

## Developing a multi-layer reference design retrieval technology for knowledge management in engineering design

Yuh-Jen Chen<sup>a,\*</sup>, Yuh-Min Chen<sup>a</sup>, Chin-Bin Wang<sup>b</sup>, Hui-Chuan Chu<sup>c</sup>, Tsung-Nan Tsai<sup>d</sup>

<sup>a</sup>*Institute of Manufacturing Engineering, National Cheng Kung University, 1, Ta-Hsueh Rd., 70101 Tainan, Taiwan, ROC*

<sup>b</sup>*Department of Information Management, Nan Hua University, Chia-Yi, Taiwan, ROC*

<sup>c</sup>*National University of Tainan, Taiwan, ROC*

<sup>d</sup>*Department of Industrial Management, Shu-Te University, Kaohsiung, Taiwan, ROC*

### Abstract

Engineering design is a knowledge-intensive process that encompasses conceptual design, detailed design, engineering analysis, assembly design, process design, and performance evaluation. Each of these tasks involves various areas of knowledge and experience. The sharing of such knowledge and experience is critical to increasing the capacity for developing products and to increasing their quality. It is also critical to reducing the duration and cost of the development cycle. Accordingly, offering engineering designers various methods for retrieving engineering knowledge is one of the most important tasks in managing engineering knowledge.

This study develops a multi-layer reference design retrieval technology for engineering knowledge management to provide engineering designers with easy access to relevant design and related knowledge. The tasks performed in this research include (i) designing a multi-layer reference design retrieval process, (ii) developing techniques associated with multi-layer reference design retrieval technology, and (iii) implementing a multi-layer reference design retrieval mechanism. The retrieval process contains three main phases—‘customer requirement-based reference design retrieval’, ‘functional requirement-based reference design retrieval’ and ‘functional feature-based reference design retrieval’. This technology involves (1) customer requirement-based reference design retrieval, which involves a structured query model for customer requirements, a case-based representation of designed entities, a customer requirement-based index structure for historical design cases, and customer requirement-based case searching, matching and ranking mechanisms, (2) functional requirement-based reference design retrieval, which includes a structured query model for functional requirements, a functional requirement-based index structure for historical design cases, and functional requirement-based case searching, matching and ranking mechanisms, and (3) functional feature-based reference design retrieval, which is a binary code-based representation for functional features, an ART1 neural network for functional feature-based case clustering and functional feature-based case ranking.

© 2005 Elsevier Ltd. All rights reserved.

**Keywords:** Knowledge management; Engineering design; Reference design retrieval; Multi-layer; ART1 neural network

### 1. Introduction

With the advent of the knowledge economy, knowledge has become an important asset of numerous firms in the 21st century. Effectively organizing, storing and sharing a firm's knowledge is critical to that firm's success. Consequently, firms must effectively organize, store and share knowledge to boost business intelligence (BI).

Engineering design, a knowledge-intensive process, includes conceptual design, detailed design, engineering

analysis, assembly design, process design and performance evaluation. Each task is conducted using various areas of knowledge and experience. Effectively organizing, storing and retrieving such knowledge and experience is a major factor in increasing product development capability and quality, and in reducing the development cycle time and cost. Hence, organizing, storing and retrieving product design information, design intent and underlying design knowledge constitute the foundation of engineering knowledge management.

Most traditional knowledge management systems and engineering data management systems provide only general document management, engineering data management and team information management (Chen & Lian, 1999; Chen & Jan, 2000; Chen, Shen, & Shr, in press; Cockshoot, 1996; Homer, Thompson, & Deacon, 2002; O'Leary, 1998;

\* Corresponding author. Tel.: +886 6275 7575; fax: +886 208 5334.  
E-mail address: [denischen@cacerl.ime.ncku.edu](mailto:denischen@cacerl.ime.ncku.edu) (Y.-J. Chen).

Rupple & Harrington, 2001; Wei, Hu, & Chen, 2002). However, no study has yet developed a technology for retrieving engineering knowledge related to engineering design by querying customer requirements, functional requirements and functional features. This task is responsible for a bottleneck in sharing valuable product information and knowledge of engineering design, so a satisfactory engineering knowledge management system has not yet been developed.

This study depicts a multi-layer reference design retrieval technology for engineering knowledge management, which involves customer requirement-based reference design retrieval, functional requirement-based reference design retrieval, and functional feature-based reference design retrieval, to give engineering designers easy access to reference information and knowledge. This study depends on the following tasks—(i) designing a multi-layer reference design retrieval process, (ii) developing techniques for multi-layer reference design retrieval, and (iii) implementing a multi-layer reference design retrieval mechanism. The techniques related to this technology include (1) customer requirement-based reference design retrieval, which involves a structured query model for customer requirements, a case-based representation of designed entities, a customer requirement-based index structure for historical design cases, and customer requirement-based case searching, matching and ranking mechanisms, (2) functional requirement-based reference design retrieval, which involves a structured query model for

functional requirements, a functional requirement-based index structure for historical design cases, and functional requirement-based case searching, matching and ranking mechanisms, and (3) functional feature-based reference design retrieval, which uses a binary code-based representation for functional features, an ART1 neural network for functional feature-based case clustering, and functional feature-based case ranking.

## 2. Scope of research

This section presents an overview of a proposed engineering knowledge management framework, and then considers the importance of knowledge retrieval and its use in engineering design.

### 2.1. Engineering knowledge management framework

Fig. 1 shows the framework as a knowledge management life cycle, which comprises the creation, capture, compilation and storage and retrieval/querying of engineering knowledge. The elements in the knowledge management life cycle are briefly explained below.

(1) *Knowledge creation.* Engineering designers normally apply various methods of structured design to perform and achieve their design objectives. An examination of engineering designer behavior reveals that four structured design methods are often employed; they

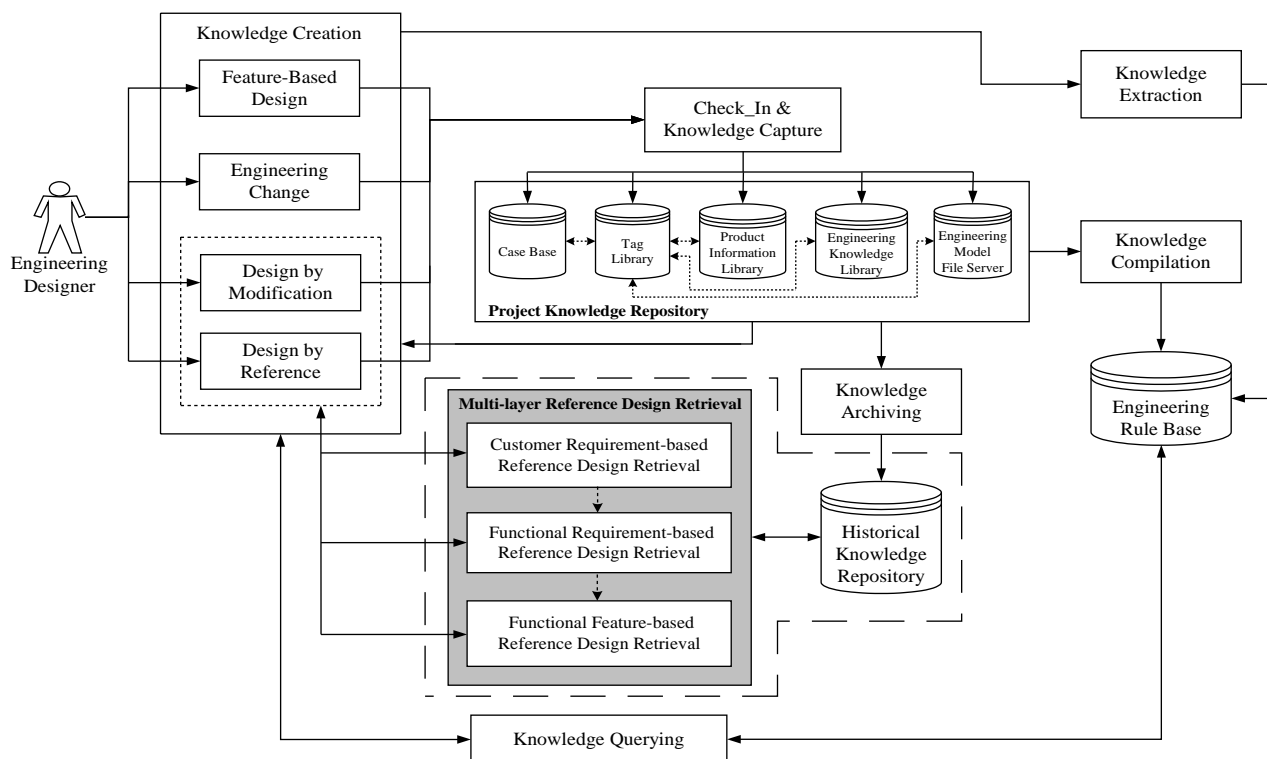


Fig. 1. Engineering knowledge management framework.

are feature-based design, engineering change, design by modification and design by reference.

- (2) *Knowledge extraction, capture, compilation and storage.* Design intent, applied engineering principles and heuristics and information related to engineering collaboration can be extracted during the engineering design process, and associated with the design object as notes for reference. When an engineering model has been completed and checked in-to a project knowledge repository, product information and knowledge on the engineering model are captured and stored in product information and engineering knowledge libraries, respectively. Additionally, the product information and engineering knowledge are compiled in rule format and deposited in an engineering rule base. Once a design project has been completed, the engineering models and associated knowledge are stored in a historical knowledge repository for later use.
- (3) *Knowledge retrieval/Querying.* Product information and engineering knowledge can be retrieved when an engineering model is examined or copied from the project knowledge repository. Similarly, historical engineering models, related product information and engineering knowledge can also be referenced or copied to provide a reference for new projects. Furthermore, engineering designers can conveniently query engineering knowledge using the knowledge query function to solve related design problems.

This study focuses primarily on knowledge retrieval, which is represented by the shaded portion and surrounded by the broken line in Fig. 1.

### 2.2. Importance of knowledge retrieval and its use in engineering design

Engineering design is a knowledge and creativity-intensive process. The execution of each task in a process depends on multifaceted design knowledge and experience. Therefore, knowledge retrieval is required to enable the sharing of engineering knowledge to support engineering knowledge management in engineering design, and to reduce the duration and cost of the product development cycle.

Engineering design (Ertas & Jones, 1993; Pahl, Beitz, & Wallace, 1984) is the systematic process of identifying customer requirements; translating them into the functional requirements of a product, and then mapping these functional requirements into functional features and engineering specifications that can be economically met during manufacture, by exploiting creativity, scientific principles and technical knowledge, as illustrated in Fig. 2.

Therefore, a multi-layer reference design retrieval, which involves customer requirement-based reference design retrieval, functional requirement-based reference design retrieval, and functional feature-based reference

design retrieval was developed herein to support various levels of knowledge retrieval in engineering design. (See Fig. 2)

## 3. Multi-layer reference design retrieval

This section outlines the process of multi-layer reference design retrieval. Fig. 3 shows that this retrieval process consists of three layers, which are (i) the customer requirement-based reference design retrieval layer, (ii) the functional requirement-based reference design retrieval layer and (iii) the functional feature-based reference design retrieval layer. Each layer is detailed below.

### 3.1. Customer requirement-based reference design retrieval layer

The customer requirement-based reference design retrieval layer involves three main steps; these are the analysis and establishment of a customer requirement-based query model; the definition and establishment of index structure for historical design cases, and the searching, matching and ranking of similar design cases. Each step uses several crucial techniques. Techniques associated with the analysis and establishment of a customer requirement-based query model include the semantic analysis and structured representation of customers' requirements. Techniques involved in the definition and establishment of an index structure for historical design cases include the case-based representation of a designed entity and the establishment of an index structure for historical design cases. Finally, the techniques for searching, matching and ranking similar design cases include customer requirement-based case searching, matching and similarity ranking.

*Step 1: Analysis and establishment of customer requirement-based query model.* A customer requirement-based query model is analyzed and established primarily to translate a customer's requirements (in non-structured natural language) described by engineering designers into a structured query model by applying semantic analysis techniques.

Initially, the semantic analysis techniques of sentences breaking, word tagging and syntactic parsing are used to break sentences and retrieve the product name, the component name and the related attribute name from the description of customer requirements. Then, the customer requirement-based query model is established from the structured model of the customer's requirements. Additionally, the weights of the required attributes of the described product and its components are also set for inclusion in the customer requirement-based query model.

*Step 2: Definition and establishment of index structure for historical design cases.* An index structure for

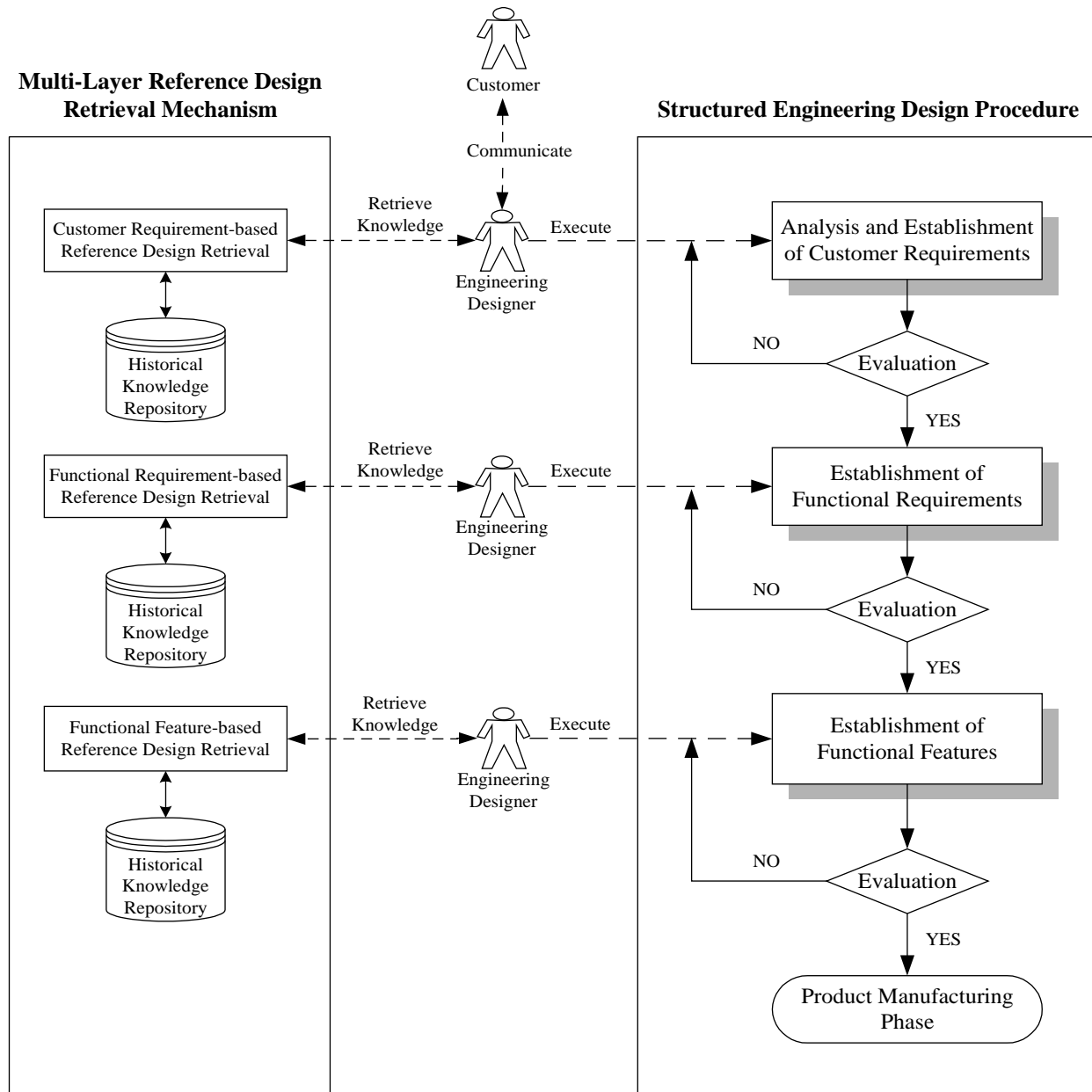


Fig. 2. A multi-layer reference design retrieval in engineering design.

historical design cases is defined and established by experts in engineering design based on the characteristics of the products and the components. After a project has been completed, all engineering data and related knowledge are collected and defined in a set of design cases, and stored in a historical knowledge repository. The index structure of a design case facilitates the searching and matching of similar design cases.

*Step 3: Searching, matching and ranking similar design cases.* Similar historical design cases are sought, matched and retrieved as determined by the customer requirement-based query model. They are then ranked in order of calculated coefficient of similarity. These ranked historical design cases can be used as the direct

references by the engineering designers or can be further refined in the functional requirement-based reference design retrieval layer.

### 3.2. Functional requirement-based reference design retrieval layer

The steps of the functional requirement-based reference design retrieval layer are the establishment of the functional requirement-based query model, the definition and establishment of index structures for historical design cases, and searching, matching and ranking similar design cases. The establishment of a functional requirement-based query

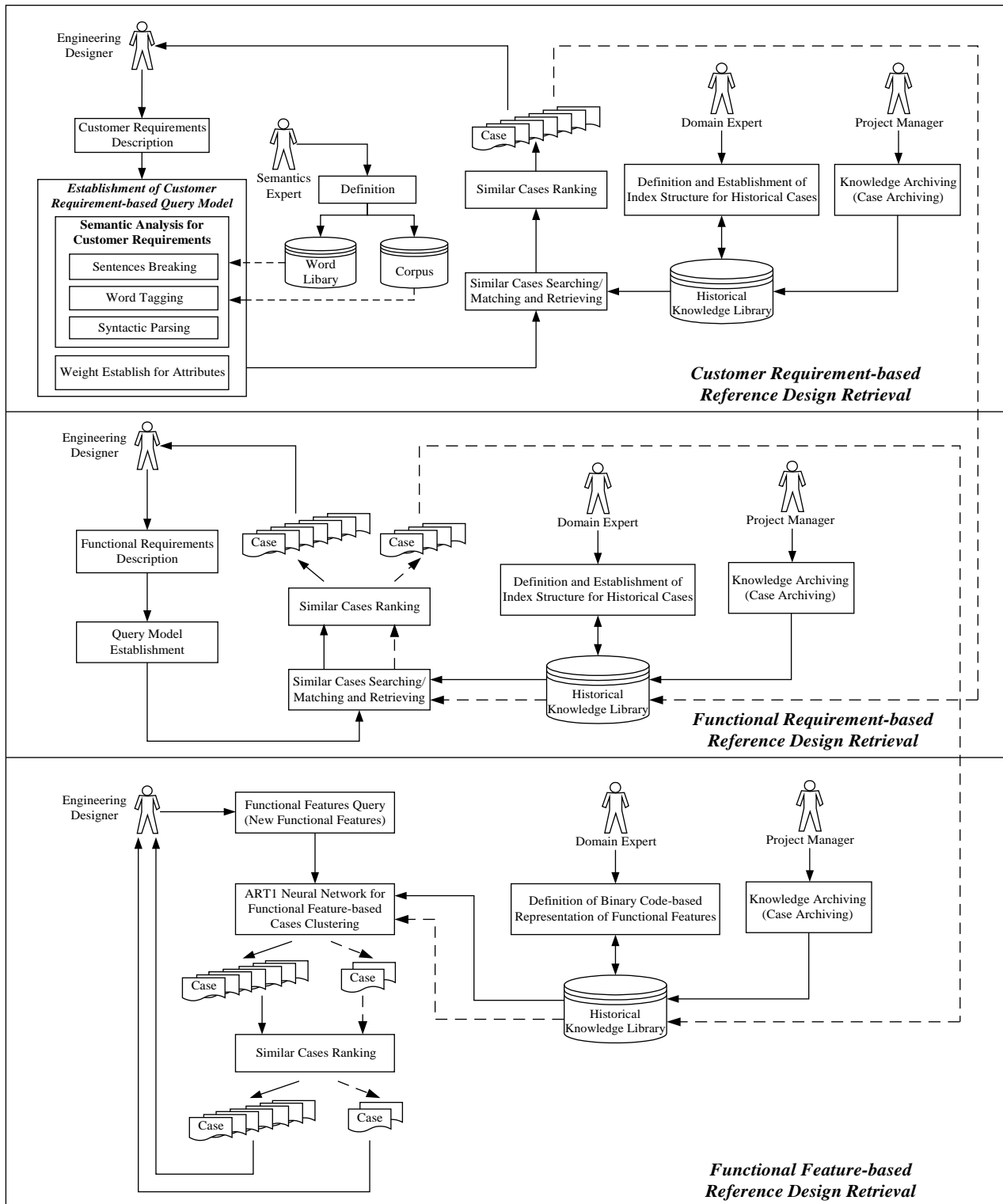


Fig. 3. Multi-layer reference design retrieval.

model involves the definition and the structured representation of the functional requirement. Moreover, the definition and establishment of index structures for historical design cases involve the establishment of index structures of historical design cases. Finally, techniques for

searching, matching and ranking similar historical cases include functional requirement-based case searching, matching and similarity ranking.

*Step 1: Establishment of a functional requirement-based query model.* The functional requirement-based query

model is established to define functional requirements, and to build a structured query model for functional requirements using a structured representation, to facilitate searching and matching similar design cases.

*Step 2: Definition and establishment of index structures for historical design cases.* In this study, index structures for historical design cases are established according to the key elements/terms in the description of functional requirements. The project manager can save completed design cases in the historical knowledge repository using the index structures. These saved design cases are collected into an appropriate set of cases based on the defined index structures. These index structures also facilitates searching and matching similar design cases to give engineering designers a convenient way to find the historical design cases that are most similar to the functional requirement-based query model from the historical knowledge repository.

*Step 3: Searching, matching and ranking similar design cases.* Similar design cases are sought, matched and ranked mainly to retrieve functional requirement-based similar design cases based on the aforementioned functional requirement-based query model and the index structure for historical design cases. Furthermore, the retrieved design cases are ranked in order of calculated similarity to the most valuable reference design. However, these ranked historical design cases can be direct references for used by the engineering designers or can be further refined in the subsequent layer, which is the functional feature-based reference design retrieval layer.

### 3.3. Functional feature-based reference design retrieval layer

The functional feature-based query and reference design retrieval layer involves functional feature representation, ART1 neural network operation, and functional feature-based case retrieval by ART1. The representation of the functional feature involves the techniques of the definition of functional features and the binary code-based representation for functional features. In the operation of the ART1 neural network model, the ART1 characteristics must be identified first, followed by the ART1 architecture and algorithm. The development of a method for retrieving similar cases involves techniques for case clustering and case similarity ranking.

As indicated in Fig. 3, before the retrieval process can be begun, the ART1 neural network must be trained and tested by training and testing historical samples of functional features. During the training of the ART1 neural network, the functional features in a historical design case are represented in binary code. When the training process has been completed, design cases in the historical knowledge repository are clustered according to their functional features. The fact that the design cases are clustered allows

the retrieval process to be initiated. The engineering designer initially specifies a set of functional features that are treated as binary variables. The ART1 neural network for functional feature-based case clustering must then be applied before the actual design cases are acquired, to provide a case classifier for the functional feature query. This query is then processed through the ART1 neural network to obtain similar cases. Before these cases are sent to the engineering designer, these similar design cases are ranked by the calculated coefficients of similarity. The engineering designer then examines the set of ranked cases to obtain useful information and knowledge.

## 4. Customer requirement-based reference design retrieval

The section specifies the techniques of customer requirement-based reference design retrieval. These techniques involves the use of a structured query model for customer requirements, a case-based representation of designed entities, a customer requirement-based index structure for historical design cases, and customer requirement-based case searching, matching and ranking mechanisms. These are detailed below.

### 4.1. Development of customer requirement-based query model

This subsection addresses the development of a customer requirement-based query model. Semantic analysis techniques (Califf & Mooney, 1997; Church, 1998; Eric, 1995) are applied to transform a non-structured description of a customer's requirements into a structured query model for searching and matching reference designs.

This study defines a structured representation of customer requirements using a semantic graph before analyzing and establishing the customer requirement-based query model (Sowa, 1984) (Fig. 4). A structured query model includes (i) the description of the customer's requirements for the product and (ii) the description of the customer's requirements for the product's components. The former includes product-related elements, such as the name of the customer, the name of the requiring verb, the name of the product and the name of the relevant attributes, while the latter includes the component name and the attribute name. Fig. 5 presents an example of a structured representation for describing customer requirements based on the semantic graph, in which the boxes reveal concepts such as entities, attributes, states and events, whereas the ellipses show the interconnection among the concepts.

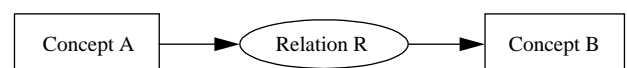


Fig. 4. Basic structure of semantic graphs.



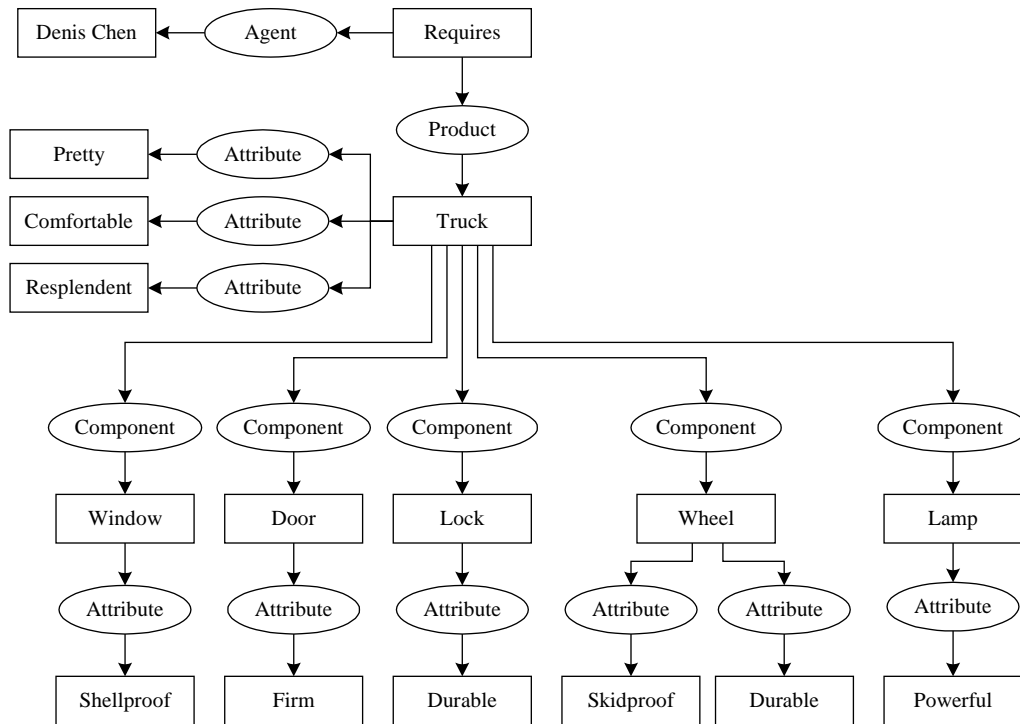


Fig. 5. Example of customer requirement model.

The customer's requirements in the semantic graph in Fig. 5 can be defined as follows

[Proposition:  
 [Requires]-  
 (Agent)→[Denis Chen]  
 (Product)→[Truck]-  
 (Attribute)→[Pretty]  
 (Attribute)→[Comfortable]  
 (Attribute)→[Resplendent]  
 (Component)→[Window]-  
 (Attribute)→[Shellproof]  
 (Component)→[Door]-  
 (Attribute)→[Firm]  
 (Component)→[Lock]-  
 (Attribute)→[Durable]  
 (Component)→[Wheel]-  
 (Attribute)→[Skidproof]  
 (Attribute)→[Durable]  
 (Component)→[Lamp]-  
 (Attribute)→[Powerful]  
 ]

Based on the foregoing structured representation of the customer requirements, the customer requirement-based query model is analyzed and established using semantic analysis techniques and tools.

The customer requirement-based query model is analyzed and established in three steps, the following three steps in the order written

(1) Sentence Breaking (Church, 1998): The customer normally uses natural language to describe his/her

requirements of the product. However, the basic unit of natural language is the sentence, so the sentence breaking techniques of semantic analysis must first be applied to handle the breaking of sentences that describe the customer's requirements.

- (2) Word Tagging (Eric, 1995): The aim of the word tagging is to tag a word in a sentence to determine its grammatical attributes. Each word is no longer just a simple string; the grammatical attributes tagged a word tagged reflect its semantic meaning. Words can be tagged in the following three ways. (i) Tagging of attributes - specifies the grammatical attribute of a word in a sentence. For instance, ⟨NN⟩ represents noun\_singular\_countable/uncountable; ⟨NNS⟩ represents noun\_plural\_countable, and ⟨NNP⟩ represents proper noun. (ii) Tagging of proper nouns - identifies keywords in a sentence. For example, the tag ⟨Term⟩ can be used to tag the name of a product, the name of a component or the name of a customer. (iii) Tagging of dependence - defines the role of a keyword in a sentence, typically following a tagged proper noun. For example, “~O” represents an active keyword and “~S” represents a passive keyword. The mutual dependency of the words can be analyzed using the dependence tags.
- (3) Syntactic Parsing (Califf & Mooney, 1997): Syntactic parsing is conducted mainly to retrieve important words (such as nouns, adjectives, modifiers and others) from the tagged word in each sentence by matching the defined requirement representation templates (shown in Fig. 6) for the customer requirement-based query model in the corpus library. These retrieved words can be

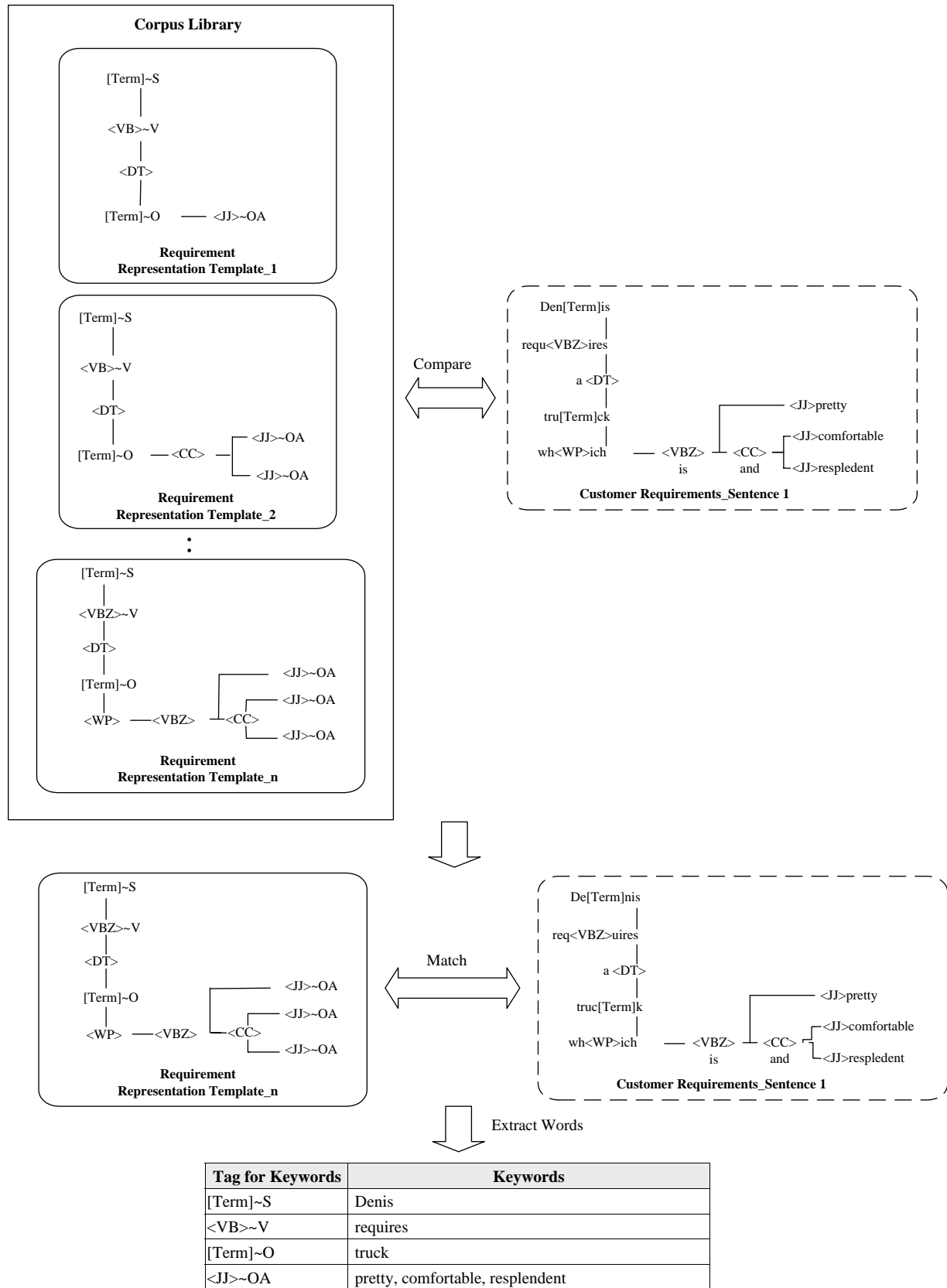


Fig. 6. Steps in syntactic parsing.



classified by attribute; a noun is treated as a Term; an adjective as the attribute of a Term, and a modifier as the weight of an attribute. This template for representing requirements is a regular semantic representation model, which was defined and developed by engineering design experts by analyzing and managing models of customer requirements, in a manner consistent with grammatical English sentences.

An illustrative example is provided to explain in detail the analysis and establishment of the customer requirement-based query model

**Example.** : Denis requires a truck, which is pretty, comfortable, and resplendent. The window, door, lock, wheel, and lamp of the truck are my emphasis items. The window must be shellproof, while the door must be firm. Furthermore, the lock is more durable. Finally, the wheel is very skid proof and durable.

*Step 1: Sentence Breaking.* Breaking the sentences that describe the above customer requirement in natural language yields the following five sentences.

Sentence\_1: Denis requires a truck, which is pretty, comfortable, and resplendent.

Sentence\_2: The window, door, lock, wheel, and lamp of the truck are my emphasis items.

Sentence\_3: The window must be shellproof, while the door must be firm.

Sentence\_4: Furthermore, the lock is more durable.

Sentence\_5: Finally, the wheel is very skid proof and durable.

*Step 2: Word tagging.* Consider Sentence\_1 as an example. The words are tagged as follows.

*Step 2.1: Tagging Attribute.* Tag all of the words in Sentence\_1, yielding the following, Denis<NNP> requires<VBZ> a<DT> truck<NNP>, which<WP> is<VBZ> pretty<JJ>, comfortable<JJ> and<CC> resplendent<JJ>.

*Step 2.2: Tagging Proper Noun.* Tag the proper noun according to the results obtained from Step 2.1. The results are as follows. Denis<NNP> [Term] requires<VBZ> a<DT> truck<NNP>[Term], which<WP> is<VBZ> pretty<JJ>, comfortable <JJ> and<CC> resplendent<JJ>.

*Step 3: Syntactic Parsing.* Retrieve keywords from the description of the customer's requirements, based on the defined requirement representation template in the corpus library and the tag of dependence in the requirement template. Fig. 6 depicts the syntactic parsing of Sentence\_1 as an example.

After the sentences in the above description of the customer's requirements have been syntactically parsed, the keywords are retrieved from each sentence and gathered to establish a table of keywords and relevant terms, as indicated in Table 1.

Table 1  
Set of keywords and relevant terms

Categories of keywords	Keywords
Customer	Denis
Verb	requires
Product	truck
Product Attributes	pretty, comfortable, resplendent
Component	window, door, lock, wheel, lamp
Component Attributes	shellproof (window), firm (door), skid proof (wheel), durable (wheel), powerful (lamp)

Weights are assigned to the required attributes of the product or component to give the customer a choice of requirements and specify those requirements. A list of weights is made up of integers from 1 to 10 and is provided to the customer as a reference for the importance of the requirements, based on the retrieved attributes of the products and components in Table 1. A larger value indicates greater importance. For example,  $w_{\text{pretty}} = 8$  and  $w_{\text{comfortable}} = 8$ .

#### 4.2. Case-based representation of a designed entity

A case-based representation of a designed entity is proposed to record and organize related product information and engineering knowledge about a designed entity and help engineering designers to access them easily and quickly. In Fig. 7, a 'case' is viewed as a box that contains tags and links product information and engineering knowledge about a design entity (such as an engineering model). The scheme of a case consists of three features—the case feature, the model feature and the semantic feature. The case feature defines the case data, including the case name, case ID, tag ID, tag name, model creator, contributor, date, language, version and location. The model feature indicates the tag for product information that records the details of a design entity, including customer requirements, functional requirements and functional features. Finally, the semantic feature represents the tags for engineering knowledge that also record the design knowledge and experience of engineering designers. These tags for engineering knowledge are classified into three categories—(i) the tag for feature-based design, (ii) the tag for engineering change, and (iii) the tag for design by modification/reference. Each of these tags points to relevant product information or engineering knowledge.

#### 4.3. Definition and establishment of index structure for historical design cases

The index structure for historical design cases is built according to the characteristics of the product structure (Fig. 8), to facilitate the searching and matching similar design cases in terms of the 'product name' and 'component

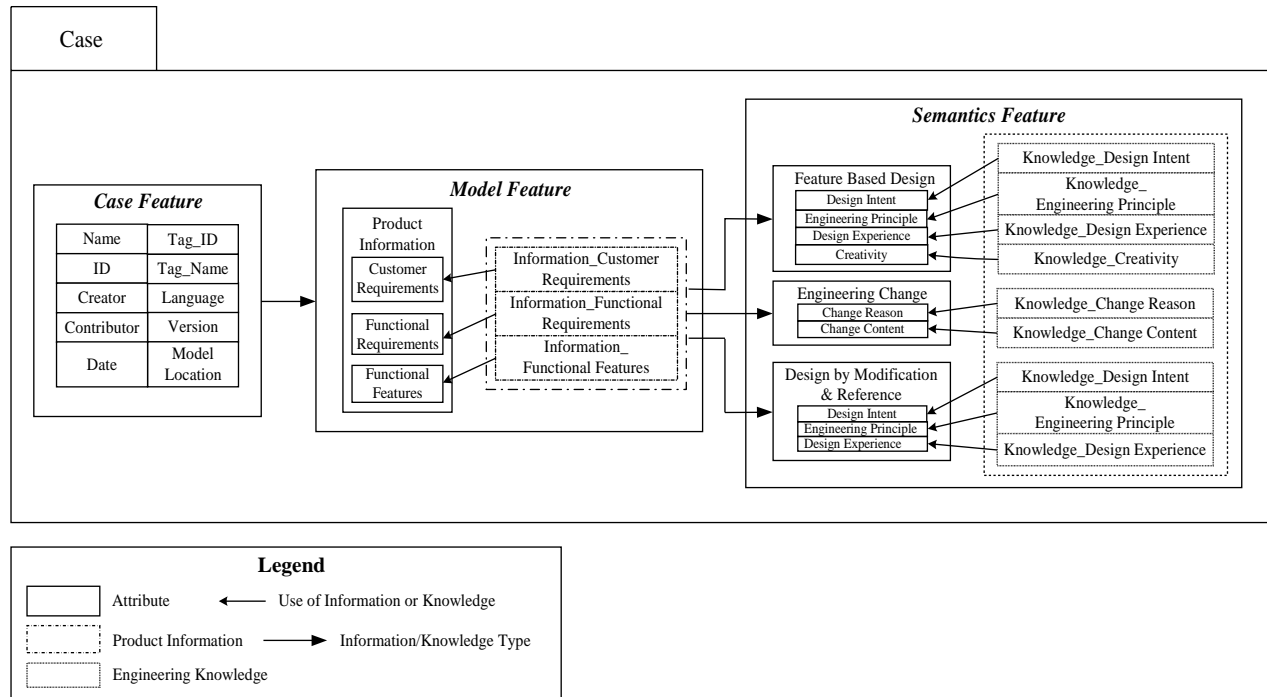


Fig. 7. Conceptual case model.

name' in the customer requirement-based query model. The index structure for historical design cases has four layers—(i) the product layer, (ii) the component layer, (iii) the component synonym layer and (iv) the case layer.

As well as elucidating the index structure for historical design cases, as stated above, this study also designs a structure for the required attributes of the product and components, according to the characteristics of those attributes, to identify the most similar design cases. Fig. 9 indicates that the structure has three layers for matching attributes - (i) the linguistic variable class layer, (ii) the linguistic variable layer and (iii) the attribute layer.

#### 4.4. Searching, matching and ranking similar design cases

This study proposes a method for similarity matching with semantic ambiguity to identify the most valuable design cases in the historical knowledge repository, and thereby to find the semantic similarities between the customer requirement-based query model and historical structured models of customer requirements, and thus to identify the most similar design cases.

The similarity matching process that handles semantic ambiguity is proposed based on the structured model for customer requirement-based query defined in Section 4.1.

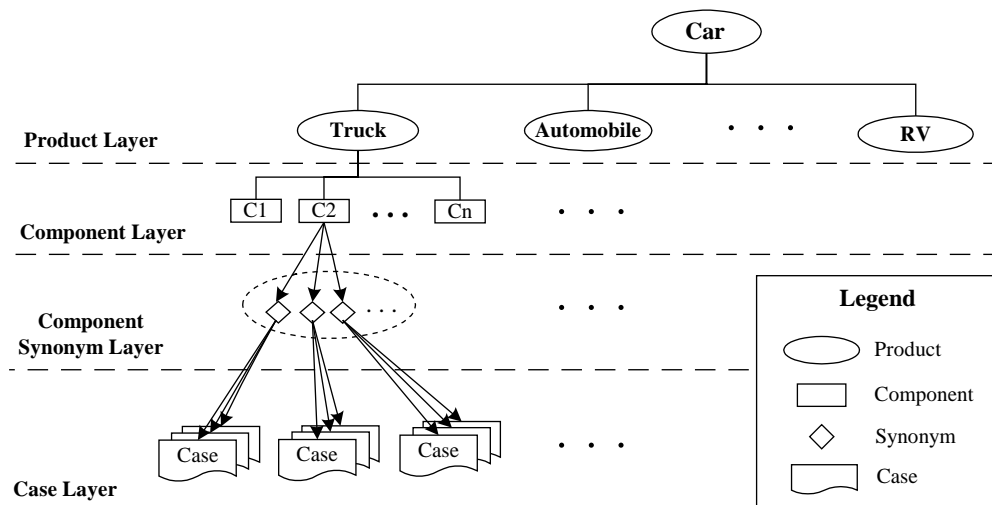


Fig. 8. Index structure of historical design cases.

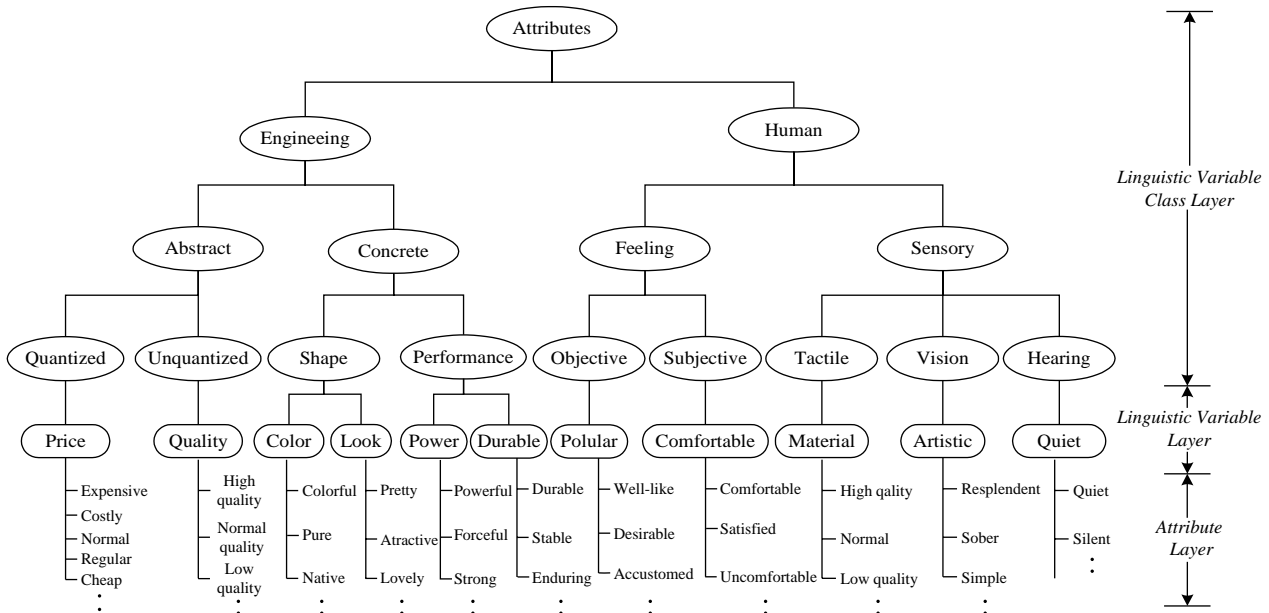


Fig. 9. Structure for matching attributes.

The process has two phases—(i) similarity matching of proper nouns and (ii) similarity matching of the attributes of proper nouns. The latter includes three steps. They are (i) similarity matching of terms of attributes, (ii) similarity matching of semantics of attributes and (iii) calculation of total similarity for related attributes of a proper noun. Finally, the total similarity in a design case is calculated to deal with the customer requirement-based case ranking.

#### 4.4.1. Similarity matching of proper nouns

The similarity matching of proper nouns involves searching and matching similar design cases according to the product name and component names in the customer requirement-based query model. Accordingly, similar design cases are identified. The method involves the Boolean

operation to filter out design cases with the same product name and similar component names in the customer requirement-based query model, according to the proper nouns (such as the product term or the component term) that appear in the customer requirement-based query model and through the above index structure for historical design cases. An example is presented below.

**Example.** : The product name and component name in the customer requirement-based query model are as follows.

(Product Term) → Truck

(Component Term) → Lamp Window

First, given the description of the product name “Truck”, that part of the index structure historical design cases with the product name “Truck” is identified (Fig. 10). Next,

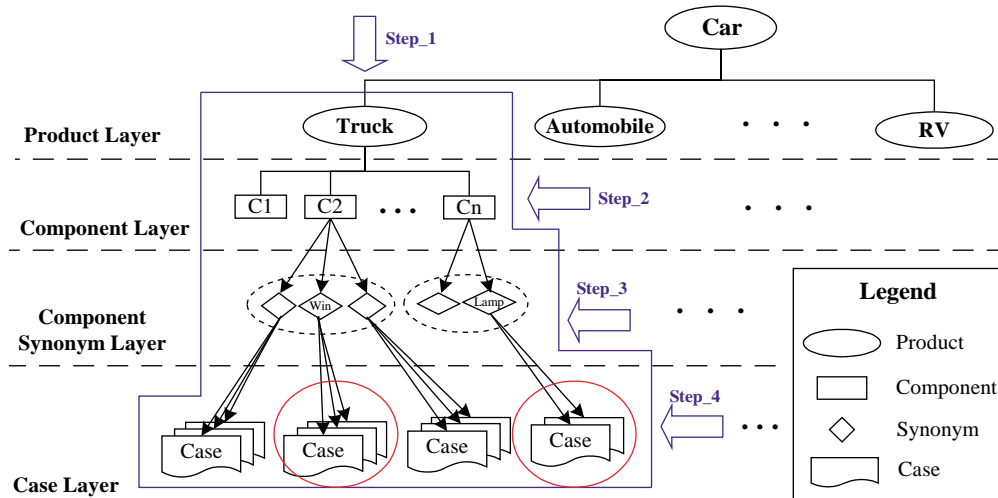


Fig. 10. Steps for searching and matching historical design cases based on product and component names.

based on that part of the index structure with the product name “Truck”, a set of synonyms for the component name, defined in both the component layer and the component synonym layer, is sought, and the component names “Lamp” and “Window” are matched. Hence, design cases with the product name “Truck” and the component name “Lamp”, and with the product name “Truck” and the component name “Window” are identified. The following sets of cases are identified.

$S1 = \{Product\ Term = Truck \ \& \ Component\ Term = Lamp\} = \{Case\_4, Case\_12, Case\_33\}$

$S2 = \{Product\ Term = Truck \ \& \ Component\ Term = Window\} = \{Case\_9, Case\_12, Case\_24, Case\_33, Case\_54\}$

The most similar design cases are identified by applying the Boolean intersection operation, as described below.

$Result = \{S1 \cap S2\} = \{Case\_12, Case\_33\}$

However, if the intersection is a null set, then the Boolean (intersection) operation must be separately applied to the component term “Lamp” in the design cases (Case\_4, Case\_12 and Case\_33); the synonym of the component term “Window” in the design cases; the component term “Window” included in the design cases (Case\_9, Case\_12, Case\_24, Case\_33 and Case\_54), and the synonym of the component term “Lamp” included in the design cases. The results thus obtained are considered to be the most similar design cases.

#### 4.4.2. Similarity matching of attributes of proper nouns

The most similar design cases are identified by the “product name” and “component name”, using the aforementioned similarity matching for proper nouns. In this section, further similarity matching is performed based on the described attributes of the product and the components in the identified design cases. In relation to the ambiguous representation of natural language, the similarity matching of the attributes of proper nouns involves the following three steps—(i) similarity matching of the term of the attribute, (ii) similarity matching of the semantics of the attribute and (iii) calculation of the total similarity of the attributes of a proper noun. These above three steps in the similarity matching of attributes of proper nouns are elucidated below.

- (i) *Similarity Matching of the Term of the Attribute.* In natural language, different people use different words to express the same meaning. Therefore, many terms have the same meaning. Similarity matching of the terms of attributes is therefore used to compare the attributes of a product and the components associated with the customer requirement-based query model with the customer requirements in design cases. This method sets the values of the similarity between

the terms, according to the defined structure for matching attributes (Fig. 9); if the terms are exactly the same, then the similarity value would be 1; if they are of the same class and have the same synonym, then the similarity would be 0.8; if they are in different classes but have the same parent class, then the similarity value is 0.6; the similarity values fall slowly in this way. Therefore, after the term similarities of the described attributes of the product and components in the obtained design cases have been compared, the term similarity between the described attribute in each design case and the product/component attributes in the customer requirement-based query model are determined.

- (ii) *Similarity Matching of the Semantics of the Attribute.* After the attribute term has been similarity-matched, the similarity of each attribute is calculated. This calculation concerns the similarity matching of attributes without considering the difference between the semantic degrees in the customer’s requirements. This difference is neglected, because even if the same term is used to express the same need, the concept of strength of the requirement may differ in individual cases. The similarity matching of the semantics of the attributes focuses on the difference between the customers’ required degrees. The end of Section 4 presents a list of weights in the form of integers from 1 to 10, which represents the different degrees of attributes of the product and components required by customers. A higher value indicates the customer most strongly requires the attribute. The list of weights includes three different fuzzy sets (Fig. 11) to enable the similarities of the strengths of the requirements attributes of the product and components to be easily determined. Weights in the same fuzzy set represent strong similarity; in other cases, similarity is calculated from the distance between weight values (Eq. (1)). An illustrative example is given as follows.

Similarity for the strength of the Requirement

$$\text{of Attribute} \times \frac{10 - (\text{Distance between Values})}{10} \quad (1)$$

**Example:** In the customer requirement-based query model, the value of the strength of the requirement of the attribute “pretty” of the truck is 2. In design Case\_1, the value of the strength of the requirement of the attribute “pretty” of the truck is 3, and in Case\_2,



Fig. 11. Fuzzy set for the strength of the requirement of an attribute.

the value of the strength of the requirement of the attribute “pretty” of the truck is 5.

The values of the strengths of the requirement of attributes 3 and 2 are in the same fuzzy set [0,3]. Therefore, the similarity values should be 1. However, the values of the strengths of the requirements 5 and 2 are not in the same fuzzy set, so the distance between 5 and 2 (5-2=3) is calculated, and the similarity is

$$\frac{10 - |5 - 2|}{10} = 0.7$$

Consequently, the similarity in Case\_1 exceeds that in Case\_2 according to the customer requirement-based query model.

- (iii) *Calculation of Total Similarity of Related Attributes of a Proper Noun.* After (i) the term of the attribute and (ii) the semantics of the attribute have been similarity-matched, the next step is to calculate the total similarity of related attributes of the product/component in the obtained design cases. Meanwhile, the equation for the total similarity of related attributes of the product/components is defined as follows.

$$\text{Similarity} = \sum_{i=1}^k \text{MAXS}_{(ai,bj)} \times \mu_i \quad (2)$$

where,

$\text{MAXS}_{(ai,bj)}$ , the term similarity of attributes  $ai$  and  $bj$ ,  
 $\mu_i$ , the semantic similarity of attributes  $ai$  and  $bj$ ,  
 $k$ , the number of attributes of a term.

#### 4.4.3. Calculation of total similarity in a design case

Section 6.2 presents the total similarity of attributes of a product (Similarity\_PA) and the total similarity of attributes of components (Similarity\_CA) in the obtained design cases. In this subsection, the total similarity in each individual design case is obtained by determining the total similarities of related attributes of a product and components. The total similarity calculation in a design case is defined as follows.

Total Similarity of a Design Case

$$= \text{Similarity PA} + \sum_{i=1}^n \text{Similarity CA}_i \quad (3)$$

The similarities in the obtained design cases are ranked according to the total similarity in an individual design case, to give engineering designers the possible design reference.

## 5. Functional requirement-based reference design retrieval

The section describes the techniques involved in the functional requirement-based reference design retrieval

layer. They are a structured query model for functional requirements, a functional requirement-based index structure for historical design cases, and functional requirement-based case searching, matching and ranking mechanisms.

### 5.1. Establishment of functional requirement-based query model

#### 5.1.1. Definition of functional requirement

This subsection defines functional requirement. A functional requirement in engineering design can be defined as “Adverb” + “Verb” + “Noun”. “Verb + Noun” is called the functional entity and “Adverb” is the modifier of the functional entity, such as “tightly combine pen-nibs” and “easily change ink-reservoir”. Additionally, types of functional design and functional decomposition are defined to provide engineering designers further descriptions of functional type. The type of functional design incorporates basic function, operation function, need function and patch function, while the type of functional decomposition includes main function, interdependent function and auxiliary function.

#### 5.1.2. Representation of functional requirement

This study adopts semantic graph theory (Fig. 4) as a foundation for developing a structured representation of functional requirements based on the definition. Fig. 12 presents an example of a structured representation of the functional requirements obtained from semantic graph theory. This structured query model has three parts—(i) the representation of the functional entity, (ii) the representation of the modifier of the functional entity, and (iii) the representation of the function type. The boxes depict concepts such as entities, attributes, states and events, and the ellipse represents the interconnection among the concepts.

Based on the semantic graph representation for functional requirement in Fig. 12, the functional requirement is defined as follows.

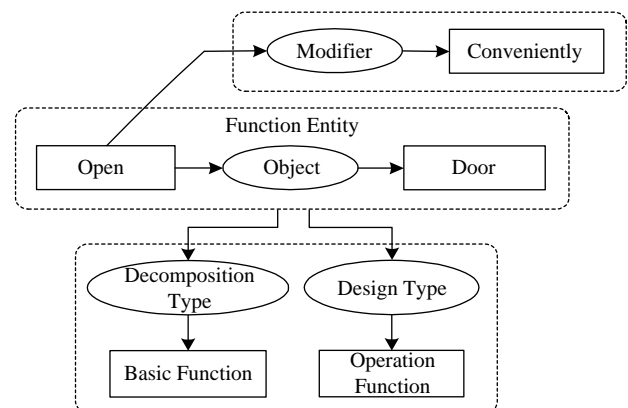


Fig. 12. Structured representation of functional requirement (an example).

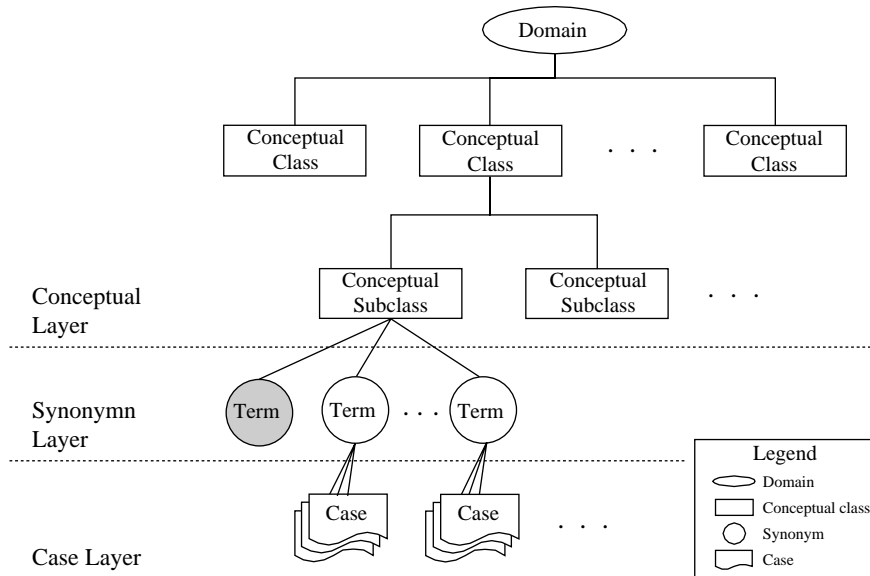


Fig. 13. Prototype index structure of historical design cases.

[Proposition:  
 [Open]-  
 (Object) → [Door]  
 (Modifier) → [Conveniently]  
 (Decomposition Type) → [Basic Function]  
 (Design Type) → [Operation Function]  
 ]

## 5.2. Definition and establishment of index structures for historical design cases

Based on the case-based representation of a designed entity presented in Section 4.2, the index structures for

historical design cases are defined and established to support searching and matching similar design cases in the functional requirement-based reference design retrieval layer. Therefore, five index structures are established according to the characteristics of key elements/terms in the functional requirement and the defined prototype of the index structure for historical design cases (Fig. 13). These five index structures are (i) the index structure for nouns, (ii) the index structure for verbs, (iii) the index structure for adverbs, (iv) the index structure for design functions and (v) the index structure for decomposition functions. The defined prototype of the index structure for historical design cases comprises three layers—(i) a conceptual layer—an abstract set for describing the classified concept, (ii) a synonym

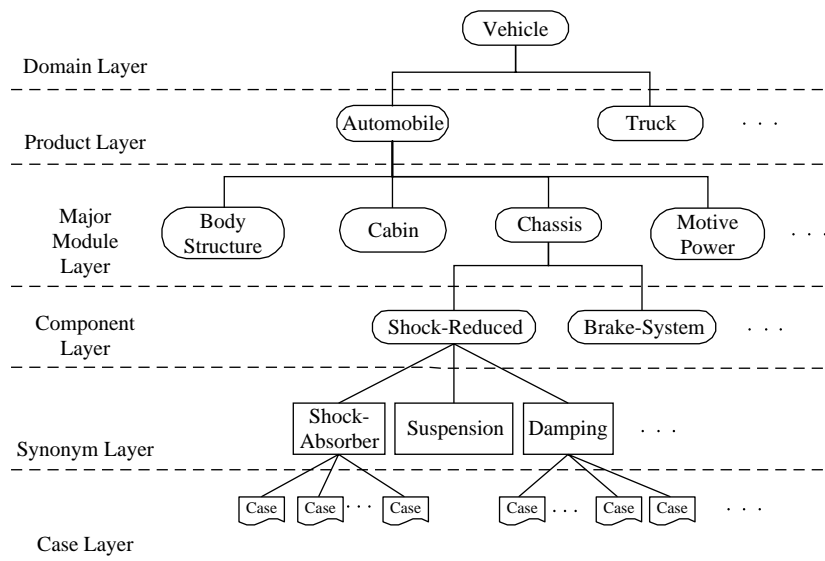


Fig. 14. Index structure for nouns (an example).



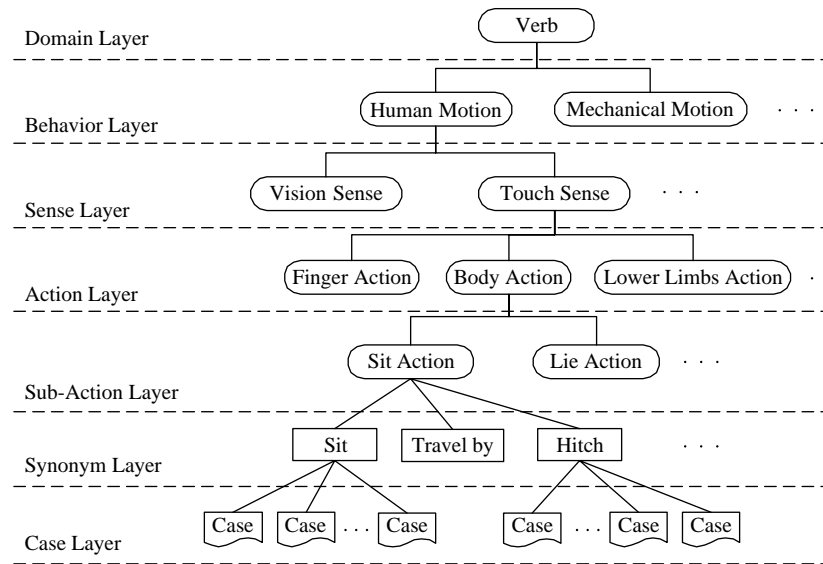


Fig. 15. Index structure for verbs (an example).

layer—a set of synonyms for describing a particular abstract concept, and (iii) a case layer—a set of design cases for recording the same elements/terms.

- (i) *Index structure for nouns.* The purpose of the index structure for nouns is to help the system to identify similar design cases based on the element “Noun” in the functional requirement-based query model. As depicted in Fig. 14, the index structure for nouns has six layers, which are (from top to bottom), (i) the domain layer, (ii) the product layer, (iii) the major module layer, (iv) the component layer, (v) the synonym layer and (vi) the case layer.
- (ii) *Index Structure for Verbs.* The index structure for verbs helps the system to find similar design cases based on the element “Verb” in the functional requirement-based query model. As indicated in

Fig. 15, the index structure for verbs has seven layers, which are in order, (i) the domain layer, (ii) the behavior layer, (iii) the sense layer, (iv) the action layer, (v) the sub-action layer, (vi) the synonym layer and (vii) the case layer.

- (iii) *Index Structure for Adverbs.* The main purpose of building the index structure for adverbs is to find similar design cases that have the same element “Adverb” in the functional requirement-based query model. As illustrated in Fig. 16, the index structure for adverbs has six layers, which are (i) the domain layer, (ii) the modifier layer, (iii) the sense layer, (iv) the sub-sense layer, (v) the synonym layer and (vi) the case layer.
- (iv) *Index Structure for Design Functions.* As shown in Fig. 17, the index structure for design functions has two layers - (i) the design-type layer and (ii) the case

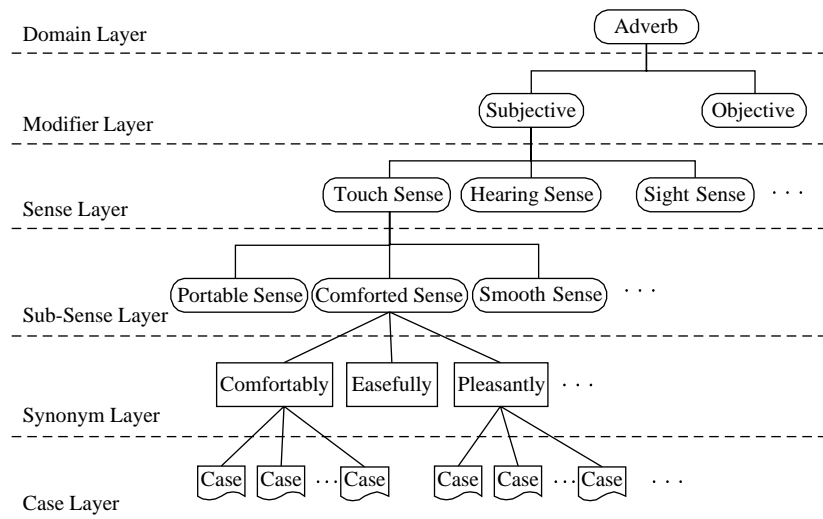


Fig. 16. Index structure for adverbs (an example).



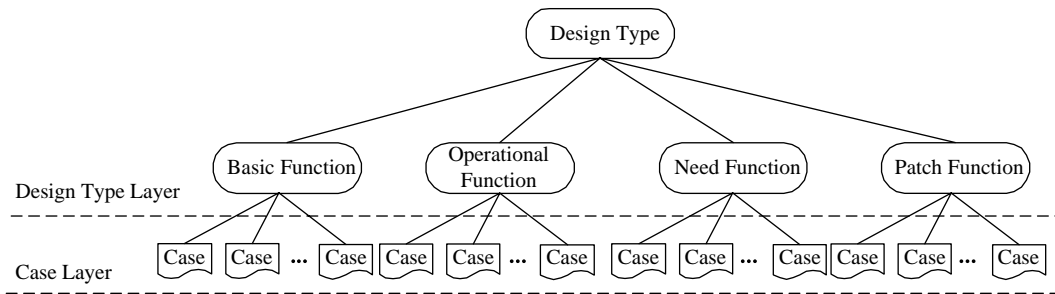


Fig. 17. Index structure for design functions.

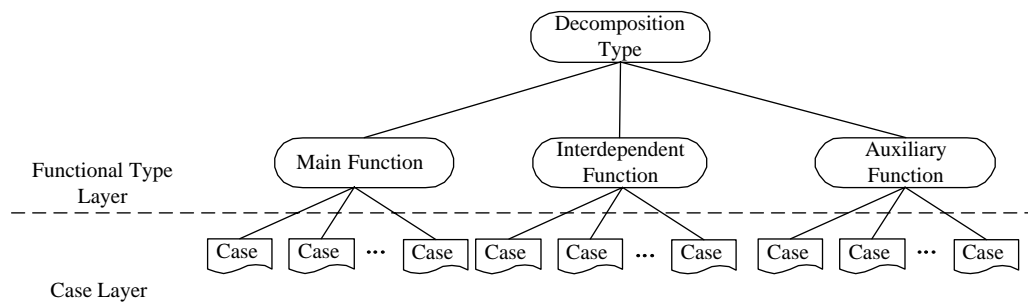


Fig. 18. Index structure for decomposition functions.

layer. The design-type layer contains the types of basic function, operation function, need function and patch function.

- (v) *Index structure for decomposition functions.* The index structure for decomposition functions is similar to that for design functions. It also has a pair of layers in the order, (i) the decomposition-type layer and (ii) the case layer. The decomposition type layer has three types of function—main function, interdependent function and auxiliary function, as presented in Fig. 18.

### 5.3. Searching, matching and ranking similar design cases

This subsection establishes a structured index method using the concept of a complete tree structure of the K-ary Tree (Dongwook, Hyuncheol, & Honglan, 1998; Lee Yoo, Yoon, & Berra, 1996) to identify rapidly the most valuable design cases as references according to the functional requirement-based query model. The primary method is to apply the UID (Unique Element Identifiers) equation to reduce the number of index entries during the search and thus increase the efficiency of the search for similar design cases. Therefore, this subsection firstly explains the establishment of this structured index. Then, the searching, matching and ranking of similar design cases are developed based on the established structured index.

#### 5.3.1. Establishment of structured index

Based on the five index structures for the historical design cases, established in Section 5.2, every node in each

tree structure is visited and assigned an UID value using the K-ary Tree method. UID was established using the top-to-bottom mode, with the value of the parent node in the highest layer taken as UID=1. The UID values of the child nodes are determined according to the UID value of the parent node and the branch locations of its child nodes. The equation for the UID value of a child node is defined as follows.

$$\text{Child}(i,j) = k(i-1) + j + 1 \quad (4)$$

where  $i$  is the UID value of the parent node;  $j$  ( $1 \leq j \leq k$ ) represents the branch location of the child node under its parent node, and  $k$  is the largest of the branches in the tree structure.

As shown in Fig. 19, and as in the example of the index structure of the nouns, the largest branch number is 3; the UID value of the node {Chassis} is 1; the UID value of its child node {Shock-Reduced} is  $3(1-1) + 1 + 1 = 2$ ,

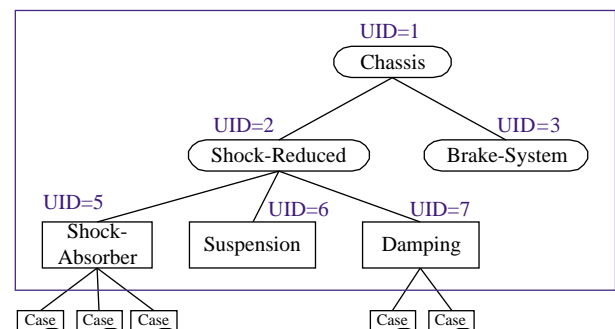


Fig. 19. K-ary tree (example of index structure for nouns).

Table 2  
Calculated UID Values

Node	UID value	Node	UID value
Chassis	1	Shock-reduced	2
Brake system	3	Shock-absorber	5
Suspension	6	Damping	7

and the UID value of the first child node <Shock-Absorber> under the node <Shock-Reduced> is  $3(2-1)+1+1=5$ . Accordingly, all of the UID values of all the nodes are determined, as presented in Table 2.

### 5.3.2. Searching and matching similar design cases

Similar design cases are sought and matched mainly to retrieve similar design cases through the established index structure for historical design cases, based on the described key elements/terms in the functional requirement-based query model.

‘Searching similar design cases’ involves two searching modes - the accurate searching mode and the similar searching mode, which are detailed below.

**5.3.2.1. Accurate searching mode.** The accurate searching mode is used to find the matching node in the index structure based on the key element/term in the functional requirement-based query model, and to determine a set of design cases with the same UID value as that of the matching node. Given the example in Fig. 14, if the node <Shock

Absorber> is exactly the described noun in the functional requirement-based query model and its UID value is 5, then a set of design cases with the same UID value of 5 is obtained; this set includes cases 2, 11 and 19, as shown in Fig. 20.

**5.3.2.2. Similar searching mode.** The similar searching mode is triggered by matching a node that does not record any case based on the key element/term in the functional requirement-based query model. Then, the similar searching mode traces back to the parent node of the matching node by calculating the UID value of the parent node Eq. (5). The UID values of the child nodes of the parent node are determined from the calculated UID value of the parent node Eq. (4). Finally, similar design cases are found through the obtained UID values of the child nodes. The equation for the UID value of a parent node is as follows.

$$\text{Parent}(i) = [(i - 2)/k + 1] \quad (5)$$

In Eq. (5),  $i$  represents the UID value of the node itself while  $k$  represents the largest number of branches in the tree structure.

As in the example presented in Fig. 14, if the conditional noun in the functional requirement-based query model is “Suspension” and this node <Suspension> in the index structure for nouns has no set of design cases, then the UID value of its parent node “Shock-Reduced” ( $((6-2)/3+1=2)$ ) is obtained by utilizing the UID value of the node

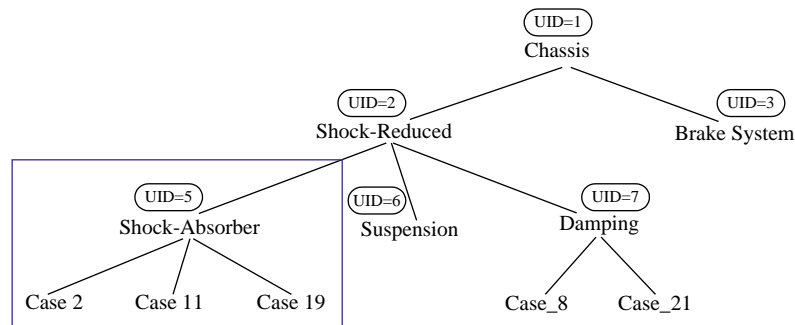


Fig. 20. Accurate searching (example of index structure for nouns).

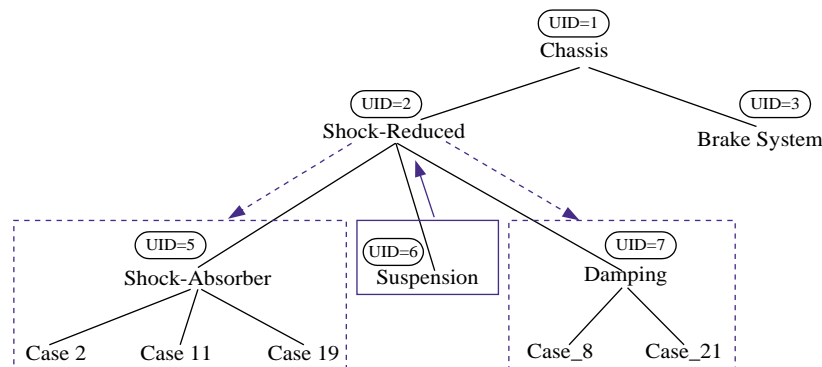


Fig. 21. Similar searching (example of index structure for nouns)

(Suspension) (UID=6) and Eq. (5). The other UID values for nodes that are synonymous with “Suspension” ((Shock Absorber)\_UID=5 and (Damping)\_UID=7) are determined through the UID value of the parent node “Shock-Reduced” (UID=2) and Eq. (4). Finally, the similar design cases that record UID=5 or UID=7 are found. They are Case 2, Case 11, Case 19, Case 8 and Case 21, as depicted in Fig. 21.

The foregoing searching modes can be represented as an algorithm written in C language, as shown in Fig. 22.

```
int times,no,num,maxLEVEL,totalnode,savedno,answerno;
float fatherUID;

for(no=1;no=totalnode;no++)
{
    if(word==table[Node][no])
    {
        ROW=no;
        tempUID[savedno]=table[UID][no];
        break;
    }
}

if(table[CASEid][ROW]==null)
{
    savedno=savedno-1;
    fatherUID=(table[UID][ROW]-1)/k+1
    for(num=1;num=maxUID;num++)
    {
        if((num<=fatherUID)&(fatherUID<num+1))
        {
            fatherUID=num;
            for(no=1;no=totalnode;no++)
            {
                if(table[UID][no]==fatherUID)
                {
                    ROW=no;
                    break;
                }
            }
        }
    }
}

if(table[CHILdno][ROW]!=0)
{
    for(times=1;times=table[CHILdno][ROW];times++)
    {
        tempUID[savedno]=k*(fatherUID+1)+times+1;
        savedno=savedno+1;
    }
}

for(no=1;no=savedno;no++)
{
    for(times=1;times=totalnode;times++)
    {
        if((tempUID[savedno]==table[UID][times])&&(table[CASEid][times]))
        {
            for(answerno=1;answerno;answerno++)
            {
                answerCASEid[answerno]=table[CASEid][times];
            }
        }
    }
}
```

Fig. 22. Algorithm for searching design cases.

Based on the results of a search for similar design cases, the Boolean operation is applied to filter out most similar design cases. Therefore, this study defines five ways to sift out the most similar design cases in the following order of priority. The former four types ((i), (ii), (iii) and (iv)) are combined to perform the intersection operation.

- (i)  $\{V\_CS\} \cap \{N\_CS\} \cap \{A\_CS\} \cap \{DES\_CS\} \cap \{DEC\_CS\}$
- (ii)  $\{V\_CS\} \cap \{N\_CS\} \cap \{A\_CS\} \cap \{DES\_CS\}$  or  $\{V\_CS\} \cap \{N\_CS\} \cap \{A\_CS\} \cap \{DEC\_CS\}$
- (iii)  $\{V\_CS\} \cap \{N\_CS\} \cap \{A\_CS\}$
- (iv)  $\{V\_CS\} \cap \{N\_CS\}$
- (v)  $\{N\_CS\}$

where

“V\_CS” represents a set of design cases that are most similar in terms of “Verb”,

“N\_CS” represents a set of design cases that are most similar in terms of “Noun”,

“A\_CS” represents a set of design cases that are most similar in terms of “Adverb”,

“DES\_CS” represents a set of design cases that are most similar in terms of “Design Type”, and

“DEC\_CS” represents a set of design cases that are most similar in terms of “Decomposition Type”.

For the five defined types used to filter out similar design cases, the design cases obtained using the first type of combination in performing the intersection operation are the most similar to the functional requirement-based query model. If the result of the intersection operation with the first type of combination is a null set, then the second type of combination is employed to perform the intersection operation, yielding design cases that are the second-most similar to the functional requirement-based query model. In this way, the intersection operation is performed continuously. Additionally, if the result obtained using the former four types of combinations in the intersection operation is a null set, then the fifth type for sifting out similar design cases is employed and the obtained design cases are regarded as the most similar design cases.

### 5.3.3. Ranking similar design cases

The obtained similar design cases considered in the previous subsection, are ranked by calculating the degree of similarity with the functional requirement-based query model to give engineering designers the best possible reference.

The calculation performed to rank similar design cases uses mainly the weight equation for various elements/terms (verbs, nouns, adverbs, design types and decomposition types) in the design case. Their weights are thus determined in their own index structure for historical design cases. The weight value can be calculated in one of two modes—accurate searching and similar searching, as discussed in

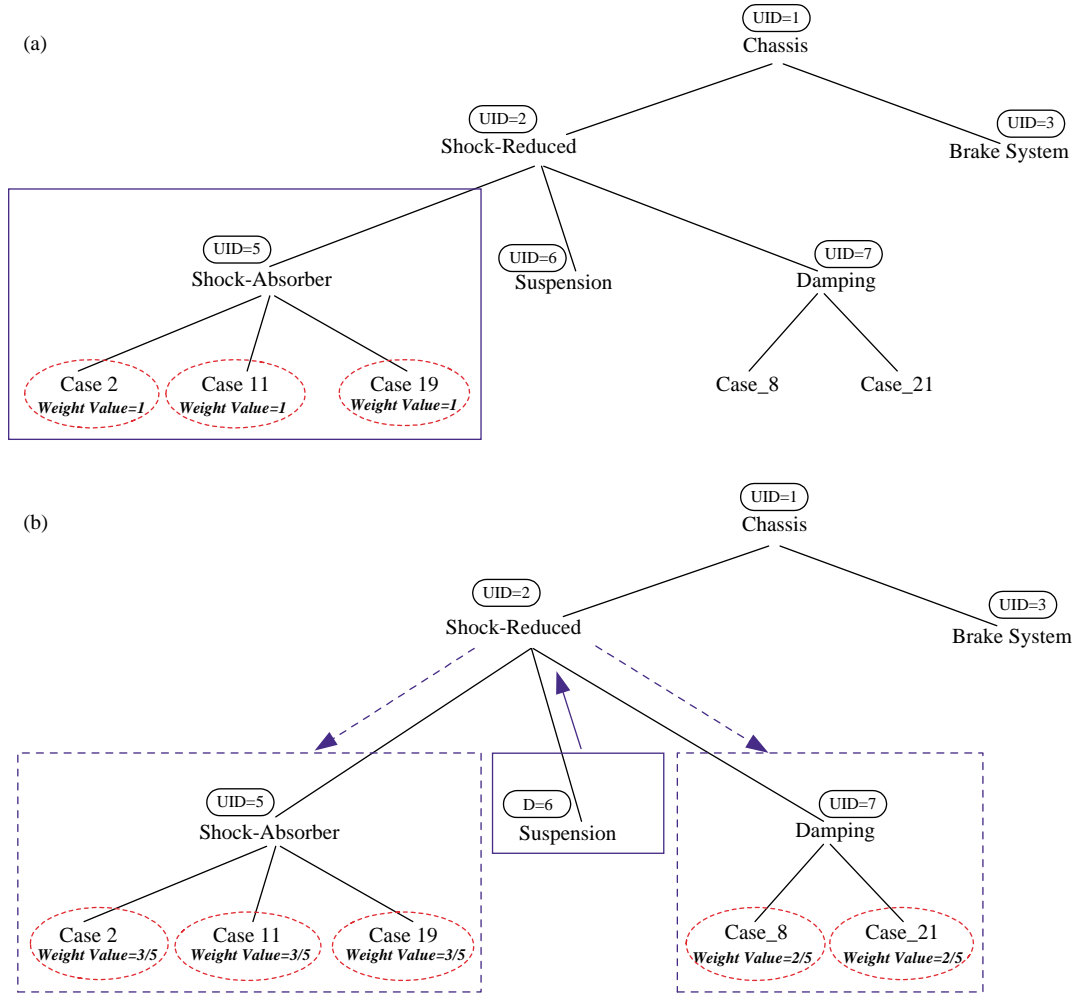


Fig. 23. Allocation of weights for (a) accurate searching and (b) similar searching.

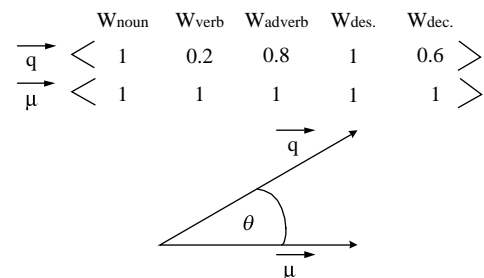
Section 5.3.2. Fig. 23 reveals that, in the accurate searching mode (Fig. 23(a)), the weight for the design cases should be 1 Eq. (6); in the similar searching mode (Fig. 23(b)), the weight for the design cases should be determined by applying the concept of entropy (Chieu & Ng, 2002) Eq. (6). A term that appears more frequently in a particular class (with more design cases) better describes this class so the obtained weight is higher.

In Eq. (6),  $d_{ij}$  represents the total number  $j$  of design cases in class  $i$  while  $W$  represents the weight of the design cases that belong to a particular term in the index structure.

$$W = \begin{cases} \frac{d_{ij}}{\sum_{j=1}^m d_{ij}}, & \text{if the Search Model is in Similar} \\ I, & \text{if the Search Model is in Accurate Searching Mode} \end{cases} \quad (6)$$

$$\text{Sim}(\vec{q}, \vec{\mu}) = \frac{\vec{q} \cdot \vec{\mu}}{|\vec{q}| \times |\vec{\mu}|} \quad (7)$$

The vector model, mentioned in information retrieval (Ricardo & Berthier, 1999), is applied to calculate the total similarity value  $\text{Sim}$  of a design case. As presented in Fig. 24, the weight values for different descriptive terms/elements in the index structures for historical design cases can be represented as the number of dimensions of a vector. For example, if the intersection operation is applied to five terms/elements in the first type of combination for this operation, then the weights of these five terms/elements are expressed in as a 5-dimensional vector equation  $\vec{q}$ ,

Fig. 24. Adopted cosine of  $\theta$  is  $\text{sim}(\mu, q)$  (an example).

and the cosine between vector  $\vec{q}$  and unit vector  $\vec{\mu}$  is obtained using Eq. (7) to yielding their similarity.

The following example clarifies the steps in searching, matching and ranking similar design cases.

If the search modes for the four key elements/terms (nouns, verbs, design types and decomposition types) in the functional requirement described by the engineering designer are all accurate searching modes, and the search mode for the other key element (adverbs) is the similar searching mode, then the steps and results are as follows.

*Step 1:* Assume that the similar design cases derived in the accurate searching mode and the similar searching mode are respectively as shown below.

**Accurate Searching Mode**

$$\begin{cases} N\_CS = \{C(i)\} = \{C_3, C_{11}, C_{13}, C_{15}\} \\ V\_CS = \{C(j)\} = \{C_2, C_3, C_5, C_{11}, C_{15}\} \\ DES\_CS = \{C(k)\} = \{C_3, C_9, C_{11}, C_{12}, C_{15}\} \end{cases}$$

**Similar Searching Mode**

$$\begin{aligned} &: \{A\_CS = \{C(m)\} \cup \{C(n)\} \cup \{C(p)\} \\ &= \{C_2, C_3, C_5, C_7, C_8, C_{11}, C_{12}, C_{15}, C_{16}\} \end{aligned}$$

where,  $C(m) = \{C_2, C_8, C_{11}\}$ ,  $C(n) = \{C_3, C_5, C_{12}, C_{16}\}$ , and  $C(p) = \{C_7, C_{15}\}$

*Step 2:* Using Eq. (6), calculate the weights for the similar design cases of each element/term.

$$\begin{aligned} W(N\_CS) &= W(C(i)) = 1(\text{i.e. } W(C_3) = W(C_{11})) \\ &= W(C_{13}) = W(C_{15}) = 1) \end{aligned}$$

$$\begin{aligned} W(V\_CS) &= W(C(j)) = 1(\text{i.e. } W(C_2) = W(C_3)) \\ &= W(C_5) = W(C_{11}) = W(C_{15}) = 1) \end{aligned}$$

$$\begin{aligned} W(DES\_CS) &= W(C(k)) = 1(\text{i.e. } W(C_3) = W(C_9)) \\ &= W(C_{11}) = W(C_{12}) = W(C_{15}) = 1) \end{aligned}$$

$$\begin{aligned} W(DEC\_CS) &= W(C(h)) = 1(\text{i.e. } W(C_3) = W(C_9)) \\ &= W(C_{11}) = W(C_{15}) = 1) \end{aligned}$$

$$\begin{aligned} W(A\_CS) &= W(C(m)) = 3/9(\text{i.e. } W(C_2) = W(C_8)) \\ &= W(C_{11}) = 3/9) \end{aligned}$$

$$\begin{aligned} W(A\_CS) &= W(C(n)) = 4/9(\text{i.e. } W(C_3) = W(C_5)) \\ &= W(C_{12}) = W(C_{16}) = 4/9) \end{aligned}$$

$$\begin{aligned} W(A\_CS) &= W(C(p)) = 2/9(\text{i.e. } W(C_7) = W(C_{15})) \\ &= 2/9) \end{aligned}$$

*Step 3:* Apply the first type of combination in the intersection operation (Boolean Operation)

$$(i) \{V\_CS\} \cap \{N\_CS\} \cap \{A\_CS\} \cap \{DES\_CS\} \cap \{DEC\_CS\} = \{C_3, C_{11}, C_{15}\}$$

*Step 4:* Determine the total similarity for each similar design case obtained in Step 3 using Eq. (7).

$$c\vec{3} = \langle 1, 1, \frac{4}{9}, 1, 1 \rangle, c1\vec{1} = \langle 1, 1, \frac{3}{9}, 1, 1 \rangle,$$

$$c1\vec{5} = \langle 1, 1, \frac{2}{9}, 1, 1 \rangle, \vec{\mu} = \langle 1, 1, 1, 1, 1 \rangle$$

$$\text{Sim}(c\vec{3}, \vec{\mu}) = \frac{c\vec{3} \cdot \vec{\mu}}{|c\vec{3}| \times |\vec{\mu}|} = 0.970$$

$$\text{Sim}(c1\vec{1}, \vec{\mu}) = \frac{c1\vec{1} \cdot \vec{\mu}}{|c1\vec{1}| \times |\vec{\mu}|} = 0.956$$

$$\text{Sim}(c1\vec{5}, \vec{\mu}) = \frac{c1\vec{5} \cdot \vec{\mu}}{|c1\vec{5}| \times |\vec{\mu}|} = 0.938$$

*Step 5:* Ranking similar design cases according to the results obtained in Step 4. The result of the first type of combination in the intersection operation is not a null set, and the obtained design cases are  $C_3$ ,  $C_{11}$  and  $C_{15}$ . Ranking in order of similarity yields the ultimate result,  $Case_3$ ,  $Case_{11}$  and  $Case_{15}$ . Consequently,  $Case_3$  is the most similar and valuable design case to the functional requirement-based query model that described by the engineering designer.

## 6. Functional feature-based reference design retrieval

This section elucidates the critical techniques for developing the functional requirement-based reference design retrieval mechanism. The critical techniques involve a binary code-based representation for functional features, an ART1 neural network for functional feature-based case clustering and functional feature-based case ranking, all of which are discussed in the following subsections.

### 6.1. Representation of functional features

Successfully utilizing the ART1 neural network in the functional feature-based case clustering requires firstly defining the functional features of parts. Subsequently, a binary code-based representation must be used to represent the defined functional features.

#### 6.1.1. Functional features definition

Functional feature identification is designed to be defined as a part and thus facilitate functional feature-based case

Interaction Type of Features	Relationship between Features	Functional Features
Positive Features and Positive Features	Adjacent_to	Convex
	Intersect	Square
		Cylinder
Positive Features and Negative Features	Add_on	Protrusion
	Is_in	Hole
		Groove/Slot
		Step
Negative Features and Negative Features	Adjacent_to	Convex
	Is_in	Convex
	Intersect	Square
		Cylinder

Fig. 25. Typical functional features in a part.

clustering. Functional features in feature-based design are investigated, and most functional features of a part are formed based on the feature interactions that depend on their spatial relationships (Chen, Chen, & Wen, 2003). Fig. 25 presents the functional features formed by the feature interactions. These features can be classified into positive and negative features. Hence, the feature interactions can be divided into three types - (i) a positive feature and a positive feature, (ii) a positive feature and a negative feature and (iii) a negative feature and a negative feature. The first type includes the relationships “adjacent\_to” and “intersect”. The second type includes the relationships “add\_on” and “is\_in”. The third type includes the relationships “adjacent\_to”, “is\_in” and “intersect”. Each type of feature interaction generates several functional features based on the specific relationship between them. For instance,

a positive feature and a negative feature may generate the “hole”, “groove/slot” or “step” function, based on the relationship “is\_in”.

### 6.1.2. Binary code-based representation of functional features

Parts are characterized using a list of functional features, which are specified by binary variables. In the previous subsection, these 11 functional features were required to record the specific features of the part. When a part is coded based on the list, “one” indicates that the part has a given functional feature, while “zero” means that it does not.

Fig. 26(a) shows the binary code-based representation of functional features of the sample part displayed in Fig. 26(b), which is represented by an 11-component vector  $(X_1, X_2, \dots, X_{11})$ . The components  $X_1$ – $X_3$  in the vector indicate the “convex”, “square” and “cylinder” functions, which are formed by the first type of feature interactions. Furthermore, the components  $X_4$ – $X_7$  represent the “protrusion”, “hole”, “groove/slot” and “step” in an ordered sequence. They are generated through the second type of feature interactions. The last four components  $X_8$ – $X_{11}$  express the functional features of the third type of feature interaction, namely “convex”, “convex”, “square” and “cylinder”. Therefore, the functional features involved in the sample part include the “hole” and “slot” functions, and are represented as “one”.

### 6.2. Adaptive resonance theory (ART1) neural network

This study adopts the adaptive resonance theory (ART1) neural network to solve the problem of functional feature-based case clustering. The ART1 neural network can be defined in terms of ART1 characteristics, ART1 architecture and ART1 algorithm, as detailed below.

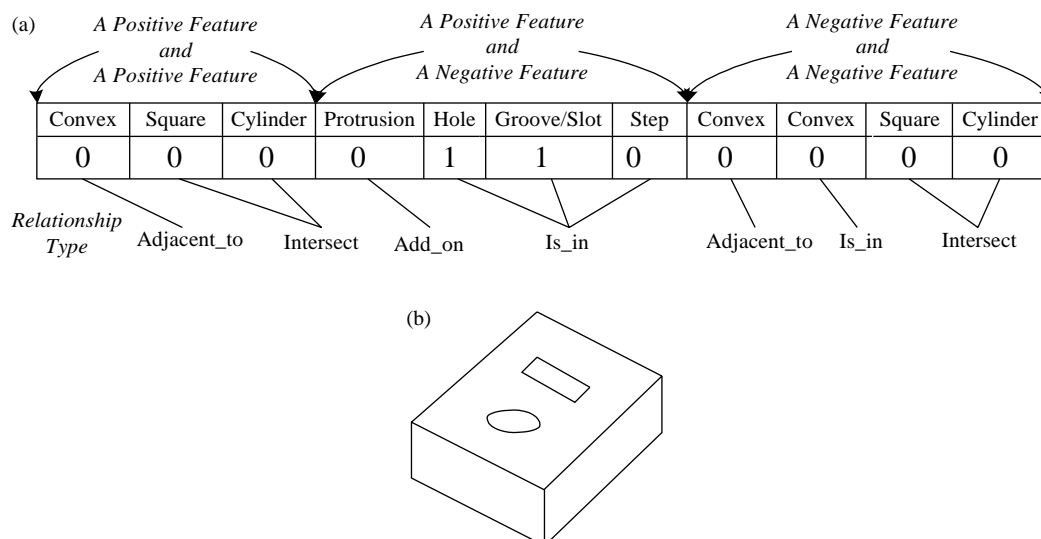


Fig. 26. (a) Binary code-based representation of functional features for the part (b) sample part.

### 6.2.1. ART1 characteristics

An examination of neural networks shows that most are either plastic (during the learning phase) or stable (during recall, when the weights are frozen), but not both. Carpenter and Grossberg (Carpenter & Grossberg, 1987) proposed the Adaptive Resonance Theory (ART1) neural network for cluster discovery through unsupervised learning, to solve the stability-plasticity dilemma faced by every learning system. This theory has the following characteristics.

- **Binary-based input vector:** ART1 is designed for binary 0/1 inputs; each input vector may have more 0/1 elements.
- **Stability and plasticity:** The ART1 network is sufficiently stable to preserve significant past learning but remains sufficiently adaptable to incorporate new information (clusters) as necessary.
- **Unsupervised learning:** In unsupervised learning, no external teacher or critic oversees the learning process and provides feedback information. Moreover, no environmental feedback is available to indicate the nature or correctness of the outputs. The network must discover patterns, features, regularities, correlations or categories of input data and code for them in the output.
- **Quick learning capability:** When an input pattern is not sufficiently similar to any existing prototype, a new node is formed to represent a new category that involves the input pattern as the prototype.
- **Concept of vigilance parameter:** “Sufficiently similar” depends on a vigilance parameter  $\rho$ , with  $0 < \rho < 1$ . The similarity condition is more easily met if  $\rho$  is small, leading to coarse categorization. However, if  $\rho$  is 1, numerous finely divided categories are formed. The value of the vigilance parameter can be adjusted during learning; increasing it leads to the subdivision of existing categories.

### 6.2.2. ART1 architecture

Fig. 27 presents the architecture of the ART1 neural network. Each input vector  $X$  has  $m$  binary 0 or 1 elements. Let the weights of the bottom-up links,  $x_j$  to  $y_i$ , be denoted by  $\bar{w}_{ij}$ , and let the weights of the top-down links,  $y_i$  to  $x_j$ , be

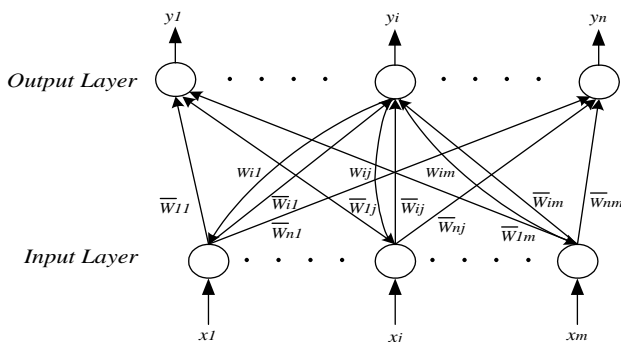


Fig. 27. Basic ART1 architecture.

represented by  $w_{ij}$ . The first subscript of a top-down weight indicates the source node, while the second subscript indicates the destination node. The weight vectors  $w_i = (w_{i1}, w_{i2}, \dots, w_{im})^T$ ,  $i = 1, 2, \dots, n$ , represent stored prototype vectors and thus are also binary 0 or 1 vectors, where  $i$  is the output nodes or categories, each of which can be enabled or disabled.

### 6.2.3. ART1 algorithm

The algorithm that characterizes ART1 clearly describes the operation of the ART1 neural network. Before the algorithm is detailed, the variables used in the ART1 neural network are summarized.

- $m$ , Number of input vector elements,
- $n$ , Number of output nodes,
- $w_{ij}$ , Weights of the top-down links, where  $i$  denotes the index of the output node ( $i = 1, 2, \dots, n$ ) and  $j$  represents the index of the input vector elements ( $j = 1, 2, \dots, m$ ).
- $\bar{w}_{ij}$ , Weights of the bottom-up links,
- $x$ , Input vector,
- $y_i$ , Net value,
- $r$ , Similarity value,
- $\rho$ , Vigilance parameter.

The algorithm of the ART1 neural network is as follows.

**Input:** A set of input vectors  $x$  to be clustered, where  $x \in \{0, 1\}^m$ .

**Output:** A set of weight vectors  $w_i = (w_{i1}, w_{i2}, \dots, w_{im})^T$ ,  $i = 1, 2, \dots, n$  representing the prototype vectors of the discovered clusters, where  $n$  represents the number of clusters identified.

**Step 0:** Set  $w_{ij}(0) = 1$ ,  $\bar{w}_{ij}(0) = 1/(1 + m)$ , for  $0 < \rho \leq 1$ .

**Step 1:** Input a new sample  $x$  to the input nodes.

**Step 2:** Enable all output nodes.

**Step 3:** Apply bottom-up processing to yield a weighted sum

$$y_i = (\bar{w}_i)^T x = \sum_{j=1}^m \bar{w}_{ij} x_j, \quad (8)$$

where  $\bar{w}_{ij}$  is the normalization of  $w_{ij}$  and is given by

$$\bar{w}_{ij} = \frac{w_{ij}}{0.5 + \sum_j w_{ij}}, \quad j = 1, 2, \dots, m. \quad (9)$$

**Step 4:** Use the **max net** procedure to identify the output node  $i$  with the greatest  $y_i$  value.

**Step 5:** Verify that  $x$  belongs to the  $i$ th cluster by performing top-down processing and forming the weighted sum  $\sum_j w_{ij} x_j$ . Then perform the following check.

$$\text{If } r = \frac{\sum_{j=1}^m w_{ij} x_j}{\|x\|} > \rho, \quad \text{where } \|x\| = \sum_{j=1}^m |x_j|, \quad (10)$$



Table 3  
Binary code for functional features of five cases

Input samples	Positive feature and positive feature			Positive feature and negative feature				Negative feature and negative feature			
	Adjacent_to		Intersect	Add_on		Is_in		Adjacent_to		Is_in	Intersect
	Convex	Square	Cylinder	Protrusion	Hole	Groove	Step	Convex	Convex	Square	Cylinder
x <sub>1</sub> (Case_1)	0	0	1	0	0	1	0	0	1	1	1
x <sub>2</sub> (Case_2)	0	0	1	1	0	0	0	1	0	0	0
x <sub>3</sub> (Case_3)	1	0	1	1	0	0	0	0	0	0	0
x <sub>4</sub> (Case_4)	0	1	0	0	0	1	1	0	1	1	1
x <sub>5</sub> (Case_5)	0	1	1	0	0	1	0	0	0	1	1

x<sub>i</sub> indicates the vector for functional features in Case\_i

**Then**  $x$  belongs to the  $i$ th cluster, proceed to Step 6,  
**Otherwise, if** the top layer has more than a single enabled node remaining, then go to Step 7,  
Alternatively, create a new output node  $i$  whose initial weights are set as in Step 0 and proceed to Step 6.  
Step 6: Update the weights as follows.

$$w_{ij}(t+1) = w_{ij}(t)x_j, \quad j = 1, 2, \dots, m. \quad (11)$$

which updates the weights of the  $i$ th cluster (either created or existing). Then go to Step 1.

Step 7: The output node  $i$  is disabled by fixing  $y_i$  to 0. This node therefore does not participate in the current cluster search. The algorithm returns to Step 3, and attempts to establish a new cluster that differs from  $i$  for pattern  $x$ .

The above ART1 algorithm includes both a learning mode and a performance model. For a given input sample  $x$ , the algorithm can terminate in two ways. First, if a matching prototype vector  $w_i$  is found, it is adjusted in Step 6 based on Eq. (11) and the category  $i$  is outputted. Second, if the stored categories contain no suitable prototype vector, then a new output node  $i^*$  is created. This output node represents a new category with a prototype vector  $w^*i$  that equals input  $x$  in Step 6 according to Eq. (11). Finally, the new category  $i^*$  is created.

### 6.3. Functional feature-based cases retrieval by art1: an example

This section uses a hypothetical part to illustrate functional feature-based case retrieval by applying the ART1 neural network. First, functional feature-based case clustering using the ART1 neural network is interpreted using an example. Then, the vector model is applied to deal with a functional feature-based case ranking.

#### 6.3.1. Functional feature-based cases clustering

The functional features of five cases (Case\_1, Case\_2, Case\_3, Case\_4 & Case\_5) are chosen to validate the aforementioned ART1 technique (as discussed in Sections 6.2.2 and 6.2.3) for functional feature-based case clustering.

Table 3 presents the binary code for these functional features.

The initialization process of ART1 sets the initial weights are  $w_{ij} = 1$  and  $\bar{w}_{ij} = 1/2$ ,  $j = 1, 2, \dots, 11$ ;  $i = 1, 2, 3, 4, 5$ . Meanwhile, the vigilance parameter  $\rho$  is set to 0.5. The input samples are then individually fed into the ART1 algorithm. Additionally, five output nodes are assumed to be available.

**Sample x<sub>1</sub> (Case\_1).** When sample  $x_1$  is fed into the algorithm, the output node with the largest output of the five is denoted as number 1. Now,  $w_{ij} = 1$  for all  $i, j$  at this time, so  $r = 1$  in Eq. (10) and the vigilance test is passed unconditionally. Consequently, the first cluster is defined unconditionally. The weights are then changed based on Eqs. (11) and (9):

$$w_{1,3} = w_{1,6} = w_{1,9} = w_{1,10} = w_{1,11} = 1$$

$$w_{1,j} = 0, \quad j = 1, 2, 4, 5, 7, 8,$$

$$\bar{w}_{1,3} = \bar{w}_{1,6} = \bar{w}_{1,9} = \bar{w}_{1,10} = \bar{w}_{1,11} = \frac{2}{11}$$

$$\bar{w}_{1,j} = 0, \quad j = 1, 2, 4, 5, 7, 8,$$

**Sample x<sub>2</sub> (Case\_2).** When sample  $x_2$  is fed into the algorithm, no top-layer node competes for clustering, since only one active node exists; that is, node 1 is the unconditional winner. The vigilance test indicates that

$$r = \frac{\sum_{j=1}^{11} w_{1j}x_j}{\|x\|} = \frac{1}{3} = 0.33 < \rho = 0.5$$

Hence, it fails the test. Output node  $i$  is now the single enabled node, so further searching is unnecessary, and sample  $x_2$  is considered to be a new cluster represented by another output node, number 2. The corresponding weights  $w_2$  and  $\bar{w}_2$  are then computed as,

$$w_{2,3} = w_{2,4} = w_{2,8} = 1, \quad w_{2,j} = 0,$$

$$j = 1, 2, 5, 6, 7, 9, 10, 11$$

$$\bar{w}_{2,3} = \bar{w}_{2,4} = \bar{w}_{2,8} = \frac{2}{7}, \quad \bar{w}_{2,j} = 0, \\ j = 1, 2, 5, 6, 7, 9, 10, 11$$

**Sample  $x_3$  (Case\_3).** When sample  $x_3$  is input, the following output values are computed based on Eq. (8).

$$y_1 = \frac{2}{11} = 0.18, \quad y_2 = \frac{4}{7} = 0.57$$

$\because y_1 < y_2$ ,  $\therefore$  output node 2 is a winner. Moreover, the vigilance test is passed because

$$r = \frac{\sum_{j=1}^{11} w_{2j} x_j}{\|x\|} = \frac{2}{3} = 0.67 > \rho = 0.5$$

Accordingly, weights  $w_2$  and  $\bar{w}_2$  must be changed based on Eqs. (11) and (9), as follows.

$$w_{2,3} = w_{2,4} = 1, \quad w_{2,j} = 0, \quad j = 1, 2, 5, 6, 7, 8, 9, 10, 11,$$

$$\bar{w}_{2,3} = \bar{w}_{2,4} = \frac{2}{5}, \quad \bar{w}_{2,j} = 0, \quad j = 1, 2, 5, 6, 7, 8, 9, 10, 11,$$

**Sample  $x_4$  (Case\_4).** When sample  $x_4$  is input, the following output values are determined based on Eq. (8):

$$y_1 = \frac{8}{11} = 0.73, \quad y_2 = 0$$

$\because y_1 > y_2$ ,  $\therefore$  output node 1 is a winner. Moreover, the vigilance test is passed because

$$r = \frac{\sum_{j=1}^{11} w_{1j} x_j}{\|x\|} = \frac{4}{6} = 0.67 > \rho = 0.5$$

Hence, weights  $w_1$  and  $\bar{w}_1$  must be changed based on Eqs. (11) and (9), as follows.

$$w_{1,6} = w_{1,9} = w_{1,10} = w_{1,11} = 1, \quad w_{1,j} = 0, \\ j = 1, 2, 3, 4, 5, 7, 8$$

$$\bar{w}_{1,6} = \bar{w}_{1,9} = \bar{w}_{1,10} = \bar{w}_{1,11} = \frac{2}{9}, \quad \bar{w}_{1,j} = 0, \\ j = 1, 2, 3, 4, 5, 7, 8,$$

**Sample  $x_5$  (Case\_5).** When sample  $x_5$  is fed, the following output values are calculated based on Eq. (8):

$$y_1 = \frac{2}{3} = 0.67, \quad y_2 = \frac{2}{5} = 0.4$$

$\because y_1 > y_2$ ,

$\therefore$  output node 1 is a winner. Moreover, the vigilance test is passed because

$$r = \frac{\sum_{j=1}^{11} w_{1j} x_j}{\|x\|} = \frac{3}{5} = 0.6 > \rho = 0.5$$

Therefore, weights  $w_1$  and  $\bar{w}_1$  must be changed according to Eqs. (11) and (9), as follows.

$$w_{1,6} = w_{1,10} = w_{1,11} = 1, \quad w_{1,j} = 0, \\ j = 1, 2, 3, 4, 5, 7, 8, 9$$

$$\bar{w}_{1,6} = \bar{w}_{1,10} = \bar{w}_{1,11} = \frac{2}{7}, \quad \bar{w}_1 = 0, \\ j = 1, 2, 3, 4, 5, 7, 8, 9$$

The above simulation of learning by the ART1 neural network identifies two categories: one contains samples  $x_1$  (Case\_1),  $x_4$  (Case\_4) and  $x_5$  (Case\_5), and other contains samples  $x_2$  (Case\_2) and  $x_3$  (Case\_3). In this example, if  $x_5$  (Case\_5) is a query pattern, then  $x_1$  (Case\_1) and  $x_4$  (Case\_4) are similar cases to  $x_5$  (Case\_5) in terms of functional features.

### 6.3.2. Functional feature-based cases ranking

The vector model defines the similarity between two terms as the cosine of the angle between their two vectors. Therefore, this vector model is adopted and slightly modified to calculate the degree of similarity between similar cases that have been acquired through functional feature-based case clustering based on the query of functional features. The query vector  $\vec{q}$  can be defined as  $\vec{q} = (x_{1,q}, x_{2,q}, \dots, x_{11,q})$ , whereas the vector for similar cases  $\vec{c}_j$  is represented by  $\vec{c}_j = (x_{1,j}, x_{2,j}, \dots, x_{11,j})$ . Therefore, a similar case  $c_j$  and user query  $q$  are represented as 11-dimensional vectors, as shown in Fig. 28. The correlation between vectors  $\vec{c}_j$  and  $\vec{q}$  is quantified as follows.

$$\text{sim}(c_j, q) = \frac{\vec{c}_j \cdot \vec{q}}{|\vec{c}_j| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{11} x_{i,j} \times x_{i,q}}{\sqrt{\sum_{i=1}^{11} x_{i,j}^2} \times \sqrt{\sum_{i=1}^{11} x_{i,q}^2}}$$

Using the example discussed at the end of the previous subsection, the correlation coefficients among  $x_1$  (Case\_1) and  $x_5$  (Case\_5),  $x_4$  (Case\_4) and  $x_5$  (Case\_5) are calculated as.

$$\text{sim}(c_1, c_5) = \frac{\sum_{i=1}^{11} x_{i,1} \cdot x_{i,5}}{\sqrt{\sum_{i=1}^{11} x_{i,1}^2} \cdot \sqrt{\sum_{i=1}^{11} x_{i,5}^2}} = \frac{4}{\sqrt{5} \cdot \sqrt{5}} = 0.8$$

$$\text{sim}(c_4, c_5) = \frac{\sum_{i=1}^{11} x_{i,4} \cdot x_{i,5}}{\sqrt{\sum_{i=1}^{11} x_{i,4}^2} \cdot \sqrt{\sum_{i=1}^{11} x_{i,5}^2}} = \frac{4}{\sqrt{6} \cdot \sqrt{5}} = 0.73$$

The above calculations indicate that  $x_1$  (Case\_1) resembles  $x_5$  (Case\_5) more than it does  $x_4$  (Case\_4).

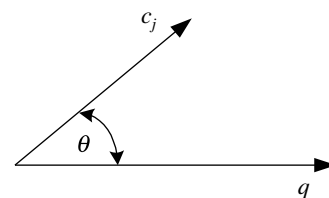
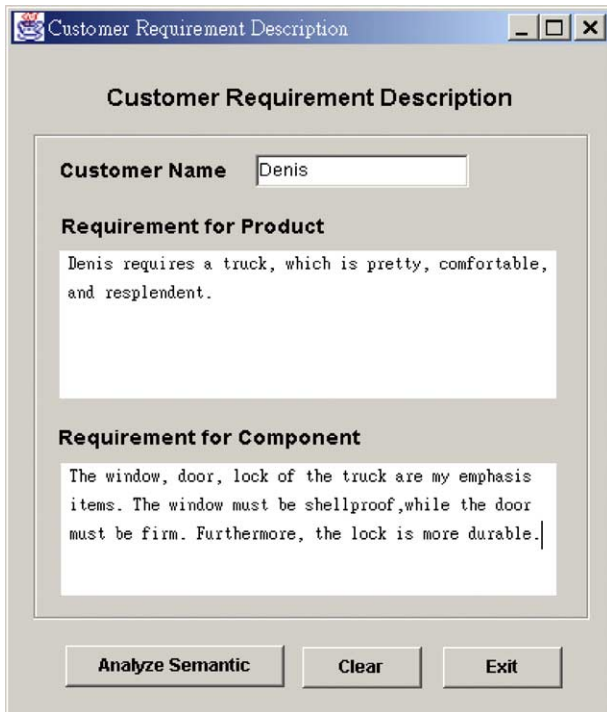


Fig. 28. Cosine of  $\theta$  is adopted as  $\text{sim}(c_j, q)$ .



**Customer Requirement Description**

**Customer Name**

**Requirement for Product**

Denis requires a truck, which is pretty, comfortable, and resplendent.

**Requirement for Component**

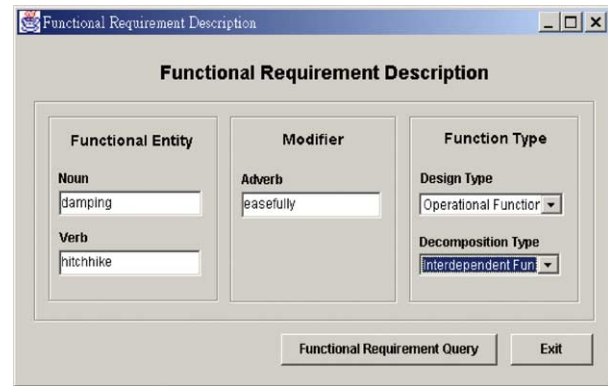
The window, door, lock of the truck are my emphasis items. The window must be shellproof, while the door must be firm. Furthermore, the lock is more durable.

Fig. 29. Description of customer requirements.

Therefore, the degrees of similarity to the query pattern  $x_5$  (Case\_5) follow the order  $x_1$  (Case\_1) and  $x_4$  (Case\_4).

## 7. Implementation and experimental example

A prototype multi-layer reference design retrieval mechanism, based on the proposed techniques for



**Functional Requirement Description**

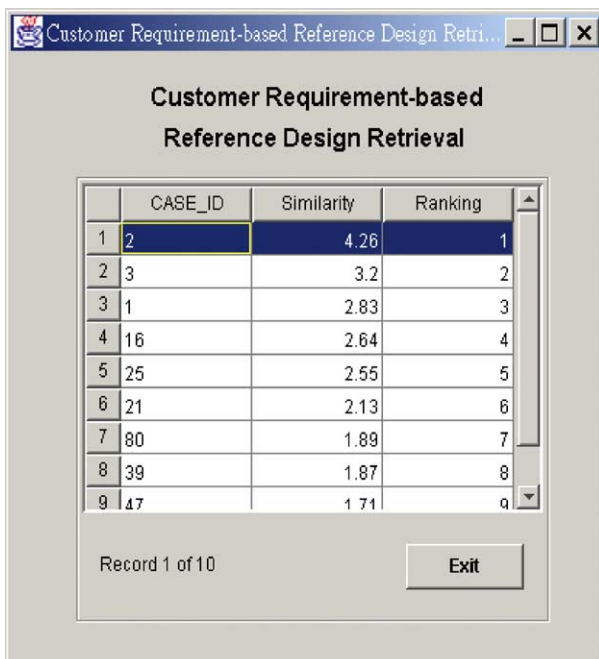
Functional Entity	Modifier	Function Type
Noun <input type="text" value="damping"/>	Adverb <input type="text" value="easily"/>	Design Type <input type="text" value="Operational Function"/>
Verb <input type="text" value="hitchhike"/>		Decomposition Type <input type="text" value="Interdependent Fun"/>

Fig. 31. Description of functional requirements.

multi-layer reference design retrieval was implemented at the Enterprise System Engineering Research Laboratory (ESERL) at National Cheng Kung University, Taiwan, ROC.

The hardware used in the experiment comprised an Acer Veriton 7100 PC and an Acer ALTOS PC server, acting as client and server, respectively. The software used to implement the mechanisms were (i) Microsoft Windows 2000 Server and Window XP Professional as the development platform, (ii) Borland Jbuilder 7.0 as the programming development tool and (iii) Microsoft SQL Server 2000 for establishing databases. A SQL database served as the engineering design case base for storing related product information and engineering knowledge.

Figs. 29–35 present parts of the user interfaces of a multi-layer reference design retrieval mechanism. Figs. 29, 31 and 33 present the customer requirement description, functional requirement description and functional feature query screens. Figs. 30, 32 and 34 display screens used for the similarity ranking of customer requirement-based, functional requirement-based and functional feature-based retrieved cases. Fig. 35 depicts the content of the most similar design case. Fig. 36 presents the schema for historical design cases in the database.

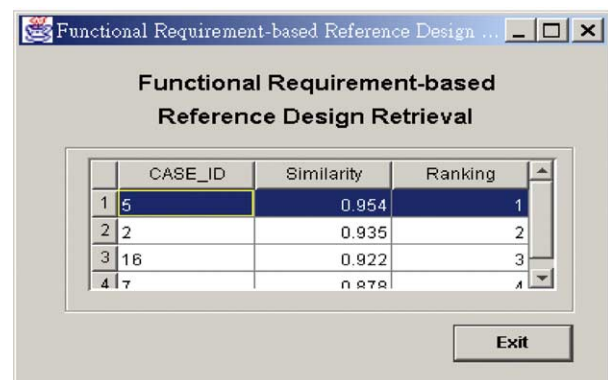


**Customer Requirement-based Reference Design Retrieval**

	CASE_ID	Similarity	Ranking
1	2	4.26	1
2	3	3.2	2
3	1	2.83	3
4	16	2.64	4
5	25	2.55	5
6	21	2.13	6
7	80	1.89	7
8	39	1.87	8
9	47	1.71	9

Record 1 of 10

Fig. 30. Similarity ranking for customer requirement-based retrieved cases.



**Functional Requirement-based Reference Design Retrieval**

	CASE_ID	Similarity	Ranking
1	5	0.954	1
2	2	0.935	2
3	16	0.922	3
4	7	0.878	4

Fig. 32. Similarity ranking for functional requirement-based retrieved cases.

**Functional Features-based Reference Design Retrieval**

**Positive Feature and Positive Feature**

Adjacent\_to\_Convex: 1  
Intersect\_Square: 1  
Intersect\_Cylinder: 0

**Positive Feature and Negative Feature**

Add\_on\_Protrusion: 1  
Is\_in\_Hole: 0  
Is\_in\_Groove: 0  
Is\_in\_Step: 1

**Negative Feature and Negative Feature**

Adjacent\_to\_Convex: 1  
Is\_in\_Convex: 1  
Intersect\_Square: 0  
Intersect\_Cylinder: 1

Case\_ID: 30 Write\_In Clear

Functional Features Query Exit

Fig. 33. Functional features query.

## 8. Conclusions and directions for future research

### 8.1. Conclusions

This study first presents an engineering knowledge management framework, and then seeks to develop a multi-

**Historical Design Case**

**Case Feature**

Case Name: RV model Language: Visio  
Case ID: 5 Version: V1  
Contributor: Maggie Date: 6/18/2003  
Creator: Denis Model Location: E:/Model file

**Tag for Product Information**

Customer Requirements: PT\_CR\_501 Link  
Functional Requirements: PT\_FR\_502 Link  
Functional Features: PT\_FF\_503 Link

**Tag for Engineering Knowledge**

Feature-based Design: KT\_FD\_501 Link  
Engineering Change: KT\_EC\_502 Link  
Design by Modification/Reference: KT\_DMR\_503 Link

Exit

Fig. 35. Content of the most similar design case.

**Functional Features-based Reference Design Retrieval**

Field Name: Cluster\_Number Acquire Results  
Field Value: 3 Calculate Similarity

Similarity

	Intersect...	NN_Intersect...	Cluster_Number	Similarity
2	1	3	0.926	
3	1	3	0.798	
4	0	3	0.668	
5	1	3	0.567	

Record 1 of 9 Exit

Fig. 34. Similarity ranking for functional feature-based retrieved cases.

layer reference design retrieval technology that includes customer requirement-based reference design retrieval, functional requirement-based reference design retrieval and functional feature-based reference design retrieval. Customer requirement-based reference design retrieval involves a structured query model for customer requirements, a case-based representation of designed entities, a customer requirement-based index structure for historical design cases and customer requirement-based case searching, matching and ranking mechanisms. The functional requirement-based reference design retrieval involves a structured query model for functional requirements, a functional requirement-based index structure for historical design cases and functional requirement-based case searching, matching and ranking mechanisms. The crucial techniques involved in functional feature-based reference design retrieval include a binary code-based representation for functional features, an ART1 neural network for functional feature-based case clustering, and functional feature-based case ranking. The multi-layer reference design retrieval mechanism is implemented based on these aforementioned techniques.

The results of this study facilitate the sharing of engineering knowledge for engineering knowledge management purposes in engineering design environments. They can thus increase product development capability; reduce development cycle time and cost, and ultimately increase the product marketability.



Case Name	Case Id	Eng_Model Loc	Eng_Model Name	Eng_Model_Creator	Eng_Model_C
Car	1	Doc_001	English	Car_Model	Marly
Car	2	Doc_101	English	car_Model	James
Car	3	Doc_006	English	Car_Model	james
Truck	4	Doc_211	English	Truck_Model	james
Car	5	Doc_234	English	Car_Model	Marly
Car	7	Doc_154	English	Car_Model	Dennis
SUV	12	Doc_687	English	SUV_Model	Dennis
SUV	15	Doc_587	English	SUV_Model	Oliver
Car	16	Doc_387	English	Car_Model	Oliver
Car	21	Doc_328	English	Car_Model	Marly
RV	23	Doc_638	English	RV_Model	Peter
Car	25	Doc_368	English	Car_Model	Peter
Car	28	Doc_638	English	Car_Model	John
Truck	34	Doc_873	English	Truck_Model	Dinos
Car	39	Doc_637	English	car_model	Dinos
Car	43	Doc_682	English	Car_Model	Dinos
Car	47	Doc_587	English	Car_Model	Wildcat
RV	78	Doc_367	English	RV_Model	Wildcat
Car	80	Doc_367	English	Car_Model	jean
Truck	90	Doc_343	English	Truck_Model	jean
Car	124	Doc_368	English	Car_Model	Raen

Fig. 36. Schema for historical design cases in database.

## 8.2. Future research directions

Future study should address the following areas to improve the sharing of engineering knowledge in engineering design.

- *Engineering specification-based reference design retrieval mechanism:* Such a mechanism would extend functional feature-based reference design retrieval. Based on the search results of the functional feature-based reference design retrieval mechanism developed in this study, the engineering specification-based reference design retrieval mechanism can filter results more precisely by considering engineering specifications.
- *Engineering knowledge query mechanism:* Using engineering rule bases, engineering designers can conveniently query engineering knowledge using the knowledge query function to solve related design problems.

The techniques for multi-layer reference design retrieval developed herein in this study can be regarded as a valuable reference model/basis and then applied to other knowledge-intensive works, including system development, planning, diagnosis, forecast, maintenance and control, to meet the challenges faced by firms in the age of the knowledge economy.

## Acknowledgements

The authors would like to thank the National Science Council of the Republic of China, Taiwan for financially supporting this research under Contract No. NSC 92-2212-E-006-065.

## References

- Califf, M. E., & Mooney, R. J. (1997). Relational learning of pattern-match rules for information extraction, In *Proceedings of the ACL workshop on natural language learning*, NJ: Association for Computational Linguistics.
- Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37, 54–115.
- Chen, Y. M., & Jan, Y. D. (2000). Enabling allied concurrent engineering through distributed engineering information management. *Robotics and Computer-Integrated Manufacturing*, 16(1), 9–27.
- Chen, Y. M., & Lian, M. W. (1999). Design and implementation of a collaborative engineering information system for allied concurrent engineering. *International Journal of Computer Integrated Manufacturing*, 13(1), 11–30.
- Chen, Y. M., Shir, W. S., & Shen, C. Y. (2002). Distributed engineering change management for allied concurrent engineering. *International Journal of Computer Integrated Manufacturing*, 15(2), 127–151.
- Chen, Y. M., Shen, C. Y., & Shr, W. S. (in press). Holonic engineering data management framework for allied concurrent engineering. *Journal of Concurrent Engineering: Research and Applications*, Special issue on Concurrent Engineering (CE) and Communication.
- Chen, Y. M., Wen, C. C., & Ho, C. T. (2003). Extraction of geometric characteristics for manufacturability assessment. *Robotics and Computer-Integrated Manufacturing*, 19(4), 371–385.
- Chieu, H. L., & Ng, H. T. (2002). A maximum entropy approach to information extraction from semi-structured and free text, *18th National Conference on Artificial Intelligence*. 786–791.
- Church, K. W. (1998). A stochastic parts program and noun phrase parser for unrestricted text. *Proceeding of the Second Conference on Applied Natural Language Processing*, 136–143.
- Cockshoot, D. W. (1996). Engineering data management for concurrent engineering globally. *Computer and Control Engineering Journal*, 7(2), 69–74.
- Dongwook, S., Hyuncheol, J., & Honglan, J. (1998). BUS: An effective indexing and retrieval scheme in structured documents. *ACM International Conference on Digital Libraries*, 235–243.
- Eric, B. (1995). *Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging*.
- Ertas, A., & Jones, J. C. (1993). *The engineering design process*. New York: Wiley.

- Homer, G. R., Thompson, D. M., & Deacon, M. (2002). A distributed document management system. *Computer and Control Engineering Journal*, 13(6), 315–318.
- Lee, Y. K., Yoo, S. J., Yoon, K., & Berra, P. B. (1996). Index structures for structured documents. *ACM International Conference on Digital Libraries*, 91–99.
- O’Leary, D. E. (1998). Enterprise knowledge management. *Computer*, 31(3), 54–61.
- Pahl, G., Beitz, W., & Wallace, K. (1984). *Engineering design*. London: Design Council.
- Ricardo, B. Y., & Berthier, R. N. (1999). *Modern information retrieval*. New York: ACM Press.
- Rupple, C. P., & Harrington, S. J. (2001). Sharing knowledge through intranets: A study of organization culture and intranet implementation. *Professional Communication, IEEE Transaction*, 44(1), 37–52.
- Sowa, J. F. (1984). *Conceptual structure: information processing in mind and machine*. New York: IBM Systems Research Institute.
- Wei, C. P., Hu, J. H., & Chen, H. H. (2002). Design and evaluation of a knowledge management system. *Software IEEE*, 19(3), 56–59.