A Systematic Approach for the Specification of Customer Requirements

Zhenyu Chen, Shengji Yao, Yong Zeng
 Concordia Institute for Information Systems Engineering
 Concordia University
 1455 de Maisonneuve West, CB-410-16
 Montreal, Quebec, Canada H3G 1M8
 E-mail: zeng@ciise.concordia.ca

Abstract

Specification of customer requirements is critical for designers to solve a design problem and to deliver a quality design. This paper proposes a systematic approach to formalize various customer requirements using the axiomatic theory of design modeling. The input of this formalization process is the design requirements described in natural language by the customers. The output is a structure underlying these design requirements for designers to use throughout the rest of the design process. This structure of design problem consists of three parts: product, environment, and structural as well as performance requirements. The formalization process includes two major steps: linguistic analysis and structure analysis. The linguistic analysis identifies objects and their relationships included in the customer description of a design problem through the lexical and syntactical properties of words and sentences that form the design problem. The structure analysis integrates all objects and relationships identified from linguistic analysis to form a structure of the design problem. A software prototype is developed to show the feasibility of this approach.

Key Words—customer requirements, formalization, linguistic analysis, axiomatic theory of design modeling, design science

1. Introduction

Specification of customer requirements is the first step in formalizing the design process. Design solutions should reflect the requirements of the customer. If customer requirements are misunderstood at the beginning, it could fail the design process and waste time and resources. A systematic approach would be beneficial to help designers to specify the customer requirements in a proper manner.

Regardless of the importance of the requirement specification, this has not been well investigated in the literature [1-5]. Two major challenges have complicated this problem: first, the information included in customer requirements is too broad and diversified to be put into a well defined structure; secondly, the customer requirements are usually vaguely described so that their modeling has been

taken as extremely difficult, if not impossible [6, 7].

This paper attempts to solve the two problems addressed above. A formal structure of design problem will be derived using the axiomatic theory of design modeling. This structure can be taken as the base to represent and model various requirements given by customers. A formalization process is then proposed to transform customer requirements into the formal structure. A software prototype is presented to validate the method proposed in this paper.

It should be noted that, this paper derives all the models and methods using the axiomatic theory of design modeling [8]. This is different from most of the research in the literature, where observation and experimentation play a major role [9-14].

2. Formulation of Customer Requirements

This section aims to provide some background information for describing the concepts proposed in this paper. A brief review of the axiomatic theory of design modeling is introduced as a mathematical tool for formulating and formalizing customer requirements. Then the base for representing various customer requirements is given, which is derived from the axiomatic theory of design modeling [2]. Finally a design example will be used to help the explanation of proposed concepts.

2.1. Review of Axiomatic Theory of Design Modeling

Axiomatic theory of design modeling is a logical tool for representing and reasoning about object structures [8]. It provides a formal approach that allows for the development of design theories following logical steps based on mathematical concepts and axioms. Three primitive concepts are used in the axiomatic theory of design modeling: universe, object, and relation. Their definitions can be found from the Random House Webster's Unabridged Dictionary as follows.

[Definition 1] The universe is the whole body of things and phenomena observed or postulated.

[Definition 2] An object is anything that can be observed or postulated in the universe.

It can be seen from the two definitions above that universe is the whole body of objects.

[Definition 3] A relation is an aspect or quality that connects two or more objects as being or belonging or working together or as being of the same kind. Relation can also be a property that holds between an ordered pair of objects.

$$R = A \sim B, \exists A, B, R, \quad (1)$$

where A and B are objects. $A \sim B$ is read as “ A relates to B ”. R is the relation from object A to object B . The basic properties of relations include idempotent, commutative, transitive, associative and distributive.

Based on these concepts, two axioms can be defined in the axiomatic theory of design modeling.

[Axiom1] Everything in the universe is an object.

[Axiom 2] There are relations between objects.

It can be seen from these two axioms that the characteristics of relations would play a critical role in the axiomatic theory of design modeling. We need to define a group of basic relations to capture the nature of object representation. Two corollaries of the axiomatic theory of design modeling can be used to represent various relations in the universe.

[Corollary 1] Every object in the universe includes other objects. Symbolically,

$$A \supseteq B, \forall A \exists B. \quad (2)$$

B is called a subobject of A . The symbol $\supseteq$ is inclusion relation. The inclusion relation is transitive and idempotent but not commutative.

[Corollary 2] Every object in the universe interacts with other objects. Symbolically,

$$C = A \otimes B, \forall A, B \exists C, \quad (3)$$

where C is called the interaction of A on B . The symbol $\otimes$ represents interaction relation. Interaction relation is idempotent but not transitive or associative. Based on Corollary 1 and 2, a new operation can be developed as follows:

[Definition 4] Structure operation, denoted by $\oplus$, is defined by the union of an object and the relation of the object to itself.

$$\oplus O = O \cup (O \otimes O), \quad (4)$$

where $\oplus O$ is the structure of object O .

The structure operation provides the aggregation mechanism for representing the object evolution in the design process.

2.2. Base for Representing Customer Requirements

This section derives a base for representing customer requirements using the axiomatic theory of design modeling. This base is the foundation for the formal study of customer requirements. It forms a coordinate system for representing various customer requirements. Especially, results from this formal study will be useful for developing computer software systems that support the processing of customer requirements.

In order to identify the base for representing customer requirements, some definitions are given as follows:

[Definition 5] An engineering system is the structure of an object (Ω) including both product (S) and its environment (E).

$$\Omega = E \cup S \quad (5)$$

According to (4), the engineering system ($\oplus \Omega$) can be represented as

$$\oplus \Omega = \oplus (E \cup S) = (\oplus E) \cup (\oplus S) \cup B, \quad (6)$$

where B is product-environment boundary defined as follows:

[Definition 6] Product environment boundary, denoted by B , is the collection of interactions between a product and its environment.

$$B = (E \otimes S) \cup (S \otimes E). \quad (7)$$

Graphically, an engineering system can be represented in Figure 1.

Figure 1. Engineering system

Product-environment boundary includes structural boundary and physical interactions. The structural boundary (B^s) is the shared physical structure between the product and its environment. The physical interactions include actions (B^a) of the environment on the product and responses (B^r) of the product to the environment. Therefore, product-environment boundary can be further represented as

$$B = B^s \cup B^a \cup B^r. \quad (8)$$

Based on the definition of engineering system, a design problem can be defined as follows.

[Definition 7] A design problem, denoted by p^d , is a proposition in which something has to be designed to meet the descriptions of a request for the design. Symbolically,

$$p^d = \lambda (\oplus \Omega_s, \oplus \Omega_0), \quad (9)$$

where $\oplus \Omega_s$ ($\Omega_s = E_s \cup S_s$) is "something to be designed" whereas $\oplus \Omega_0$ ($\Omega_0 = E_0 \cup S_0$) is "the descriptions of a request for the design". E_s and S_s are the environment and product by the end of the design process. E_0 and S_0 are the environment and product at the beginning of the design process. λ is the inclusion relation ($\supseteq$). A design problem is solved when p^d assumes the value of "true". In the stage of formulating a design problem, $\oplus \Omega_s$ is an unknown; $\oplus \Omega_0$ is the only thing defined. Obviously, $\oplus \Omega_s$ should include all the information in $\oplus \Omega_0$.

According to (6), we have

$$\begin{aligned} \oplus \Omega_0 &= (\oplus E_0) \cup (\oplus S_0) \cup B_0 \\ \oplus \Omega_s &= (\oplus E_s) \cup (\oplus S_s) \cup B_s. \end{aligned} \quad (10)$$

Finally, the following equation can be derived from (9) and (10) [15]:

$$P^d = \lambda(\oplus E_s, \oplus E_0) \wedge \lambda(\oplus S_s, \oplus S_0) \wedge \lambda(B_s, B_0). \quad (11)$$

Since the product environment does not change before and after the design problem is solved, the formulation in (11) implies following theorem:

Theorem of Design Problem Structure. A *design problem* is composed of three parts: the environment in which the designed product is expected to work, the requirements on product structure, and the requirements on the product-environment boundary.

This theorem can be shown in Table 1. The boundary requirements indeed include two parts: one is the requirements on the structural geometric boundary and another is on its performances. This can be seen intuitively from (8). The derivation is given in [15].

Table 1. Elements of design problem

Design Problem: P^d	
Product Environment	E_0
Boundary Requirements	$\lambda(B_s, B_0)$
Structural Requirements	$\lambda(S_s, S_0)$

2.3. Example

A rivet setting tool design example is adapted from the book by Hubka et al [16] to illustrate the concepts proposed in this paper. The task of this problem is to design a tool for riveting brake linings onto brake shoes for internal drum brakes as shown in Figure 2.

Figure 2. Internal drum brake [3].

The additional information regarding this design problem includes: the user of this tool is a car mechanic. The hand force, foot force, and working height should follow ergonomic standards. The use of this tool should conform to the related industry safety standards. The service life of this tool should be around 5 years. The tool should be easy for transportation and maintenance. The tool will be manufactured in a specific workshop, which has specified equipment. The cost of this tool cannot be over \$190.00.

According to theorem of design problem structure, rivet setting tool design problem is given in Table 2.

Table 2. Design problem: rivet setting tool design.

Rivet Setting Tool Design	
Product Environment	R-E1. Nature. R-E2. Mechanics. R-E3. Manufacturing shop. R-E4. Transportation facilities. R-E5. Brake shoe and lining.
Boundary Requirements	R-R1. To rivet brake linings onto brake shoes. R-R2. The hand and foot forces should follow ergonomic standards. R-R3. The use of tool should conform to related industry safety standards. R-R4. The tool can be manufactured in the specific workshop. R-R5. The service life of tool will be around 5 years. R-R6. The tool should be easy for transportation and maintenance. R-R7. The cost of tool cannot be over \$190.00
Structural Requirements	R-R8. The working height should follow ergonomic standards

3. Formalization of Customer Requirements

In engineering applications, customer requirements are usually informally described with natural language. The purpose of formalizing customer requirements is to identify product, environment, and their relations from its natural language-based description. Language is a symbolic system human beings used to describe the universe. By common agreement among its users, its symbols (letters and words) stand for *ideas in the mind* or *objects in the environment*, or they fulfill certain structural functions in the language pattern [17]. In delivering a message, sentences are the basic construct carrying the complete meaning. The axiomatic theory of design modeling is indeed a formal system aiming to code the universe through its objects and relations [8]. By comparing a language with the axiomatic theory of design modeling, it can be seen that both "ideas in the mind" and "objects in the environment" are objects in the universe. An implicit mapping exists between a natural language and the axiomatic theory of design modeling.

This section firstly develops a graphic scheme, called recursive object modeling (ROM), to represent the axiomatic theory of design modeling. Section 3.2 discusses the linguistic structure of English, followed by the formal structure in 3.3.

3.1. Recursive Object Modeling (ROM)

To apply the axiomatic theory of design modeling in formalizing engineering problems, the Recursive Object Modeling (ROM) is proposed to illustrate the modeling components graphically. This section will introduce these components that are relevant to the formalization of user requirements. The details of ROM will be introduced in a separate paper.

Word in a solid box represents a real-entity, which equals to a noun in English. For example:

$$\boxed{O_r}$$

$$\oplus O_r = O_r \cup (O_r \otimes O_r)$$

A solid arrow represents an interaction relationship, which equals to a transitive verb in English.

A solid arrow with a circle at its end represents a relationship that is indicated by prepositions, participles, and other connection words in English.

Word in a dashed box represents a virtual entity, which equals to adjectives, adverbs, or helping verbs etc.

A composite object is represented by a dashed box with other objects in it. For example:

Based on the above conventions, the theorem of design problem structure can be graphically illustrated in Figure 3.

Figure 3. Design problem structure

3.2. Linguistic Structure of User Requirements

3.2.1. English words and sentence patterns

English language has eight traditional parts of speech in its grammar: noun, verb, adjective, adverb, pronoun, preposition, conjunction, and interjection. For the sake of brevity, this paper will only formally discuss nouns and verbs.

A noun is a word used to name a person, place, thing, quality, idea, or action. In a sentence, it tells who or what did the action or was acted upon by the verb. Any noun in a natural language names an object in the universe. Pronouns are used as replacements or substitutes for nouns and noun phrases in a sentence.

A verb is a word used to indicate the action from/to/on an object or the state of an object. There are four principal verb types: helping, linking, intransitive, and transitive. Helping verb shades the meaning of the main verb in some desired manner. Linking verb links a noun to a complement to indicate the state of the noun. Intransitive verb itself indicates a state of a noun. Transitive verb shows actions from one object to another.

The basic English sentence takes the pattern: subject + predicate. The predicate may be only a verb or a verb plus other elements, such as complement, direct object, indirect

object, and objective complement. On the basis of the predicate structure, there are five basic sentence patterns:

Subject + intransitive verb

Subject + linking verb + subjective complement

Subject + transitive verb + direct object

Subject + transitive verb + indirect object + direct object

Subject + transitive verb + direct object + objective complement

3.2.2. Formal structure of linguistic components

No matter how complicated user requirements might be for a design problem, they can be ultimately structured into a group of sentences, which assume the above five patterns. On the other hand, as shown in Figure 1 and (11), any design requirement can be formulated as a constraint on certain part of engineering system. Therefore, if the relationship between those five sentence patterns and the engineering system can be established, then all design requirements can be logically formalized.

As in Figure 1, there are six objects in an engineering system: product, environment, two relations between the product and the environment, one relation on the product, and one relation on the environment. These six objects correspond to nouns in a sentence. To associate verbs to the engineering system, here gives again the definition of the relation in the context of axiomatic theory of design modeling. A relation is an aspect or quality that connects two or more objects as being or belonging or working together or as being of the same kind. Relation can also be a property that holds between an ordered pair of objects. Obviously, the first part of this definition corresponds to the linking verb whereas the second part the transitive and intransitive verbs. The following will formally associate the three types of verbs describing design requirements to the engineering system.

Linking verb

In describing a design requirement, a linking verb connects two nouns. It links the first noun to a complement, which is also a noun or a noun phrase, to form a complete sentence. In this case, the complement can be seen as a constraint defining the requirement while the first noun can be seen as the object being constrained. The requirement “the service life of the tool should be around 5 years” is such an example.

The object can be a part of the product or the environment. Using $\otimes_{lv}$ to represent the relation corresponding to a linking verb, p_{lv} for the sentence pattern 2, we have

$$p_{lv} \subseteq O_1 \otimes_{lv} O_2. \quad (12)$$

Intransitive verb

An intransitive verb only involves one object and has the form ‘noun verb’, so it describes a relation on itself, which is a state of the object.

“Spring deforms” is such an example. The intransitive verb can usually be viewed as a relation between two states of an object.

Using $\otimes_{iv}$ to represent the relation corresponding to an

intransitive verb, p_{iv} for the sentence pattern 1, we have

$$p_{iv} \subseteq O(t_1) \otimes_{iv} O(t_2), \quad (13)$$

where $O(t)$ is an object with the time t as its part.

Transitive verb

In the case of transitive verbs, the verb or the verb together with its direct object constitutes a relation between two objects.

An example of such a sentence pattern is “the rivet setting tool put rivets into an assembly of the brake lining and shoe”.

Using $\otimes_{tv}$ to represent the relation corresponding to a transitive verb, p_{tv} for the sentence patterns 3, 4, and 5, we have

$$p_{tv} \subseteq O1 \otimes_{tv} O2. \quad (14)$$

3.3. Formal Structure of User Requirements

This section will address three major steps in the formalization process of user requirements described in natural language. The three major steps are lexical analysis, syntactical analysis, and structure analysis.

3.3.1. Lexical Analysis

Language is a symbolic system that is built up on words. Lexical analysis is the first step in understanding a language. The purpose of this analysis is to identify the lexical property of words forming a design problem. It has four main steps: decomposition of sentence, reduction of word, determination of part of speech, and identification of phrase. The four steps are explained as follows.

At the stage of decomposition, a sentence is decomposed into single words with possible lexical attributes. Figure 4 gives an example from the rivet setting tool design.

Figure 4. Decomposition of sentence

The inflection of word is reduced to base form: 1) nouns have two forms, singular and plurality; 2) verbs have 7 forms: base form, infinitive, past simple, past participle, present participle, present simple and third person singular; and 3) adjectives and adverbs have three forms: base form, comparative degree and superlative degree. The lexicon server should be able to reduce them to the base form.

Part of speech of each word is determined. According to English Grammar [18], there are eight parts of speech including verb, noun, adjective, adverb, pronoun, preposition, conjunction, and interjection. Verbs, nouns, and adjectives can also be further classified. To analyze English

language more accurately, we have classified the parts of speech of words into 17 types.

Since some words may have multiple attributes, ambiguities exist in determining the part of speech for a word. For example, word ‘use’ can be a noun as well as a verb. However, in most cases, the attribute of a word can be determined by using some simple rules and its context. For example, an article is followed by a noun after ignoring the modifier. This rule can be expressed as a regular expression: article (adjective phrase | noun phrase)* noun.

Phrases are identified from words by using rules such as: 1) if a phrase has the form “noun A of noun B”, it is a noun phrase in which the noun A is the head-word and the noun B is a modifier; and 2) if a phrase has the form “helping verb + main verb”, it is a verb phrase and the main verb is the head-word. In terms of these two rules, we can find that ‘the use of the tool’ is a noun phrase and ‘should conform to’ is a verb phrase from the example shown in Figure 4.

3.3.2. Syntactic Analysis

The objective of the syntactic analysis is to identify the pattern of a sentence that describes a design problem. For the purpose of our study, an English sentence is analyzed based on an English grammar library called Gram-Lib. The English grammar library is composed of a set of basic sentence patterns that describe the relationship between subject, predicate, and object of a sentence. The syntactic analysis has two main steps: decomposition of complex sentence and identification of sentence patterns. These two main steps are explained as follows.

In the step of decomposition of complex sentence, a sentence may be decomposed into some basic sentence patterns. According to the principle ‘one sentence, one predicate’, a complex sentence is decomposed into a set of phrases and clauses which has and only has one verb. Furthermore, each simple sentence should be contained in the Gram-Lib. If some sentences cannot be analyzed, Gram-Lib should be extended.

Each decomposed sentence is compared against basic sentence patterns until a relevant one is found. Then, the sentence pattern is identified.

Table 3. Example: English sentence pattern

Phrases	Attribute	Function
The <u>use</u> of the tool	Common Noun	Subject
should <u>conform</u> to	Transitive Verb	Predicate
related industry safety <u>standards</u>	Common Noun	Object

Table 3 shows the parts of the sentence example given in Figure 4. It corresponds to the pattern of “Subject + Linking verb + Noun/Pronoun” in Gram-lib.

3.3.3. Structure Analysis

Structure Analysis is the final step toward the generation of the formal structure for a design problem described by natural language. Since the pattern of a sentence is identified after syntactic analysis, the objective of structure analysis becomes the mapping of syntactic elements into the objects in

an ROM model. The structure analysis consists of two steps. The first step is to map each simple sentence to an ROM diagram. The second step is to identify product, environment, and performance as well as structural requirements from the ROM diagram.

Table 4. ROM diagrams of each sentence

# 1	The tool rivets brake linings onto brakes shoes.	
# 2	The user of the tool is the car mechanic.	
# 3	The hand force, foot force, and the working height of the tool should follow ergonomic standards.	
# 4	The use of the tool should conform to related industry safety standards.	
# 5	The service life of the tool should be around 5 years.	
# 6	The tool should be easy for transportation and maintenance.	
# 7	The tool will be manufactured in a specific workshop, which has specified equipment	
# 8	The cost of the tool cannot be over CAN\$190.0	

In the context of engineering systems, a noun names either of product, environment, or the relations between them; a verb names the type of relation involved in a sentence. In the example of rivet setting tool shown in Table 2, there are eight sentences to describe the design problem. Each sentence is mapped into its corresponding ROM diagram shown in Table 4.

4. Software Prototype

4.1. Problem Formulation

The central task in solving a design problem is to specify user requirements. A ROM diagram is used to present the elements of requirements and the relationships between them.

We are developing a software system that can automatically translate user requirements described by natural language into a ROM diagram. The input of this software system is user requirements described in English and the corresponding ROM diagram is the output. In this section, we will present the architecture of a prototype system and some results from formalizing user requirements.

4.2. System Architecture

We have implemented a software system named ROMA (ROM analysis system). This system contains four modules: ROM Client, Lexicon Server, Syntax Server, Knowledge-Base Server. The architecture of this system is shown in Figure 5.

ROM Client is the user interface of the entire system. It does not involve core logical operations, whereas it invokes these operations from servers. The two main tasks of ROM Client are: to display the ROM diagram which had been analyzed by other modules, and to provide an interactive terminal that enables end users to participate in decision-making in the case of emerging ambiguity of various solutions. To a group of users, they are able to run ROM Client application individually to process respective requirement documents, and build the ROM diagram that belongs to themselves.

Lexicon Server is responsible for the reduction of single word, returning search results from ROM_Dictionary to ROM Client. ROM_Dictionary is the kernel of Lexicon Server, which includes 67181 base-form words, and 15141 inflected form words at present. To improve query performance, over 80 thousand words are saved in a two-dimension hash table. Meanwhile, for the purpose of running on multiple platforms, the Lex-Server is divided into three layers: core layer, interface layer, and application layer (Figure 6). All core operations are placed in the core layer, such as: loading the dictionary data from file or database, searching word from the dictionary, reducing the inflected word to base form, etc. An example of search results is shown in Figure 10.

Figure 5. Architecture of ROMA

Figure 6. Structure of Lexicon Server

Syntax Server is responsible for the analysis of sentence structure, mapping the sentence structure into ROM structure, and making them into XML data package, then sending the package to ROM Client. In this module, we adjust the technology of nested invoking between C++ and Prolog as shown in Figure 7. This kind of bi-directional invoking can benefit from each other. C++ has higher flexibility and performance as well as more friendly user interface. On the other hand, Prolog has powerful capacity in logical deduction. The core of Syntax Server consists of three components: determination of part of speech, identification of phrase, and identification of sentence pattern as is shown in Figure 5. All of them are implemented with Prolog. The results of syntax analysis are complex. To organize data structure efficiently, XML technology is introduced as data exchange format between Syntax server and ROM Client. The content of XML data package is shown in Figure 11.

The objective of *Knowledge-Base Server* is to process the sentence for which the pure syntactical analysis does not work. This part is still under development.

Since the foundation of this software system is the axiomatic theory of design modeling, a data structure is developed to describe the objects in this theory, as is shown in Figure 8.

Figure 7. Structure of Syntax Server

Figure 8. Hierarchical structure of ROM object

This hierarchical architecture describes essential elements of the world, such as the object and the relationship between objects in ROMA. CROM_Obj is an abstract class of all objects in the world. It has two subclasses: CROM_Entity and CROM_Relationship. CROM_Entity has two subclasses: CROM_RealEntity comes from noun in English Grammar and CROM_VirtualEntity comes from adjective, adverb etc. CROM_Relationship represents all kinds of relationships between CROM_Entities, and between CROM_Composites, which can be decomposed into CROM_Entities and CROM_Relationships at a lower level of granularity. In this structure, relationship is categorized into actions and modifications. Generally speaking, actions come from verbs and modifications come from sorts of relationship between modifiers and modified objects. Actions include three kinds of verb, intransitive verb, transitive verb that has one object, and transitive verb that has direct object and indirect object. Modifications include two branches. Some modification relationships are implied in context, such as adjectives modify nouns and adverbs modify adjective or verb. Other modification relationships are represented through preposition, participle, and other relationship words. For example, in phrase of 'the user of the tool', preposition 'of' represents the ownership between 'user' and 'tool'.

4.3. Experimental Results

Firstly, a user requirement document is separated into sentences and each sentence is decomposed into a series of single words. Each single word is reduced in Lexicon Server, and sent back to ROM Client. When we send a single sentence as input, Figure 9 shows the process of decomposing sentence and reducing single word. Figure 10

shows the result from Lexicon Server.

Figure 9. Analysis result of single sentence

Figure 10. Result of lexicon analysis

Figure 11. Result of syntax analysis

ROM Client sends Syntax Server a data package, which has only one sentence including the result of lexicon analysis. Each sentence is analyzed in Syntax Server, and the outcome of this step is a sentence structure, which is organized into XML package shown in Figure 11.

Based on the outcome of syntax analysis, ROM Client generates the ROM diagram in Figure 12. In this version, we

have not been able to analyze passive sentence, and this part will be the future job. Figure 12 lists the result from formulizing seven requirements for Table 2.

It can be seen from Figure 12 that tool is the product to be designed. Environment includes 'car mechanic', 'standards', 'brake shoes' and 'brake linings', etc. The interaction between the product and environment includes 'rivet', 'force', 'cost', 'life', etc. Some objects do not seem to fall into any category at this stage, such as 'transportation' and 'maintenance' because no corresponding environment has been explicitly stated in the problem description. This is up to the proceeding design process for the clarification of the design problem.

Figure 12. ROM diagram of example

5. Conclusion

In this paper, a formalization process is proposed to transform user requirements described in natural language into its formal structure. At the core of the formalization process are 1) lexical analysis; 2) syntactic analysis; and 3) structure analysis. A rivet setting tool design example is used to illustrate the process. Some progresses are reported from an on-going software prototype for testing the concepts.

Future work includes the extension of the library for sentence patterns, ontology for user requirement description, and the integration of geometric information.

Acknowledgement

This work is partially supported by NSERC (Grant number RGPIN 298255).

References

- [1] Agouridas, V., Baxter, J., de Pennington, A. and McKay, A., 2001, "On Defining Product Requirements: A Case Study in the Uk

- Health Care Sector," *Proceedings of ASME 2001 Design Engineering Technical Conferences and Computer and Information in Engineering Conference*, Pittsburgh, Pennsylvania pp. DETC2001/DTM-21692.
- [2] Gershenson, J. K. and Stauffer, L. A., 1999, "A Taxonomy for Design Requirements from Corporate Customers," *Research in Engineering Design*, **11**, pp. 103-115.
- [3] Lin, L. and Chen, L. C., 2002, "Constraints Modeling in Product Design," *Journal of Engineering Design*, **13**(3), pp. 205-214.
- [4] McKay, A., de Pennington, A. and Baxter, J., 2001, "Requirements Management: A Representation Scheme for Product Specifications," *Computer-Aided Design*, **33**, pp. 511-520.
- [5] Rounds, K. S. and Cooper, J. S., 2002, "Development of Product Design Requirements Using Taxonomies of Environmental Issues," *Research in Engineering Design*, **13**, pp. 94-108.
- [6] Farley, B., 2001, "Extracting Information from Free-Text Aircraft Repair Notes," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **15**, pp. 295-305.
- [7] Gangopadhyay, A., 2001, "Conceptual Modeling from Natural Language Functional Specifications," *Artificial Intelligence in Engineering*, **15**, pp. 207-218.
- [8] Zeng, Y., 2002, "Axiomatic Theory of Design Modeling," *Transaction of SDPS: Journal of Integrated Design and Process Science*, **6**(3), pp. 1-28.
- [9] Altshuller, G. S., 1984, *Creativity as an Exact Science: The Theory of the Solutions of Inventive Problems*, Gordon and Breach Science Publishers.
- [10] Cross, N., 1993, "Science and Design Methodology: A Review," *Research in Engineering Design*, **5**, pp. 63-69.
- [11] Eekels, J., 2000, "On the Fundamentals of Engineering Design Science: The Geography of Engineering Design Science. Part 1," *Journal of Engineering Design*, **11**(4), pp. 377-397.
- [12] Hubka, V. and Eder, W., 1996, *Design Science*, Springer.
- [13] Pahl, G. and Beitz, W., 1988, *Engineering Design: A Systematic Approach*, Springer-Verlag.
- [14] Roozenburg, N. F. M. and Eekels, J., 1995, *Product Design, Fundamentals and Methods*, Chichester: Wiley.
- [15] Zeng, Y., 2004, "Mathematical Formulation of Design Problems," *Journal of Engineering Design*, submitted.
- [16] Hubka, V., Andreasen, M. and Eder, W., 1988, *Practical Studies in Systematic Design*, Butterworths.
- [17] Turner, R., 1971, *Grammar Review for Technical Writers*, Rinehart Press.
- [18] Schmidt, H., 1995, *Advanced English Grammar*, Prentice Hall.