Matching: On the Influence of Underlying Cost Function.* IEEE Transactions PAMI, 1999. 21: p. 917-922.

33. Almohamed, H., *A linear Programming Approach for the Weighted Graph Matching Problem.* IEEE Transactions on PAMI, 1993. 15: p. 522-525.

34. Umeyama, S., *An Eigen Decomposition Approach to Weighted Graph Matching Problems.* IEEE Transactions PAMI, 1988. 10: p. 695-703.

35. Ullman, J.R., *An Algorithm for Subgraph Isomorphism.* Journal of the Association for Computing Machinery, 1976. 23(1): p. 31-42.

36. McGregor, J., *BackTrack Search Algorithms and the Maximal Subgraph Problem.* Software Practice and Experience, 1982. 12: p. 23-34.

37. Levi, G., *A Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs.* Calcolo, 1972. 9: p. 341-354.

38. Pelilo, M., *A Unifying Framework for Relational Structure Matching*, in *14th ICPR*. 1998: Brisbane.

39. Sanfeliu, A., Fu, K.,, *A Distance Measure Between Attributed Relational Graphs for Pattern Recognition.* IEEE Transactions SMC, 1983. 13: p. 353-363.

40. Shapiro, L., Haralick, R., *Structural Descriptions and Inexact Matching.* IEEE Transactions on PAMI, 1981. 3: p. 504-519.

41. Tsai, W., Fu, K., *Error-Correcting Isomorphisms of Attributed Relational Graphs for Pattern Recognition.* IEEE Transactions on SMC, 1979. 9: p. 757-768.

42. Lobner, S., *Understanding Semantics*. 2002, New York: Oxford University Press.

43. Lyons, J., *Linguistic Semantics - An Introduction*. 1995, New York: Cambridge University Press.

44. Katz, J.J., *Semantic Theory*. 1972, New York: Harper and Row Publishers.

45. Ullmann, S., *Principles of Semantics*. 1963, New York: Glaslow University Publications.

46. Hornstein, N., *Logic As Grammar*. 1984: Bradford Books.

47. O' Grady, W.D., *Contemprory Linguistics - An Introduction*. 1989, New York: St. Martin's Press.

48. Paducheva, E.V., *Semantic Features and Selection Restrictions*, in *Proceedings of the Fifth Conference on European Chapter of the Association for Computational Linguistics*. 1991: Berlin, Germany.

49. Cera, C.D.R., Braude, I., Shapirstein, Y., Foster, C. V., *A Collaborative 3D Environment for Authoring Design Semantics.* Computer Graphics and Applications, IEEE, 2002. May/Jun 22(3): p. 43-55.

50. Ullman, D., *The Mechanical Design Process*. 1992, New York: McGraw-Hill.

51. Chakrabarti, A., *Engineering Design Synthesis: Understanding, Approaches, and Tools*. 2002, New York: Springer.

52. Summers, J., *Reasoning in Engineering Design*, in *ASME Internation Design Engineering Technical Conference*. 2005: Long Beach, LA.

53. Lee, K.-Y., Lee, W.-J., Roh, M.-I, *Development of a Semantic Product Modeling System for initial hull structure in shipbuilding.* Robotics and Computer-Integrated Manufacturing, 2004. 20(3): p. 211-223.

54. Ganeshan, R., G.J., Finger, S., *A Framework for Representing Design Intent.* Design Studies Journal, 1984. 15(1): p. 59-84.

55. Burge, J., Brown, D., *Reasoning with Design Rationale.* Artificial Intelligence in Design, 2000. http://web.cs.wpi.edu/~dcb/Papers/AID00-janet.pdf.

56. Hounsell, M.S., Case, K., *Morphological and Volumetrical Feature-based Designer's Intents.* Advances in Manufacturing Technology, 1997: p. 64-68.

57. Nielson, E.H., Dixon, J.R., Zinsmeister. *Capturing and Using Designer Intent in a Design-with-Features System.* in *DTM'91: Third International Conference on Design Theory and Methodology (ASME-DE).* 1991. Miami, FL, USA, 95-102.

58. Grumman, N., *Performance Specification for the Geostationary Operational Environmental Satellite*. 1997.

59. Schmid, J., *Submerged Water Activity Platform*, U. States, Editor. 2004.

60. Case, K., Gao, J., *Feature Technology: An overview.* International Journal of Computer Aided Manufacturing, 1993. 6(1 & 2): p. 2 - 12.

61. Shah, J., *Assesment of Features Technology.* Computer Aided Design, 1991. 23(5): p. 331 - 343.

62. Davis, R., Shrobe, H., Szolovits, P., *What is Knowledge Representation?* . AI Magazine, 1993. 14(1): p. 17-33.

63. Luger, G., Stubblefield, W., *Artificial Intelligence and the Design of Expert Systems*. 1989, California: The Benjamin/Cummings Publishing Company.

64. Shapiro, S., *Knowledge Representation.* Encylopedia of Cognitive Science, 2003. 2: p. 671-680.

65. Reichgelt, H., *Knowledge Representation: An AI Perspective*. 1991: Ablex Pub. Corp.

66. T., J., M., Bench - Capon, *Knowledge Representation - An Approach to Artificial Intelligence*. 1990: Academic Press.

67. McCarthy, J., Hayes, P. J., *Some philosophical problems from the standpoint of artificial intelligence.* Machine Intelligence, 1969. 4: p. 463 - 502.

68. Rich, E., Knight, K., *Artificial Intelligence*. 1991, New York: McGraw-Hill, Inc.

69. Hasse, P., Stojanovic, L., *The Semantic Web: Research and Applications*. Lecture Notes in Computer Science. Vol. 3532/2005. 2005, Berlin / Heidelberg: Springer.

70. Joseph B. Kopena, W.C.R. *Design Repositories on the Semantic Web With Description-Logic Enabled Services*. in *Semantic Web and Databases Workshop*. 2003.

71. Udoyen, N., Rosen, D., *Description Logic Representation of Finite Element Analysis Models for Auotmated Retrieval*, in *ASME 2006 International Design Engineering Technical Conferences*. 2006: Philadelphia, PA.

72. McGuiness, D.L., Wright, A. J. R., *Conceptual modelling for configuration : A description logic-based approach.* Artificial intelligence for engineering design, analysis and manufacturing 1998. 12(4).

73. Moore, R., *The role of Logic in Knowledge Representation and Commonsense Reasoning.* AAAI, 1982: p. 428 - 433.

74. Rechenmann, F. *Declarative and procedural object-based knowledge modelling*. in *International Conference on Systems, Man and Cybernetics.* 1993.

75. Hartley, R., Pfeiffer, H., *Visual Representation of Procedural Knowledge.* IEEE International Symposium on Visual Languages, 2000.

76. Hayes-Roth, F., Waterman, D., Lenat, D., ed. *D. Waterman, Hayes - Roth (Ed.) Principles of Pattern Directed Inference Systems*. 1978, Academic Press: New York.

77. Peters, S., Shrobe, H., *Semantic Networks for Knowledge Representation in an Intelligent Environment*, in *1st Annual IEEE International Conference on Pervasive Computing and Communications*. 2003: Ft. Worth, Texas.

78. Hartley, R.T., Barnden, H.T., *Semantic Networks: Visualizations of Knowledge.* Trends in Cognitive Science, 1997. 1(5): p. 169-175.

79. Benjamin, K., *A Frame for Frames: Representing Knowledge for Recognition.* Representing and Understanding: Studies in Cognitive Science, 1975: p. 151-184.

80. Lonnekar, B. *The Frame Idea in Semantics and Knowledge Representation*. in *5th Multiconference on Systemics, Cybernetics and Informatics (Sci 2001)*. 2001. Orlando, Florida, USA.

81. Winston, P.H., ed. *psychology of computer vision*. 1975, McGraw Hill.

82. Fikes, R., Kehler, T., *The role of frame-based representation in reasoning*. Communications of the ACM, 1985. 28: p. 235 - 246.

83. Hayes, P., ed. *D. Metzing (Ed.), Frame Conceptions and text understanding*. 1979.

84. V. Harmandas, M.S., M.D. Dunlop. *Image Retrieval by Hypertext Links*. in *20th International Conference on Research and Development in Information Retrieval*. 1997.

85. Carson, C., Thomas, M., Belongie, S., Hellerstein, J.M., Malik, J., *Blobworld: A system for region based image indexing and retrieval*, in *Third International Conference on Visual Information Systems (VISUAL '99)*. 1999.

86. Iwanska, L., Shapiro, Stuart., ed. *Natural language processing and knowledge representation : language for knowledge and knowledge for language* 2000, MIT Press.

87. Guarino, N. *Formal Ontology and Information Systems*. in *Proceedings of FOIS' 98* 1998. Trento, Italy.

88. Sim, S., Duffy, AH, *Toward an ontology of generic engineering design activities.* Research in Engineering Design, 2003. 14(4): p. 220 - 223.

89. Stahovich, T., Davis, Randall., Shrobe, Howard., *An Ontology of Mechanical Devices.* Association for the Advancement of Artificial Intelligence, Working Notes, Reasoning about Fucntion, 1993: p. 137-140.

90. Zhanjun Li et al. *Semantics-based design knowledge and retrieval*. in *Proceedings of IDETC/CIE* 2005. Long Beach, CA.

91. http:://torre.fffcmpa.tche.br/Volumes/SPSS/SPSS+Products+and+Services/SPSS +Product+Catalog/Spec+Sheets/TextSmart.pdf. [cited.

92. http:://www.megaputer.com/products/a/index.php3. [cited.

93. Esser, W., *Fault-tolerant Fulltext Search for Large Multilingual Scientific Text Corpora.* Journal of Digital Information, 2004. 6(1).

94. Park, E.K., Moon, S.I., Ra, D. Y., Jang, M.G. *Web Document Retrieval Using Sentence-query Similarity.* in *Proceedings of the 11 TextREtrieval Conference (TREC-11), notebook version* 2002.

95. Lynch, P., at al. *An Evaluation of New and Old Similarity Ranking Algorithms*. in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC '04)*. 2004.

96. Salton, G., Buckley, C., *Term Weighting Approaches in Automatic Text Retrieval.* Information Processing and Management, 1988. 24: p. 513-523.

97. Besancon, R., Rajman, M., Chappelier, J.-C. *Textual similarities based on a distributional approach*. in *Proceedings of the Tenth International Workshop on Database and Expert Systems Applications*. 1999. Florence, Italy.

98. Zobel, J., Dart, Ph., *Finding Approximate Matches in Large Lexicons.* Software Practice and Experience, 1995: p. 331-345.

99. Odel, M.K., Russel, R. 1918.

100. Ristad, E.S., Yianilos, P.N., *Learning string-edit distance.* IEEE Transactions on PAMI, 1998. 20(5): p. 522-532.

101. Navarro G., B.-Y.R., Sutinen E., Tarhio, J., *Indexing Methods for Approximate String Matching.* IEEE Bulletin of the Technical Comittee on Data Engineering, 2001. 24(4): p. 19-27.

102. Hatzivassilohlou, V.e.a. *Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning.* in *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. 1999. College Park, Maryland.

103. Olivier Ferret, B.G., Nicolas Masson. *Thematic segmentation of texts: two methods for two kinds of texts*. in *Proceedings of the 17th International Conference on Computational Linguistics*. 1998.

104. Masson, N., *An Automatic Method for Document Structuring*, in *Proceedings of the 18th Annual International ACM-SIGIR Conference on Research and Developement in Information Retrieval* 1995: Seattle, Washington.

105. Poli., C., *Design for manufacturing : a structured approach* 2001: Butterworth-Heinemann.

106. Kalpakjian, S., *Manufacturing engineering and technology*. 2nd ed. 1992: Addison-Wesley Pub. Co.

107. Lindberg, R., *Processes and materials of manufacture*. Allyn and Bacon series in engineering. 1990: Allyn and Bacon.

108. Sanders, M., McCormick, E., *Human Factors in Engineering and Design*. 1993, New York: McGraw-Hill.

109. Gao, W., Gao, S., Liu, Y.. *3D CAD Model Similarity Assessment and Retrieval using DBS*. in *Proceedings of ASME DETC*. 2005. Long Beach, CA.

110. Jawalkar, S., Campbell, M. *Automated Synthesis of MEMS Fabrication Sequences Using Graph Grammars*. in *Proceedings of ASME DETC* 2007. Las Vegas, NV.

111. Koen Lamberts, D.S., ed. *Knowledge, Concepts, and Categories*. 1997, The MIT Press: Cambridge.

112. Bernstein et al. *How Similar is it? Towards Personalized Similarity Measures in Ontologies*. in *Proceedings of the 7th Internationale Tagung Wirtschaftsinformatik*. 2005. Bamberg, Germany.

113. Smith E., E., Osheron, E., E., , ed. *Thinking: An Invitation to Cognitive Science*. Vol. 3. 1995, MIT Press, MA.

114. Dekang, L., *An Information-Theoretic Definition of Similarity*, in *Proceedings of the International Conference on Machine Design*. 1998: Madison, Wisconsin.

115. Herrmann, J.W., Balasubramaniam, S., Singh, G., *Defining Specialized Design Similarity Measures.* International Journal of Production Research, 2000. **38**(15): p. 3603 - 3621.

116. Elinson, A., Nau, D., Regli, W. *Feature-based similarity assessment of solid models*. in *Proceedings of the fourth ACM symposium on Solid modeling and applications* 1997. Atlanta, GA.