Formalisation of product requirements: from natural language descriptions to formal specifications

Zhen Yu Chen, Shengji Yao
and Jian Qiang Lin

Concordia Institute for Information Systems Engineering
Concordia University
1455 de Maisonneuve West, EVO07.640
Montreal, QC, H3G 1M8, Canada

Yong Zeng*

Concordia Institute for Information Systems Engineering
Concordia University
1455 de Maisonneuve West, EVO07.633
Montreal, QC, H3G 1M8, Canada
Fax: +1–514–848–3171
E-mail: zeng@ciise.concordia.ca
*Corresponding author

Armin Eberlein

School of Engineering
American University of Sharjah
P.O. Box 26666
Sharjah, United Arab Emirates

Abstract: In engineering design, customers usually provide product requirements in the form of a natural language while computer-aided design systems may prefer more formal and structured specifications. In this paper, a formalisation process is proposed to transform product requirements from its natural language descriptions to a formal specification. The formal specification is based on the product environment and the formulation of design problem, which identifies the components included in a design problem in terms of the product environment. Through the lexical, syntactic, and structure analysis of natural language descriptions of a design problem, the formalisation process identifies the product to be designed, its environment components, and their relations. A software prototype is developed to validate the formalisation process. An example of rivet setting tool design shows that both the formalisation process and software prototype are feasible.

Keywords: product requirements; formalisation; formulation; natural language; axiomatic theory of design modelling.

Reference to this paper should be made as follows: Chen, Z.Y., Yao, S., Lin, J.Q., Zeng, Y. and Eberlein, A. (2007) 'Formalisation of product requirements: from natural language descriptions to formal specifications', *Int. J. Manufacturing Research*, Vol. 2, No. 3, pp.362–387.

Biographical notes: Zhen Yu Chen received his Master's degree from Department of Electrical and Computer Engineering at Concordia University, Montreal, Canada in 2006. His research interests include requirements engineering, service engineering, and image processing. He received his BEng. degree from Zhejiang University of Technology in 1994. Prior to his graduate studies, he worked as a software developer and project manager in the telecommunication and manufacturing industries in China for ten years.

Shengji Yao is a PhD candidate in Mechanical Engineering at Concordia University, Canada. Her research interests encompass exploring product design innovation by cognitive experiment and specification of product requirements. Yao received her Master of Applied Science in General Mechanics from Dalian University of Technology, China and her Bachelor of Engineering in Industrial Automation at Jiangxi University of Science and Technology in 2001 and 1998, respectively.

Mr. Jian Qiang Lin is a Master's student in the Concordia Institute for Information Systems Engineering at Concordia University, Montreal, Canada. His research interests include requirements engineering, formal methods, and natural language processing. He received his BEng. degree from Tsinghua University in 1993.

Dr. Yong Zeng is an Associate Professor in the Concordia Institute for Information Systems Engineering at Concordia University, Montreal, Canada. He is Canada Research Chair in Design Science. He received his BEng degree in structural engineering from the Institute of Logisticals Engineering and MSc and PhD degrees in computational mechanics from Dalian University of Technology in 1986, 1989, and 1992, respectively. He received his second PhD degree in design engineering from the University of Calgary in 2001. His research interests include the science of design, requirements engineering, human factors engineering, computer-aided product development, computational geometry, and finite element modelling.

Dr. Armin Eberlein received the Dipl-Ing (FH) in Telecommunications Engineering at the Mannheim University of Applied Sciences in Germany, the MSc in Communications Systems, and the PhD in Software Engineering from the University of Wales, Swansea, UK. He is currently an Associate Professor and Head of the Computer Engineering Department at the American University of Sharjah in the UAE. Previously, he was an Associate Professor at the University of Calgary, Canada, where he was the Director of the Software Engineering programme and one of a Co-Director of the Alberta Software Engineering Research Consortium (ASERC). His research focuses on the improvement of requirements engineering practices.

1 Introduction

A design process includes several major stages: gathering of product requirements, requirements specification, conceptual design, configuration design, and detailed design. At the beginning of a product design process, based on a request made by customers, designers start to identify the customer's real intent in order to gather all explicit and implicit requirements that the product has to satisfy. The gathered product requirements are then specified to support the following design process. During the conceptual design

stage, some design concepts will be generated and evaluated to satisfy the specified product requirements. After that, configuration as well as detailed design transforms and refines the generated concepts into product descriptions that can guide the implementation of the design solution. Over the last few decades, a great deal of efforts has been made to support the configuration and detailed design. Relatively, early design stages are not well investigated. Especially, the gathering and specification of product requirements are not well understood. However, study has shown that 70% of the cost of a product is determined by the decisions made in the early design stages. Poor decisions at the early stages of product development may require significantly more efforts in the late stages or may cause the project to be cancelled. As the first step in the product design process, the correct identification and specification of product requirements is especially critical. A systematic approach would be beneficial to help designers to specify the product requirements in a proper manner.

Some researchers defined the conceptual design process as the mapping of well-defined functional requirements to physical properties (Suh, 1990; Braha and Maimon, 1998; Yoshikawa, 1981). However, the mapping of the customer requirements to functional requirements is not adequately discussed. Other researchers developed steps and procedures for identifying product requirements based on empirical experience from experts (Hubka and Eder, 1988; Pahl and Beitz, 1988). Among them, Quality Function Deployment (QFD) is a typical method to transform customer requirements into engineering specifications (Akao, 1990). In recent years, a great deal of efforts has been made to elicit and organise customer requirements implied in a design problem (Agouridas *et al.*, 2001; Chittaro and Kumar, 1998; Deng *et al.*, 2000; Gershenson and Stauffer, 1999; Hirtz *et al.*, 2002; Jiao *et al.*, 1998; Jiao and Tseng, 2004; Lossack *et al.*, 1998; McKay *et al.*, 2001; Rounds and Cooper, 2002; Chen *et al.*, 2002; 2005). Gangopadhyay (2001) utilised the theory of conceptual dependencies (Schank, 1975) to develop a conceptual data model from functional specifications expressed in natural languages. McKay *et al.* (2001) proposed a specification data model to define a structure for product specification using the EXPRESS data specification language and demonstrated its validity through the use of a mechanical product case study. Stacey *et al.* (1999) proposed a modelling approach, known as Signposting, and developed suitable methods to include route selection and process simulation in order to identify the best route through the design process and to optimise resource allocation. Chen *et al.* (2005) proposed a prototype of a Product Definition and Customization System (PDCS) comprising two phases, namely, product definition using the laddering technique and a novel Design Knowledge Hierarchy (DKH) and product customisation using an integrated methodology of Conjoint Analysis (CA) and Kohonen Association (KA) techniques. Jiao and Tseng (2004) identified two sources of customisability, namely, design changes and process variations and evaluated the cost effectiveness of a design to be customised in order to meet individual customer needs.

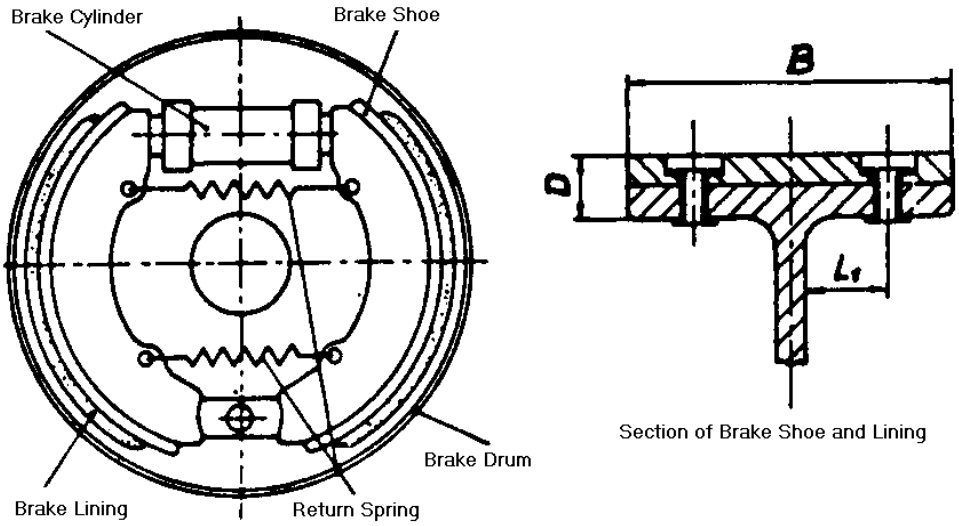
Although existing approaches attempt to propose different representations in modelling customer requirements, constraints, design tasks, design intent, design goals and objectives to better understand a design problem, few can tackle all these aspects within a systematic and integrated framework. Moreover, the generic relationships among different requirements are not shown in the structure of specified product requirements. Some challenges remaining in specifying product requirements include:

- Product requirement documents are usually written in natural language (Chen, 1983), which easily leads to ambiguous or distorted understanding of the customer's original intent (Oxman, 2004). This makes the management of customer requirements challenging.
- Product requirements often include many different types of information. This may easily confuse and frustrate both designers and requirement providers (Darlington and Culley, 2004).
- In developing a product family from an original product, a variety of new product requirements are usually introduced (Jiao and Tseng, 2004). It is difficult to predict what type of requirements may appear in the product development process. This requires that a flexible structure be used to manage product requirements (Zeng, 2004a).

To solve the problems above, Zeng (2004a) derived an environment-based formulation of design problem, based on which Chen and Zeng (2006) further classified product requirements in terms of product life cycle and the level of priority. The derived formal representation (Zeng, 2004a; Chen and Zeng, 2006) of product requirements provides a foundation for a new design process model (Zeng, 2004b), which addresses the simultaneous evolution of design problem and design solutions. To integrate the product requirement in the design process model, this paper studies how natural language description of product requirements can be formalised into the formal environment-based formulation of design problem.

Section 2 will summarise our research results in formulating design problem, based on which a formalisation process is proposed in Section 3 to transform natural language descriptions into formal specifications. Section 4 presents a software prototype we have developed to validate the formalisation process. Section 5 is the conclusion of this paper. Future research directions are also pointed out in this section.

Figure 1 Internal drum brake



Source: Hubka *et al*. (1988)

To illustrate the concepts presented in this research, a rivet setting tool design example is adapted from the book by Hubka *et al.* (1988) 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 1. 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 five years. The tool should be easy for transportation and maintenance. It will be manufactured in a specific workshop, which has specified equipments. The cost of this tool cannot be over $190.00.

2 Formulation of product requirements

This section aims to provide the theoretical background for formalising product requirements. Firstly, we will introduce some fundamental concepts in the axiomatic theory of design modelling for the formulation and formalisation of product requirements. Secondly, two theorems about the design problem will be presented to show the formal structure of the design problem, which provides a formal framework for representing product requirements.

2.1 Axiomatic theory of design modelling

Axiomatic theory of design modelling is a logical tool for representing and reasoning about object structures (Zeng, 2002), which is different from Suh's (1990) theory of axiomatic design. It provides a formal approach that allows for the development of design theories following logical steps based on mathematical concepts and axioms. The primitive concepts of universe, object, and relation are used in the axiomatic theory of design modelling. 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.*

$$R = A \sim B, \exists A, B, R, \tag{1}$$

where *A* and *B* are objects. A~B is read as '*A* relates to *B*'. *R* is a relation from object *A* to object *B*. Basic properties of relations include idempotent, commutative, transitive, associative and distributive.

Based on these concepts, two axioms are defined in the axiomatic theory of design modelling.

[Axiom 1] 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 play a critical role in the axiomatic theory of design modelling. We need to define a group of basic relations to capture the nature of object representation. Two corollaries of the axiomatic theory of design modelling are used to represent various relations in the universe.

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

$$\mathrm{A} \supseteq \mathrm{B}, \forall \mathrm{A}\ \exists \mathrm{B}, \tag{2}$$

where *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,

$$\mathrm{C} = \mathrm{A} \otimes \mathrm{B}, \forall \mathrm{A}\ \exists \mathrm{B}, \mathrm{C}, \tag{3}$$

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 the Corollary 1 and 2, the structure operation is developed.

Definition 4 *Structure operation, denoted by* $\oplus$, *is defined by the union of an object and the interaction of the object with itself.*

$$\oplus\ \mathrm{O} = \mathrm{O} \cup (\mathrm{O} \otimes \mathrm{O}), \tag{4}$$

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 Product system

Based on the structure operation, the concept of product system is introduced.

Definition 5 *A product system is the structure of an object* (Ω) *including both a product (S) and its environment (E).*

$$\Omega = \mathrm{E} \cup \mathrm{S}, \forall \mathrm{E},\ \mathrm{S}\,[\mathrm{E} \cap \mathrm{S} = \Phi], \tag{5}$$

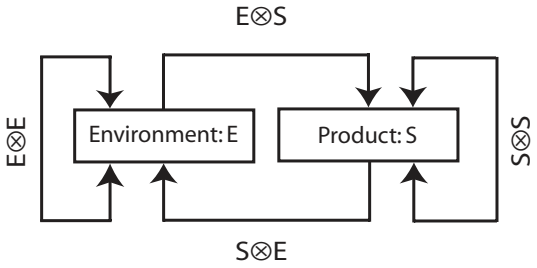
where Φ is the object that is included in any object.

The product system ($\oplus\ \Omega$) can then be expanded as follows:

$$\oplus\Omega = \oplus(\mathrm{E} \cup \mathrm{S}) = (\oplus\mathrm{E}) \cup (\oplus\mathrm{S}) \cup (\mathrm{E} \otimes \mathrm{S}) \cup (\mathrm{S} \otimes \mathrm{E}), \tag{6}$$

where $\oplus$E and $\oplus$S are structures of the environment and product, respectively; E$\otimes$S and S$\otimes$E are the interactions between environment and product. A product system can be illustrated in Figure 2.

Figure 2 Product system



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

$$B = (E \otimes S) \cup (S \otimes E). \tag{7}$$

There are two types of product boundary: structural boundary and physical interactions. The structural boundary (B^s) is the common physical structure between a 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 its environment. Therefore, product environment boundary can be further represented as:

$$B = B^s \cup B^a \cup B^r, \forall B^s, B^a, B^r[(B^s \cap B^a = \Phi) \wedge (B^s \cap B^r = \Phi) \wedge (B^a \cap B^r = \Phi)]. \tag{8}$$

Since environment as well as product may have components, structures $\oplus E$ and $\oplus S$ can be further decomposed into the structures of these components as well as their mutual interactions according to the definition of structure operation. Equation 6 indeed presents a recursive structure of a product system.

2.3 Formal structure of design problem

A design problem can be literally defined as a request to design something that meets a set of descriptions of the request. Based on the axiomatic theory of design modelling, both 'something' and 'descriptions of the request' can be seen as objects and can be further seen as product systems in the context of formulating design problem. Thus a design problem, denoted by P^d, can be formally represented as:

$$P^d = \lambda(\oplus\Omega_0, \oplus\Omega_s), \tag{9}$$

where $\oplus\Omega_0$ ($\Omega_0 = E_0 \cup S_0$, $E_0 \cap S_0 = \Phi$) can be seen as the descriptions of a request for the design, $\oplus\Omega_s$ ($\Omega_s = E_s \cup S_s$, $E_s \cap S_s = \Phi$) is something to be designed, and λ is the 'inclusion' relation ($\supseteq$) implying that $\oplus\Omega_s$ will be a part of $\oplus\Omega_0$ so that the designed product will meet the descriptions of the design. Obviously, if $\oplus\Omega_s$ is a part of $\oplus\Omega_0$, then Equation (9) is satisfied. At the beginning of the design process, $\oplus\Omega_s$ is an unknown and $\oplus\Omega_0$ is the only thing defined. The truth value of P^d is undetermined, which means the request is yet to be met.

According to Equation (6) and Equation (7), we have:

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

Based on Equations (8), (9) and (10), the following equation can be derived using the axiomatic theory of design modelling (Zeng, 2004a).

$$P^d = \lambda(\oplus E_0, \oplus E_s) \wedge \lambda(\oplus S_0, \oplus S_s) \wedge \lambda(B_0^s,\ B_s^s) \wedge \lambda(B_0^a,\ B_s^a) \wedge \lambda(B_0^r,\ B_s^r), \tag{11}$$

where the symbol $\wedge$ denotes logical 'and'; $\lambda(\oplus E_0, \oplus E_s)$, $\lambda(\oplus S_0, \oplus S_s) \wedge \lambda(B_0^s,\ B_s^s)$ and $\lambda(B_0^a,\ B_s^a) \wedge \lambda(B_0^r,\ B_s^r)$ represent requirements on product environment, structural requirements, and performance requirements, respectively.

In general, the product environment can be partitioned into a finite number of sub-environments:

$$E = \bigcup_{i=1}^{n} E_i, \exists E_1, E_2, \cdots, E_n, \forall i, j, i \neq j, E_i \cap E_j = \Phi, \tag{12}$$

where n is a finite positive number. Each E_i can be an individual environment. This partition of product environment partitions the product requirements in Equation (11). Since product environment E_0 itself can be taken as an invariant of a design problem and $\lambda(\oplus S_0, \oplus S_s)$ can be viewed as an implication of $\lambda(B_0^s,\ B_s^s)$, then we can get:

$$P^d = \left[\lambda\left(\bigcup_{i=1}^{n} B_0^i, \bigcup_{i=1}^{n} B_s^i \right) \right], \forall E = \bigcup_{i=0}^{n} E_i, \tag{13}$$

where both B_0^i and B_s^i are defined as follows:

$$\begin{aligned} B_0^i &= (E_i \otimes S_0) \cup (S_0 \otimes E_i), \\ B_s^i &= (E_i \otimes S_s) \cup (S_s \otimes E_i). \end{aligned} \tag{14}$$

Equations (11), (13) and (14) imply (15) the following two theorems (Zeng, 2004a):

1 *Theorem of structure of design problem*. A design problem is implied in a product system and 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 performance of the designed product.

2 *Theorem of source of product requirements*. All the product requirements in a design problem are imposed by the environment in which the product is expected to work.

For the rivet setting tool design, the design problem is given in Table 1.

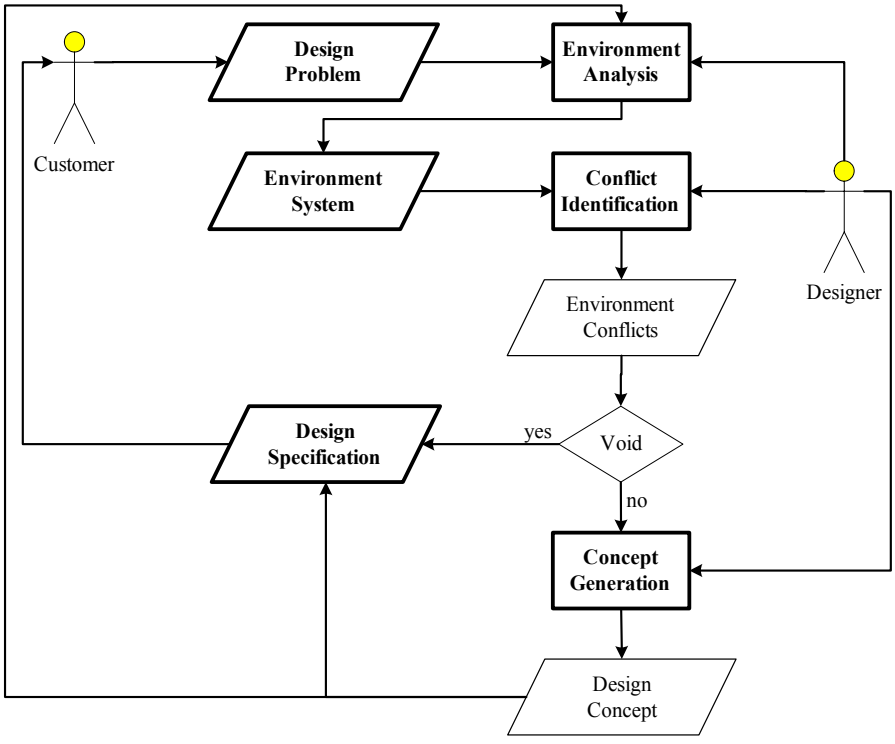
These two theorems define an invariant part of a design problem, which is the product environment. All components in a design problem can be defined through the product environment. Therefore, product environment provides a foundation for the classification and management of product requirements.

On the basis of these two theorems, a new design methodology named Environment-Based Design (EBD) is proposed by Zeng to accomplish a design task from requirements gathering to solution generation. Different from traditional design methodologies, which are largely based on the understanding that a generic design process comprises analysis, synthesis, and evaluation, the environment-based design methodology includes three main stages: environment analysis, conflict identification, and concept generation. These three stages work together to progressively and simultaneously generate and refine the design specifications and design solutions. The EBD methodology is illustrated in Figure 3.

Table 1 Design problem: rivet setting tool design

Product	*Tool*
Product environment	E1: Nature
	E2: Mechanics
	E3: Manufacturing shop
	E4: Transportation facilities
	E5: Market
	E6: Brake: linings and shoe
Performance 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 the 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 the tool will be around five years.
	R-R6. The tool should be easy for transportation and maintenance.
	R-R7. The cost of the tool cannot be over $190.00.
Structural requirements	R-R8. The working height should follow ergonomic standards.

Figure 3 Environment-based design: process flow



The objective of environment analysis is to find out the key environment components, in which the product works, and the relationships between the environment components. From the environment implied in the design problem described by the customer(s), the designer will introduce extra environment components that are relevant to the design problem at hands. The results from this analysis constitute an environment system. One of the key methods for environment analysis is linguistic analysis. Following the environment analysis, conflicts should be identified among the relations between environment components. At the third stage of EBD, a set of key environment conflicts will be chosen to be resolved by generating some design concepts. This process continues until no more unacceptable environment conflicts exist.

The environment based design is two things. First, it is a prescriptive model of design (which is a design methodology) that guides designers from the gathering of customer requirements throughout the generation and evaluation of design concepts. Secondly, it is a descriptive model of the natural design process that illustrates how designers conduct a design task.

As a part of the first step in the Environment-Based Design methodology, the natural language based design problem description needs to be transformed into a formal environment system. This environment system includes product environment as well as the relations between product and environment. Based on Equations (11), (13) and (14), the identification of the environment is a key to specify product requirements. Chen and Zeng (2006) discussed the potential environment components that may contribute to the product requirements. The rest of this paper will focus on a systematic approach to identifying the environment system from the natural language description of a design problem.

3 Formalisation of product requirements

Some formal methods such as Z (Spivey, 1988), VDM (Bjorner and Jones, 1982), and Larch (Guttag and Horning, 1993) focus on specifying the behaviour of sequential systems. States are described in terms of rich mathematical structures such as sets, relations, and functions; state transitions are given in terms of pre- and post-conditions. They are often called abstract model methods. Other methods, such as Communicating Sequential Processes (CSP) (Hoare, 1985), Calculus of Communicating Systems (CCS) (Milner, 1982), state charts (Harel, 1987), temporal logic (Manna and Pnueli, 1991), and I/O automata (Lynch and Tuttle, 1987), focus on specifying the behaviour of concurrent systems. The method proposed in this paper does not distinguish between sequential and concurrent systems, but rather focuses on the structure of a design problem.

3.1 Statement of the problem

In engineering applications, design problems are usually described using natural language. It is not feasible to force designers and customers to describe design problems in the form of mathematical formulation introduced in Section 2. Hence, it is essential to establish a step-by-step process to formalise a design problem described by natural language into its formal specification. This formalisation process will help designers and customers to easily and accurately identify product requirements implied in a design

problem description. Though some researchers from the field of software engineering have also studied the natural language based requirements specification (Fliedl *et al.*, 2004; Liu *et al.*, 2004), their intention is to output the standard format such as UML. Product requirements represented in those formats do not well support current design methodology. The relation between the requirements specification and the succeeding design process is not closely established.

As indicated in Section 2, product requirements can be found by analysing the structure of design problems. If it is intended to find the formal specification of product requirements from the natural language descriptions of a design problem, the formal structure of natural language needs to be developed in the first place. Language is a symbol system human beings used to describe the universe (Turner, 1971). By common agreement among its users, its symbols (letters and words) usually stand *for ideas in the mind* or *objects in the environment*. The symbols in a language may also fulfil certain structural functions in the language pattern so that ideas and objects can be combined to form more complex meanings (Turner, 1971). In delivering a message, sentences are the basic construct carrying the complete meaning. The axiomatic theory of design modelling is indeed a formal system aiming to code the universe through its objects and relations (Zeng, 2002). By comparing a language with the axiomatic theory of design modelling, it can be seen that both 'ideas in the mind' and 'objects in the environment' are objects in the universe. The 'structural functions' are indeed relations between words. Thus an implicit mapping exists between a natural language and the axiomatic theory of design modelling. Hence, the linguistic structure of words and sentences will be built using the axiomatic theory of design modelling. When all the sentences in a design problem description are transformed into a formal structure, the product requirements are specified by identifying product, environment and their relations implied in the formal structure. The general procedure of transforming the natural language description into the formal specification of design problem is given in Figure 4.

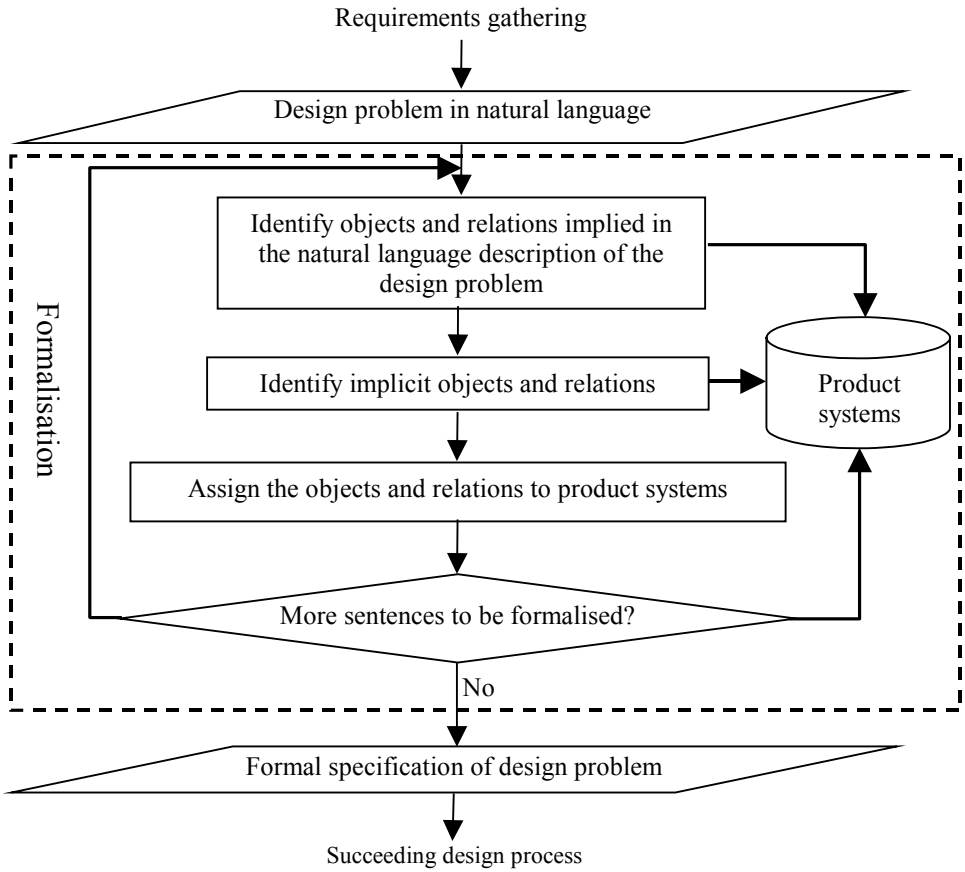
In this section, we firstly discuss the linguistic structure of English sentences and represent words and sentences graphically using the axiomatic theory of design modelling. Then a formal structure representing product requirements is built up through lexical analysis, syntactic analysis and structure analysis. The example of rivet setting tool is used to illustrate the concepts.

3.2 Representation of linguistic structure

Although the descriptions of product requirements might be complex, they can be ultimately structured into a group of sentences and sentences can be decomposed into words. In this paper, we only discuss the English linguistic structure. For words, the 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.

Figure 4 Formalisation process of design requirements



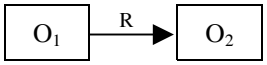
A verb is a word used to indicate the action from/to/on an object or the state of an object. There are four principal verb types: helping, linking, intransitive, and transitive. Helping verb shades the meaning of the main verb in some desired manner. A linking verb connects two nouns. It links the first noun to a complement, which is also a noun or a noun phrase to indicate the state of the noun. An intransitive verb only involves one object. It describes a relation on itself, which indicates a state of a noun. Transitive verb shows actions from one object to another. The verb or the verb together with its direct object constitutes a relation between two objects.

The following graphic symbols are used to represent nouns and verbs, corresponding to objects and relations in the axiomatic theory of design modelling.

A word surrounded by a solid line box represents a concrete-entity that equals a noun in English:

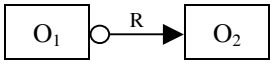
$$\boxed{\text{O}} \qquad \oplus \text{O} = \text{O} \cup (\text{O} \otimes \text{O}).$$

An arrow with a solid line represents an action relationship that equals a transitive verb in English:



$R = O_1 \otimes O_2$.

An arrow with a solid line attached by a circle at the other end represents a modification relationship that is indicated by prepositions, participles, and other relationship words in English.

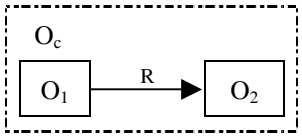


$R = O_1 \otimes O_2$.

Word surrounded by a dash-line box represents an abstract entity that equals adjectives, adverbs or helping verbs, *etc.* Usually, an abstract entity is used with a modification relationship flag.

$O_1 = O_2 \otimes O_2$.

A dash-dot line box represents a composite object, which consists of other kinds of objects.



$\oplus O_c = \oplus O_1 \cup \oplus O_2 \cup R$

where $R = O_1 \otimes O_2$.

A 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:

1 Subject + intransitive verb

2 Subject + linking verb + subjective complement

3 Subject + transitive verb + direct object

4 Subject + transitive verb + indirect object + direct object

5 Subject + transitive verb + direct object + objective complement.

Using $\otimes_{iv}$ to represent the relation corresponding to an intransitive verb, p_{iv} for the sentence pattern 1, we have:

$$p_{iv} \subseteq O(t_1) \otimes_{iv} O(t_2). \tag{15}$$

Using $\otimes_{lv}$ to represent the relation corresponding to a linking verb, p_{lv} for the sentence pattern 2, we have:

$$p_{lv} \subseteq O_1 \otimes_{lv} O_2. \tag{16}$$

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

$$p_{tv} \subseteq O_1 \otimes_{tv} O_2. \tag{17}$$

3.3 Formal structure of product requirements

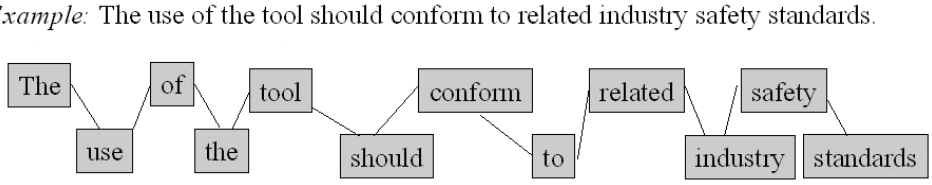
After the linguistic analysis of sentences is conducted, we need to generate a formal structure to specify product requirements. The input of this formalisation process is the product requirements described in natural language. The output is a structure showing the product, environment and their relations, which can be used for the rest of the design process. The formalisation process will address three major steps: lexical analysis, syntactic analysis, and structure analysis. Lexical analysis is the first step in understanding a language by determining the property of a word in a sentence. Syntactic analysis analyses the role of a word or a phrase in a sentence and identifies the sentence patterns. The formal structure underlying product requirements is generated by structure analysis.

3.3.1 Lexical analysis

Words are fundamental units in every sentence. Lexical analysis identifies word's lexical properties, such as attribute, base form, function, and related phrases. It is the prerequisite step for the following syntactic analysis. Lexical analysis starts from a sentence. It includes decomposition of the sentence, reduction of the form of a word, determination of the part of speech and identification of phrases.

Firstly, a sentence is decomposed into single words. Figure 5 gives an example of decomposing the sentence of 'The use of the tool should conform to related industry safety standards'. It has 12 words.

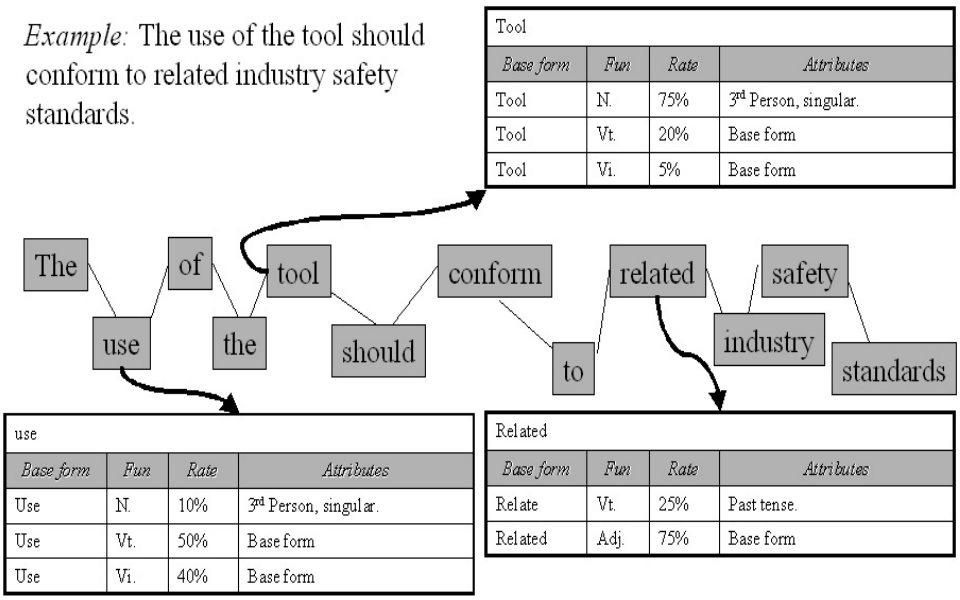
Figure 5 Decomposed words in a sentence



Then the part of speech for each word is determined. According to English Grammar (Schmidt, 1995), there are eight major parts of speech including verb, noun, adjective, adverb, pronoun, preposition, conjunction, and interjection. Verbs, nouns, and adjectives can be further classified into specific parts of speech. Since some words may have multiple attributes, confusion may exist in determining the part of speech for a word. For example, the word 'use' can be a noun as well as a verb. In the above example, 'use' must be a noun since it follows the article word 'the'.

Nouns, verbs, adjectives and adverbs often take inflections. When these words are determined by their parts of speech, the form of each word shall be identified and be reduced to its basic form. For nouns, there are two forms, singular and plural. Verbs have six forms: base form, infinitive, past simple, past participle, present participle, present simple forms. Adjectives and adverbs have three forms: base form, comparative degree, and superlative degree. Figure 6 shows a word's part of speech (in the column 'Fun'), its inflection (in the column 'attributes'), its base form, and the rate of its use recorded in a database.

Figure 6 Lexical analysis



Tool

Base form	*Fun*	*Rate*	*Attributes*
Tool	N.	75%	3rd Person, singular.
Tool	Vt.	20%	Base form
Tool	Vi.	5%	Base form

use

Base form	*Fun*	*Rate*	*Attributes*
Use	N.	10%	3rd Person, singular.
Use	Vt.	50%	Base form
Use	Vi.	40%	Base form

Related

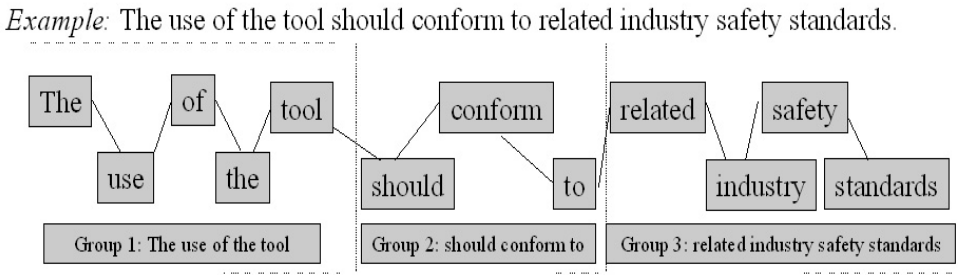
Base form	*Fun*	*Rate*	*Attributes*
Relate	Vt.	25%	Past tense.
Related	Adj.	75%	Base form

Phrases are used to group a set of words. They are identified from words by using rules such as:

- 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
- 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 7.

Figure 7 Identification of phrases



The property of a word should also include the position of a word in a sentence, such as subject, predicate or object, but the position of a word can only be determined when the sentence structure is considered. This will be discussed in syntactic analysis.

3.3.2 Syntactic analysis

After the lexical property of a word is identified, syntactic analysis will be conducted to identify the patterns of sentences that describe a design problem. The pattern of a sentence is analysed based on English grammar. The English grammar consists of a set of basic sentence patterns as shown in Table 2, which describe the relationship between subject, predicate, and object of a sentence.

Table 2 Basic sentence patterns

Pattern class	*Pattern subclass*	*No.*
Subject + Vi.	S + Vi	1
	S + Vi. + Adv.	2
	S + Vi. + Prep. Phrase	3
	S + Vi. + Infinitive	4
	S + Vi. + Participle	5
Subject + predicate + object.	S + Vt. + N./Pron.	6
	S + Vt. + Infinitive	7
	S + Vt. + Wh-Word + Infinitive	8
	S + Vt. + Gerund	9
	S + Vt. + That-clause	10
Subject + linking verb + predicative.	S + Lv + N./Pron.	11
	S + Lv + Adj	12
	S + Lv + Adv	13
	S + Lv + Prep Phrase	14
	S + Lv + Participle	15
Subject + predicate + direct obj + indirect obj.	S + Vt. + N./Pron. + N.	16
	S + Vt. + N./Pron. + To/for-phrase	17
Subject + predicate + object + complement.	S + Vt. + N./Pron. + Adj	18
	S + Vt. + N./Pron. + Prep Phrase	19
	S + Vt. + N./Pron. + Infinitive	20
	S + Vt. + N./Pron. + Participle	21
	S + Vt. + N./Pron. + Wh-word + Infinitive	22
	S + Vt. + N./Pron. + That-clause	23
	S + Vt. + N./Pron. + Wh-Clause	24

In syntactic analysis, a complex sentence is first decomposed into a set of phrases and clauses. In a phrase, the key words need to be found. They are usually verbs and nouns, which are modified by other words. Each clause can be taken as a simple sentence and has only one verb. According to the principle of 'one sentence, one predicate', a complex sentence is made up of several simple sentences. Then each decomposed sentence is compared against the basic sentence patterns until a relevant one is found. Basically, the pattern of each simple sentence is contained in Table 2. If some sentences cannot be analysed, the patterns in Table 2 should be extended. Table 3 shows the decomposed phrases, the attributes of keywords (which are underlined), and the functions of respective phrases. This sentence corresponds to the pattern of 'Subject + Linking verb + Noun/Pronoun' in Table 1.

Table 3 Decomposed sentence

Phrases	*Attribute*	*Function*
The *use* of the tool	Common noun	Subject
should *conform* to	Transitive verb	Predicate
related industry safety *standards*	Common noun	Object

3.3.3 Structure analysis

Structure analysis is the final step to generate the formal structure for a design problem described by natural language. After lexical analysis and syntactic analysis, the pattern of a sentence and the property of words in a sentence are identified. The next step is to convert each sentence into a formal structure defined by the axiomatic theory of design modelling, based on the results of lexical analysis and syntactic analysis. This formal structure reflects the relations between words. When a set of sentences representing product requirements are structured, we can identify the product, environment, and their mutual relations.

The structure analysis consists of two steps. The first step converts each simple sentence into a diagram using the representation of linguistic structure discussed earlier. In the example of the rivet setting tool design shown in given at the end of Section 1, eight sentences are used to describe the product requirements for the design problem. Each sentence is mapped into its corresponding diagram shown in Table 4.

Table 4 Formalisation of product requirements

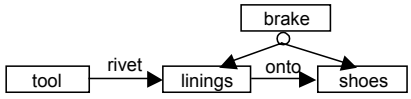
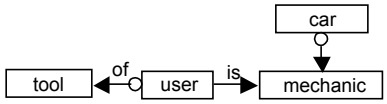
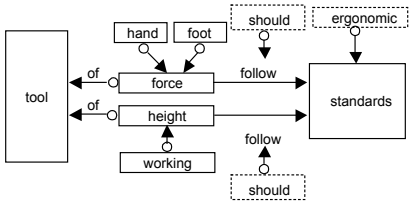
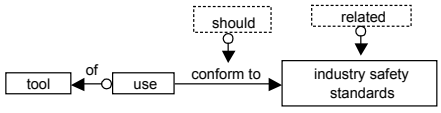
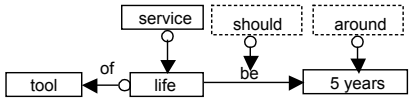
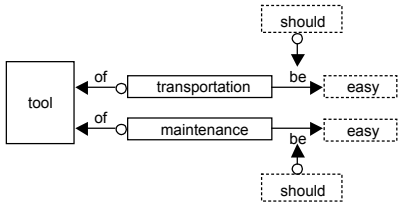
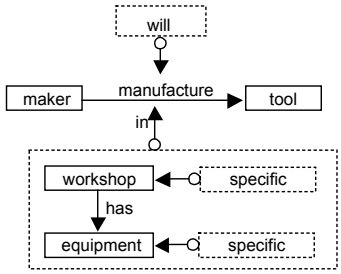
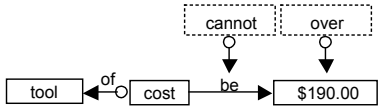
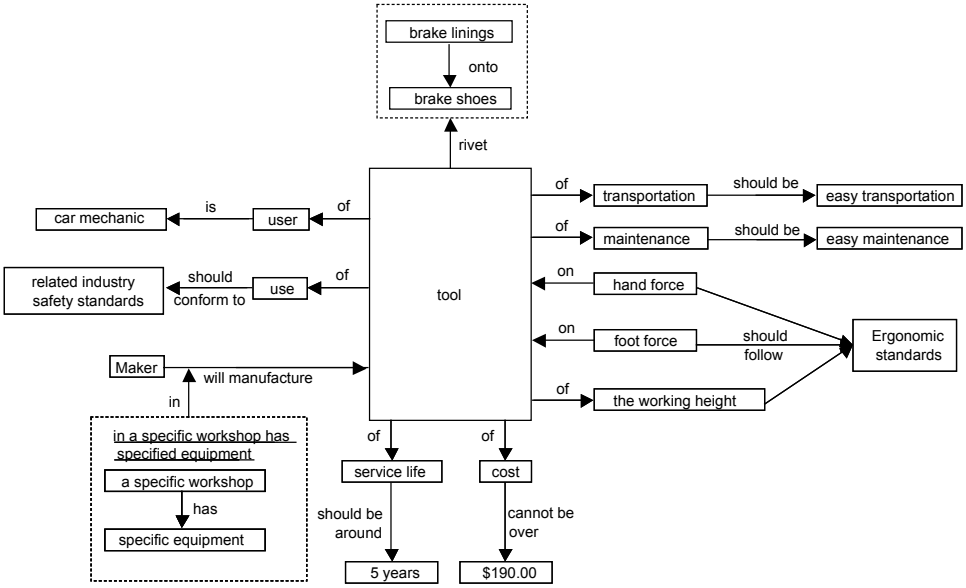
#1	The tool rivets brake linings onto brakes shoes.	
#2	The user of the tool is the car mechanic.	
#3	The hand force, foot force, and the working height of the tool should follow ergonomic standards.	
#4	The use of the tool should conform to related industry safety standards.	
#5	The service life of the tool should be around 5 years.	

Table 4 Formalisation of product requirements (continued)

#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	

The second step is to integrate the above diagrams into a formal structure and identify product, environment, and their relations. Figure 8 shows the formal structure of the rivet setting tool example. In the context of product system, a noun names either of product, environment, or the relations between them while a verb names the type of relation involved in a sentence. From this structure shown in Figure 8, we can find out the product to be designed is a tool and all the product requirements of the tool are clearly expressed around the tool.

Figure 8 Formal structure of rivet setting tool design example



4 Software prototype

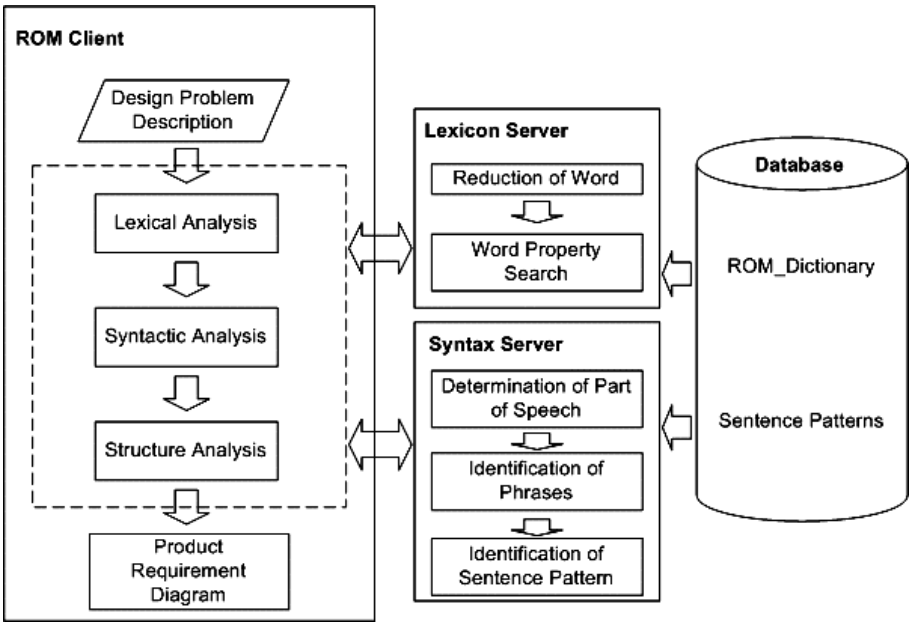
4.1 Problem formulation

As discussed earlier, to specify product requirements is to transform the description of a design problem in natural language into a formal structure that defines the product, environment, and their relations. A software prototype has been developed to translate automatically product requirements described by natural language into a formal structure diagram. The input of this software prototype is a set of sentences describing a design problem and the output is a diagram showing the relations between objects. In this section, we will present the architecture of this software prototype and the experimental results using the example of the rivet setting tool design.

4.2 System architecture

The software prototype is named *ROM* and contains three modules: ROM Client, Lexicon Server, and Syntax Server. The architecture of this system is shown in Figure 9.

Figure 9 System architecture



ROM Client is the user interface of the entire system. It does not participate in the core logical operations, but it calls these operations from servers. The main tasks of the ROM Client are: to display the formal structure diagram which is resulted from 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 various requirement documents, and to build the formal structure diagram individually.

Lexicon Server is responsible for the reduction of words and returning search results from ROM_Dictionary to ROM Client. ROM_Dictionary is the kernel of the Lexicon Server, which includes so far 67 181 base-form words and 15 141 inflected form words. To improve query performance, over 80 000 words are saved in a two-dimensional hash table. Meanwhile, for the purpose of running on multiple platforms, the Lexicon Server is divided into three layers: core layer, interface layer, and application layer as shown in Figure 11. All core operations are placed in the core layer, such as: loading the dictionary data from file or database, searching for a word in the dictionary, reducing the inflected word to base form, *etc.* An example of lexical analysis is shown in Figure 11. For the word, 'rivet', it returns the search results about parts of speech and the infected forms of the word.

Figure 10 Structure of lexicon server

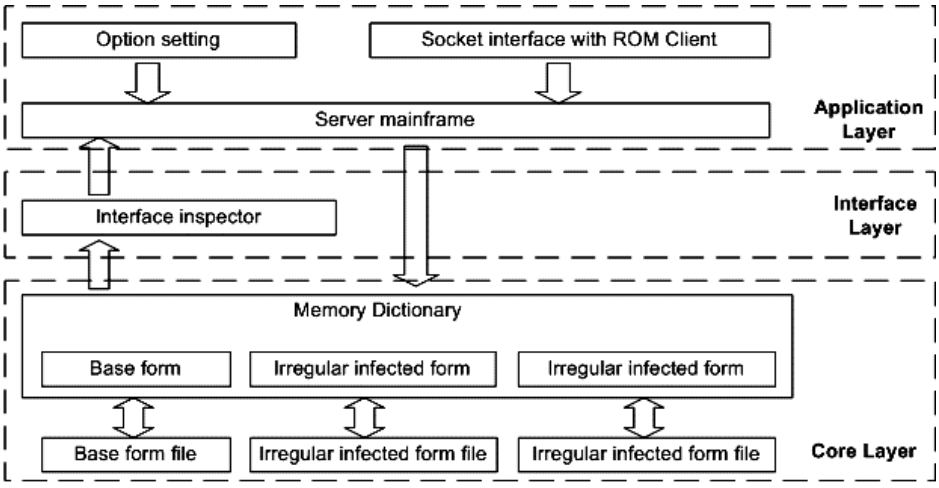
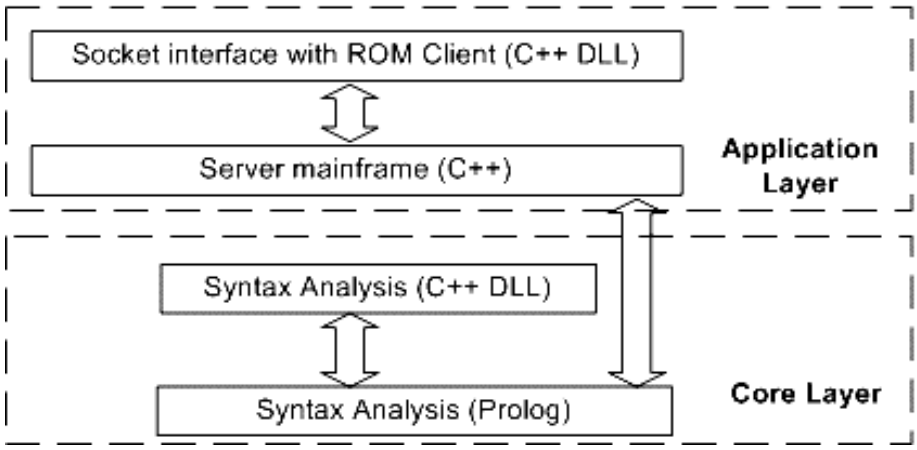


Figure 11 Result of lexicon analysis

```
---Searched from Lex_Server---
rivets
>>>{rivets, noun, rivet, PLURAL[3]}{rivets, transitive verb, rivet, THIRD PERSON SINGULAR[3]}
```

Syntax Server is used to analyse the structure of a sentence. It first maps the sentence structure into the formal structure diagram, converts them into an XML data package, and then sends the package to ROM Client. In this module, we adjust the technology of nested invoking between C++ and Prolog as shown in Figure 12. Since C++ has higher flexibility and performance as well as available supporting libraries for user interface whereas Prolog has powerful capacity in logical deduction, this kind of bi-directional invoking can benefit from each other. The tasks of the Syntax Server consist of three components: determination of part of speech, identification of phrase, and identification of sentence pattern as shown in Figure 12. These three components are implemented with Prolog. The results of the syntax analysis are complex. To organise the data structure efficiently, XML technology is introduced as data exchange format between Syntax Server and ROM Client.

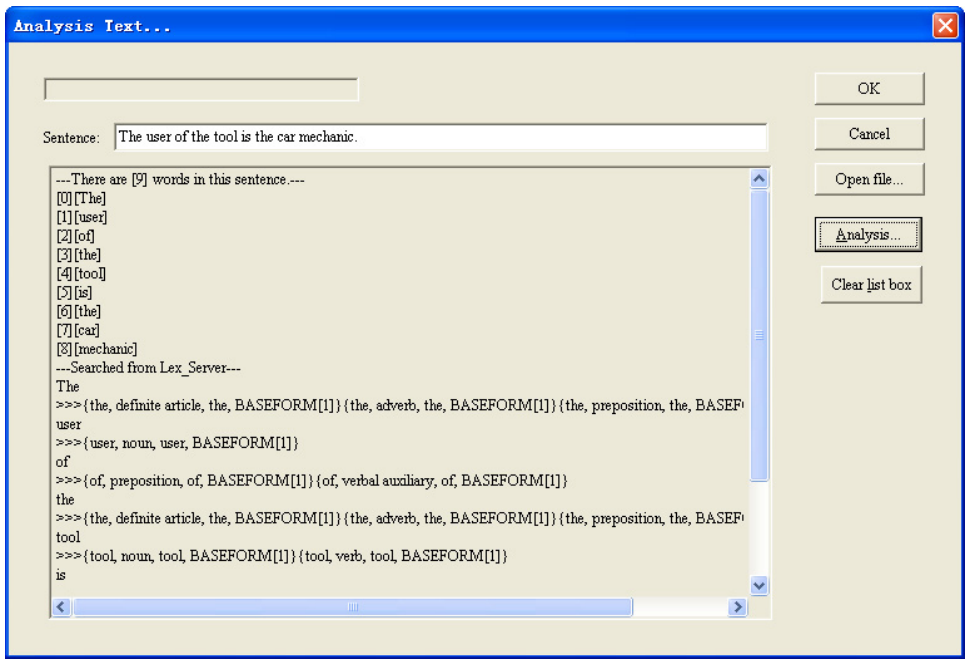
Figure 12 Structure of syntax server



4.3 Experimental results

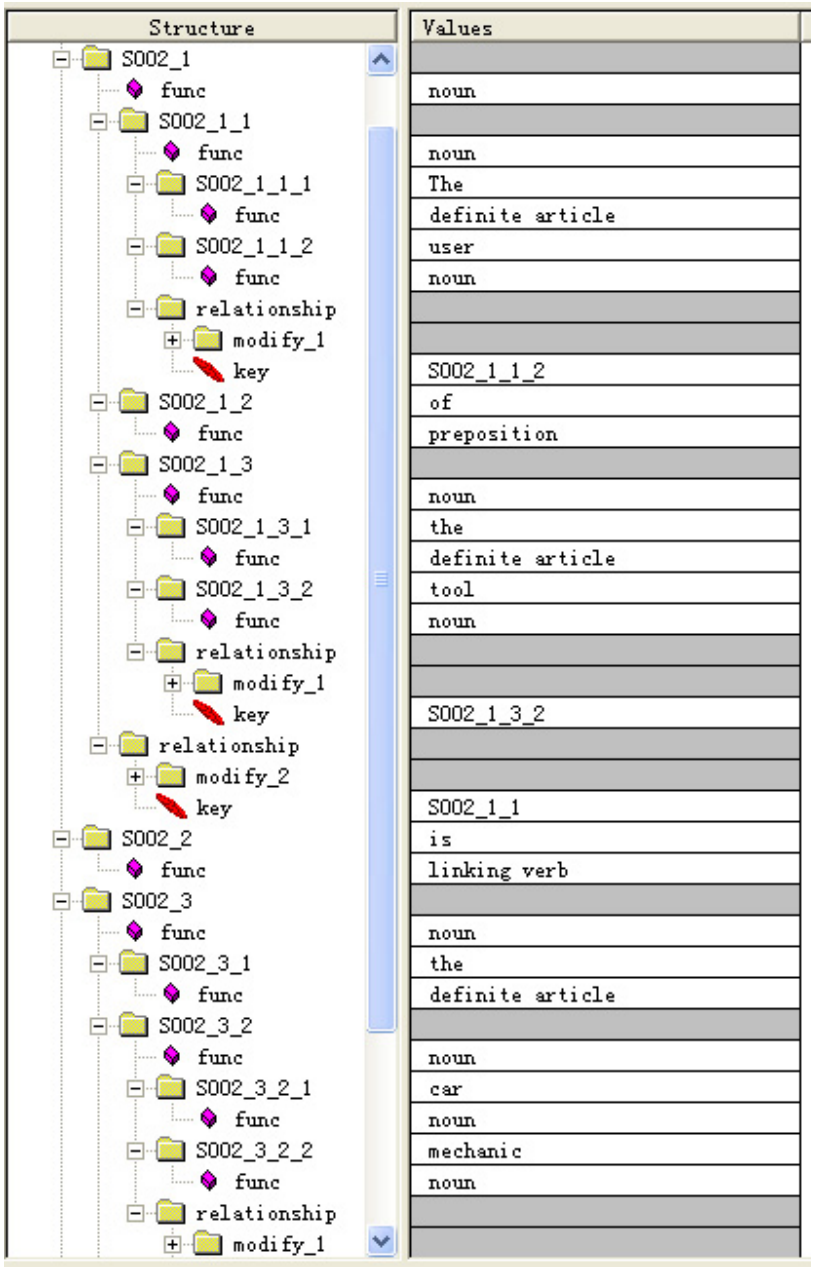
The rivet setting tool design is used to illustrate the feasibility of this software system. Firstly, the description of the design problem is separated into simple sentences. Then each sentence is decomposed into single words, which will be reduced by the Lexicon Server and sent back to the ROM Client. Figure 13 shows the result of decomposing the sentence and reducing it into single words from the Lexicon Server when a single sentence is sent as the input.

Figure 13 Analysis result of single sentence



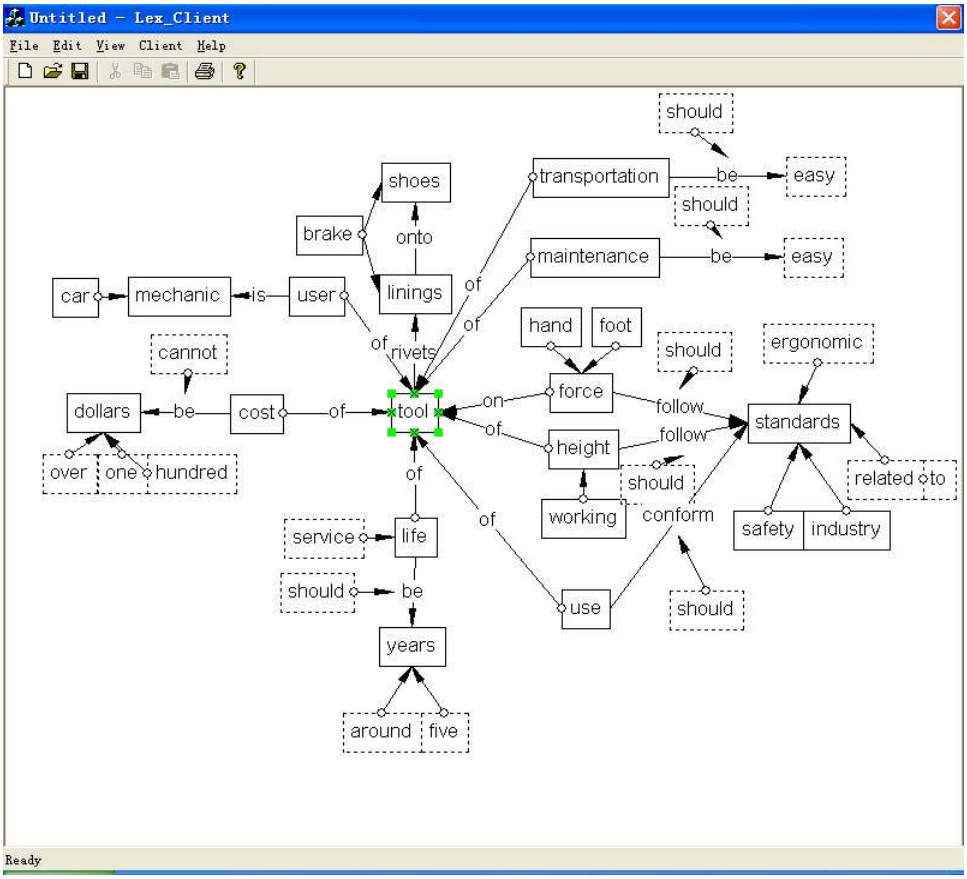
Next, ROM Client sends a data package, which only contains the lexical analysis of one single sentence to the Syntax Server. Then the sentence is analysed in the Syntax Server. The outcome of this step is the structure diagram of the sentence, which is organised into an XML package. The content of the XML data package of an example, 'The user of the tool is the car mechanic', is shown in Figure 14. Then the XML data package is sent to ROM Client.

Figure 14 Result of syntax analysis



Based on the outcome from the syntax server, ROM Client generates the formal structure diagram shown in Figure 15 by analysing all the sentences in the design problem. It can be seen from Figure 15 that the tool is the product to be designed. The 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 will be specified by designers in the rest of the design process.

Figure 15 Formal structure diagram



5 Conclusion

In this paper, a formalisation process is proposed to transform a design problem described in natural language into a formal specification. The foundation of this formalisation process is two theorems of product requirements derived by using the axiomatic theory of design modelling. The first theorem identifies three types of information included in a design problem: environment, structural requirements, and performance requirements.

The second theorem indicates that the source of product requirements is product environment. Through the lexical, syntactic, and structure analysis of natural language based design problem descriptions, the formalisation process identifies the product to be designed, its environment components, and their relations from the generated formal structure diagram. A software prototype is developed to implement the formalisation process. Initial experiment shows that the formalisation process is feasible.

This research can may be used as a core of a computer-aided product design system. Based on the work presented in this paper, a web-based distributed Product Life-cycle Management (PLM) collaborative product design software system is under development for managing the smooth and seamless management of product requirements throughout the new product development in the aerospace industry. An ontology is being developed to define a template for describing product environment and potential relationships between the product and its environment. A new methodology and corresponding software system is being developed for requirements elicitation based on the linguistic analysis method proposed in this paper. We are also integrating a sketch-based conceptual design system into this system to deal with the geometric information appearing in the description of the design problem.

Acknowledgement

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

References

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*, DETC2001/DTM-21692, Pittsburgh, Pennsylvania.

Akao, Y. (1990) *Quality Function Deployment: Integrating Customer Requirements into Product Design*, Productivity Press.

Bjorner, D. and Jones, C.B. (1982) *Formal Specification and Software Development*, Prentice-Hall International Series in Computer Science, Prentice/Hall International.

Braha, D. and Maimon, O. (1998) *A Mathematical Theory of Design*, Kluwer Academic Publishers.

Chen, C.H., Khoo, L.P. and Yan, W. (2002) 'A strategy for acquiring customer requirement patterns using laddering technique and ART2 neural network', *Advanced Engineering Informatics*, Vol. 16, No. 3, pp.229–240.

Chen, C.H., Khoo, L.P. and Yan, W. (2005) 'PDCS-a product definition and customisation system for product concept development', *Expert Systems with Applications*, Vol. 28, No. 3, pp.591–602.

Chen, P. (1983) 'English sentence structures and entity-relationship diagram', *Information Sciences*, Vol. 29, pp.127–149.

Chen, Z.Y. and Zeng, Y. (2006) 'Classification of product requirements based on product environment', *Concurrent Engineering: Research and Applications, an International Journal*, Vol. 14, No. 3, pp.219–230.

Chittaro, L. and Kumar, A.N. (1998) 'Reasoning about function and its applications to engineering', *Artificial Intelligence in Engineering*, Vol. 12, pp.331–336.

Darlington, M.J. and Culley, S.J. (2004) 'A model of factors influencing the design requirement', *Design Studies*, Vol. 25, No. 4, pp.329–350.

Deng, Y.M., Tor, S.B. and Britton, G.A. (2000) 'Abstracting and exploring functional design information for conceptual mechanical product design', *Engineering with Computers*, Vol. 16, pp.36–52.

Fliedl, G., Kop, C., Mayr, H., Weber, G. and Winkler, C. (2004) 'Linguistic analysis of unstructured text by chunk-parsing, tagging, and semantic interpretation', *Proceedings of the 17th International Conference 'Software & Systems Engineering and their Applications' (ICSSEA'04)*, Paris.

Gangopadhyay, A. (2001) 'Conceptual modeling from natural language functional specifications', *Artificial Intelligence in Engineering*, Vol. 15, pp.207–218.

Gershenson, J.K. and Stauffer, L.A. (1999) 'A taxonomy for design requirements from corporate customers', *Research in Engineering Design*, Vol. 11, pp.103–115.

Guttag, J.V. and Horning, J.J. (1993) 'Larch: languages and tools for formal specification', *Texts and Monographs in Computer Science*, Springer.

Harel, D. (1987) 'Statecharts: a visual formalism for complex systems', *Science of Core Programming*, Vol. 8, pp.231–274.

Hirtz, J., Stone, R.B., McAdams, D.A., Szykman, S. and Wood, K.L. (2002) 'A functional basis for engineering design: reconciling and evolving previous efforts', *Research in Engineering Design*, Vol. 13, pp.65–82.

Hoare, C.A.R. (1985) *Communicating Sequential Processes*, Prentice-Hall International Series in Computer Science, Prentice/Hall International, Vol. 256.

Hubka, V., Andreasen, M. and Eder, W. (1988) *Practical Studies in Systematic Design*, Butterworths.

Hubka, V. and Eder, W. (1988) *Theory of Technical Systems*, Springer-Verlag.

Jiao, J. and Tseng, M. (2004) 'Customizability analysis in design for mass customization', *Computer Aided Design*, Vol. 36, No. 8, pp.745–757.

Jiao, J., Tseng, M., Duffy, V.G. and Lin, F. (1998) 'Product family modeling for mass customization', *Computers & Industrial Engineering*, Vol. 35, Nos. 3–4, pp.495–498.

Liu, D., Subramaniam, K., Eberlein, A. and Far, B.H. (2004) 'Natural language requirements analysis and class model generation using UCDA', in R. Orchard, C. Yang and M. Ali (Eds.) *Proceedings of the 17th International Conference on Innovations in Applied Artificial Intelligence, IEA/AIE 2004*, Ottawa, Canada, Berlin Heidelberg: Springer-Verlag, 17–20 May, LNCS 3029, pp.295–304.

Lossack, R.S., Umeda, Y. and Tomiyama, T. (1998) 'Requirement, function and physical principle modeling as the basis for a model of synthesis', *Computer Aided Conceptual Design'98, Proceedings of the 1998 Lancaster International Workshop on Engineering Design*, pp.165–179.

Lynch, N.A. and Tuttle, M.R. (1987) 'Hierarchical correctness proofs for distributed algorithms', MIT Technical Report, pp.137–151.

Manna, Z. and Pnueli, A. (1991) *The Temporal Logic of Reactive and Concurrent Systems: Specification*, Springer.

McKay, A., de Pennington, A. and Baxter, J. (2001) 'Requirements management: a representation scheme for product specifications', *Computer-Aided Design*, Vol. 33, pp.511–520.

Milner, R. (1982) *A Calculus of Communicating Systems*, New York: Springer-Verlag.

Oxman, R. (2004) 'Think-maps: teaching design thinking in design education', *Design Studies*, Vol. 25, pp.63–91.

Pahl, G. and Beitz, W. (1988) *Engineering Design: A Systematic Approach*, Spinger-Verlag.

Rounds, K.S. and Cooper, J.S. (2002) 'Development of product design requirements using taxonomies of environmental issues', *Research in Engineering Design*, Vol. 13, pp.94–108.

Schank, R.C. (1975) *Conceptual Dependency Theory. Conceptual Information Processing*, North-Holland/Elsevier, Amsterdam/Berlin.

Schmidt, H. (1995) *Advanced English Grammar*, Prentice Hall.

Spivey, J.M. (1988) 'Understanding Z: a specification language and its formal semantics', *Cambridge Tracts in Theoretical Computer Science*, New York: Cambridge University Press.

Stacey, M.K., Clarkson, P.J. and Eckert, C.M. (1999) 'Signposting: an AI approach to supporting human decision making in design', Technical Report CUED/C-EDC/TR85, Cambridge University Engineering Department.

Suh, N. (1990) *The Principles of Design*, Oxford University Press.

Turner, R. (1971) *Grammar Review for Technical Writers*, Rinehart Press.

Yoshikawa, H. (1981) 'General design theory and cad system', in T. Sata and E.A. Warman (Eds.) *Man-machine Communication in CAD/CAM, Proceedings IFIP WG 5.2 Working Conference*, North Holland, Amsterdam, pp.35–38.

Zeng, Y. (2002) 'Axiomatic theory of design modeling', *Transaction of SDPS: Journal of Integrated Design and Process Science*, Vol. 6, No. 3, pp.1–28.

Zeng, Y. (2004a) 'Environment-based formulation of design problem', *Transaction of SDPS: Journal of Integrated Design and Process Science*, Vol. 8, No. 4, pp.45–63.

Zeng, Y. (2004b) *Environment-based Design: Process Model*, Concordia Institute for Information Systems Engineering, Concordia University, Montreal.