

Computer-assisted transformation of design documents from a natural
language description to structured modeling languages

Lei Chen

A Thesis

In

The Department

Of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science
(Quality Systems Engineering) at
Concordia University
Montreal, Quebec, Canada

August 2008

© Lei Chen, 2008



Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file *Votre référence*

ISBN: 978-0-494-45333-9

Our file *Notre référence*

ISBN: 978-0-494-45333-9

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

ABSTRACT

Computer-assisted transformation of design documents from a natural language description to structured modeling languages

Lei Chen

In the present thesis, a novel approach is proposed to transform design documents described by a natural language into a structured modeling languages, particularly UML diagrams and FBS models. The transformation consists of two steps:

- i. From natural language to an intermediate graphic language called Recursive Object Model (ROM).
- ii. From a ROM diagram to a modeling language.

The ROM diagram corresponding to a text includes the main semantic information implied in the text by modeling the relations between the words in a text. Based on the semantics implied in the ROM diagram, a set of criteria is proposed to mine the semantic meaning of the original text corresponding to the ROM diagram. Once the semantic meaning of the design documents through their corresponding ROM diagram is captured, a set of mapping rules from the ROM diagram criteria to the modeling language elements is proposed. After that, a set of generation rules to explore the relationship between these elements is proposed to generate UML diagrams and FBS models based on a ROM diagram. A software prototype R2U is presented as a proof of concept for transforming

ROM diagrams to UML diagrams. Another software prototype R2FBS is also presented as a proof of concept for transforming ROM diagrams to FBS models. Several case studies show that the proposed approach is feasible. The proposed approach can be applied to requirements modeling in various engineering fields such as software engineering, automotive engineering, and aerospace engineering. Future work is indicated at the end of the present thesis.

Acknowledgements

I would like to thank my supervisor Dr. Yong Zeng for his support in the present thesis. Dr. Zeng was always there to listen and to give me advice. The present thesis would not have been possible without his encouragement and guidance. The good advice, support and friendship of Dr. Zeng have been invaluable on both an academic and a personal level, for which I am extremely grateful.

Besides my advisors, I would like to thank my thesis examining committee: Dr. Chun Wang and Dr. Olga Ormandjieva, who provided invaluable comments to my thesis. Also, I am grateful to Bruce Peterson, who gave insightful comments and reviewed my work on a very short notice.

My gratitude also goes to all the members in our design lab research group for interesting discussions and being fun to be with me in the past two years.

Above all, I would like to thank my wife Nan for her personal support and great patience at all times. My parents and sister have given me their unequivocal support throughout, as always, for which my mere expression of thanks likewise does not suffice.

Table of Contents

List of Figures	viii
List of Tables.....	x
Chapter 1 Introduction	1
1.1 Motivation.....	1
1.2 Objective	4
1.3 Challenge.....	7
1.4 Approach.....	10
1.5 Literature Review	11
1.5.1 Graphic specification language	11
1.5.2 NLP in requirements engineering.....	12
1.6 Thesis Organization.....	22
Chapter 2 Theoretical Foundations	24
2.1 Axiomatic Theory of Design Modeling	24
2.2 Semantic Analysis of a Requirements Text	25
2.2.1 Recursive Object Model (ROM): introduction.....	26
2.2.2 ROMA: ROM analysis.....	27
2.2.3 Semantics from ROM diagram.....	32
2.3 Formalization of Design Requirements.....	34
2.4 Function-Behavior-State (FBS) Modeling	35
Chapter 3 Framework for Automatic Transformation from ROM Diagram to Modeling Languages	41
3.1 General Framework for the Semantic Analysis of ROM Diagrams.....	41
3.1.1 Semantic analysis of ROM diagrams	41
3.1.2 Knowledge base for the ROM diagram.....	43
3.2 Framework for Automatic Generation from ROM Diagram to UML.....	49
3.3 Framework for the Automatic Formalization from ROM Diagram to FBS Models.....	51
Chapter 4 Transformation from ROM Diagram to UML Diagram - R2U	54
4.1 Generation Rules.....	54
4.2 Software Prototype.....	57
4.3 Case Study.....	59

Chapter 5 Transformation from ROM Diagram to FBS Models - R2FBS	63
5.1 Identify the FBS Schema from ROM Diagram.....	63
5.1.1 Function decomposition rules for ROM diagram.....	63
5.1.2 Mapping function elements to FBS schema.....	69
5.2 Transformation Algorithm	73
5.3 Case Studies	75
5.3.1 Formalization of requirements text	75
5.3.2 Formalization of design patents	79
Chapter 6 Conclusion and Future Work.....	83
6.1 Conclusion.....	83
6.2 Future Work	84

List of Figures

Figure 1 Representations describing design information [7].	5
Figure 2 Analysis process and modeling process [22].	13
Figure 3 Hierarchy and instances of classes of verbs [22].	14
Figure 4 Mapping rules [22].	15
Figure 5 Process of generating natural language texts [22].	16
Figure 6 Requirements elicitation and modeling [21].	18
Figure 7 Architecture of ER-Converter tool [23].	20
Figure 8 Assisted requirements analysis process [20].	21
Figure 9 ROMA process	28
Figure 10 ROMA snapshot	29
Figure 11 Engineering system [34].	34
Figure 12 Formalization processes of design requirements [34].	35
Figure 13 The relationship between function, behavior, and state[3].	37
Figure 14 Architecture of the an FBS modeler [3].	38
Figure 15 Differences between an FBS and an FBPhPhS model [35].	39
Figure 16 Eight levels of requirements[36].	43
Figure 17 Performance scheme[40].	47
Figure 18 Representations for describing design information	50
Figure 19 Framework for the transformation of a requirements document into conceptual models	51
Figure 20 Data flow diagram of an R2U application	57
Figure 21 ROM diagram of the test requirement text	60
Figure 22 Use Case diagram output of R2U 1.0	61
Figure 23 Domain Diagram (Class Diagram) output of R2U 1.0	62
Figure 24 A FBS diagram [3].	64
Figure 25 Function decomposition logic.	69
Figure 26 Data flow diagram of automatic formalization tools	73
Figure 27 ROM diagram of the test requirement text	77
Figure 28 Function schema gain from a ROM diagram by R2FBS 1.0 for requirements.	77
Figure 29 A FBS model output of R2FBS 1.0 for design requirements	78
Figure 30 A ROM diagram for the low temperature clothes dryer	80

Figure 31 Function schema of the low temperature clothes dryer generated by R2FBS 1.0 .81

Figure 32 FBS models of low-temperature clothes dryer82

List of Tables

Table 1 Elements of Recursive Object Model (ROM) [16]	26
Table 2 XRD terminology	29

Chapter 1

Introduction

1.1 Motivation

Designing is a creative endeavor, an aesthetic action taken by a designer to consider the appearance, functionality, performance, and many other aspects of a product or a process. Certain stages of a design process include understanding the requirements, the conceptual modeling, the system design, and the detail design whereas a requirement is the foundation of the design process. The basis of a design process is the complete and correct understanding of the original design requirements. A requirement is a condition that must be met or possessed by a system or a system component to satisfy a contract, a standard, a specification, or other formally imposed documents. A well-formed requirement is a statement of system functionality (a capability) that must be met or possessed by a system to satisfy a customer's need or to achieve a customer's objective, and that is qualified by measurable conditions and bounded by constraints (IEEE Standard 830-1998) [1]. According to this definition, a requirement, to a certain extent, represents the customer's voice about what is needed and what is wanted. Normally a design requirement can be roughly divided into two parts: functional requirement and non-functional requirement. The traditional way in requirements engineering is modeling the functional requirement and state out the non-functional requirement. This also means the separation of the design activities into functional design and detail design. Functional design, also known as conceptual design, is the key to the whole design process. Thus

how to clearly present the functions described in a design requirement is the crucial problem in the design process. Also, the designer needs to find out the relationship between those functions, the structure of the conceptual design, the state from one function to another function, and the trigger in each state changing.

To ensure the success of a design process, it is crucial to identify the accurate requirements for the whole design, especially in the requirement specification stage.

However, difficulties exist in obtaining the accurate requirements specification:

- 1) As the requirements are gathered from the customer, the customer's need for a product or procedure can be ascertained. For various reasons, customers may not be able to describe their needs accurately [2].
- 2) A requirement is normally described in natural language, which is unrestricted from computer representation point of view. In contrast, computer-aided design is usually supported by a structured process that needs a formal specification of requirements. It is not realistic to ask a designer to analyze the requirements and to structure the information obtained from a natural language and then to generate the formal specifications of a requirements text.
- 3) Product requirements are the backbone necessary for the integration of enterprise applications and the management of the product lifecycle. In many engineering projects, the documents specifying the requirements are often very long and are recorded in a text format. It is quite challenging for various partners to follow

and/or to maintain the document. In addition, as advanced enterprise applications become commercially available, the transformation of legacy design documents into those systems creates a bottleneck for any enterprise that wants to take advantage of such advanced systems.

- 4) Nowadays, as systems are becoming larger than ever before, the requirements documents are becoming more and more complex. In the mean time, only a design team can fulfill the design task. However, even in a well-formed design team, because of the nature of human beings, no two designers have the same understanding of even one segment of the requirements documents, thereby often causing variations in the final design documents.

To provide accurate specifications for design, formal and structured languages, such as FBS Models [3], UML [4], SysML [5], have been developed. However, these formal tools are often too rigid to capture the customer's intention, especially in the early stages of product development. Moreover, even though all the structured languages just mentioned provide well-formed modeling tools for the designer, they provide no assistance in the analysis of the requirement documents. Then, unfortunately, the different backgrounds of engineers, their different focuses on requirement documents, and their different understanding of any fuzzy segment of the requirement documents will necessarily and invariably lead to a failed project.

Therefore, methodologies should be created and used to automatically generate a formal modeling language from the requirements described in natural language.

1.2 Objective

In engineering design, just as in all other design problems, a precise and complete description of design requirements is crucial for the successful and efficient completion of a design task [6]. In describing the product requirements, various representations may be involved, such as verbal statements, graphic models, and mathematical expressions. This variety of design representations can be illustrated in Figure 1 [7]. Smith and Browne have classified design representations into natural language, mathematical models, diagrams of physical objects and processes, and three-dimensional models [8].

Geometric models define the shape of an object: a physical object or visual object. This object can be a 2D geometric model or a 3D geometric model, which is mainly used in manufacturing. Like geometric models, sketches can also describe the shape of an object though not in as much detail as that given by geometric models. This is because geometric models define shapes by using algorithms. Unlike the previous two types of representation, graphic language is not a representation of shape but is a symbolic expression of a design text such as UML and concept maps. If the graphic language is a symbolic expression of design, then mathematical language represents the design in terms of formulas. Mathematical language is also the most precise expression of any engineering problem or solution. Compared with all other representation methods in the

designing process, natural language is the most ambiguous and unrestricted expression method. However, since any design process begins with the customer needs, customers can express their needs only through natural language and the communication tools between designer and customer could only be through natural language. Natural language expression is the unique method during the early design stage. Moreover, since all the detail design is determined from the early stages of the design process, the success of the whole project relies on the understanding of the customer's voices, which are of course expressed in natural language. Consequently, understanding natural language is critical for innovative and creative design.

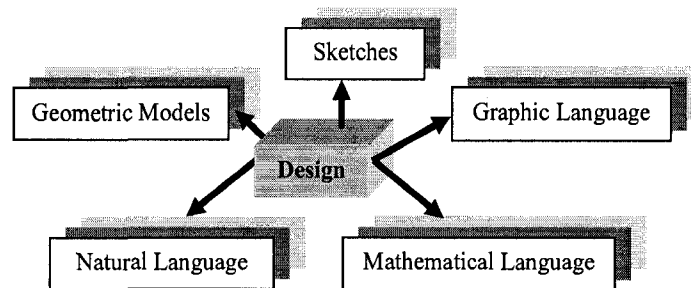


Figure 1 Representations describing design information [7].

Among all other representations, graphic models are the most effective and the most efficient; mathematical language is however the most precise. Whereas the best structured representation is mathematical language, engineers prefer graphic models, especially the standardized models.

Because of the advantage of graphic language, many graphic languages have been proposed to represent the requirements. However, as mentioned above, the most common

representation of requirements is in natural language. To deal with the bias in the understanding of natural language requirements, an intermediate graphic language should be proposed to unify the semantic meaning of natural language and to extract standard information from it. Furthermore, this graphic language should also act as a bridge between natural language and structured modeling languages such as FBS models and UML. The approach proposed in the present thesis is based on such an intermediate representation: Recursive Object Model (ROM). ROM can represent all the linguistic elements in natural language whereas it is derived from a mathematical theory [9].

Accordingly, the objectives of the present thesis are as follows:

- 1) Extract the semantic meaning of natural language requirements based on linguistic analysis and generate the output: the ROM diagram.
- 2) Find the generation rules that are required to generate graphical representations of the requirements such as UML and FBS models from the semantic meaning.
- 3) Derive a methodology combining natural language processing and conceptual modeling.

A systematic graphic language that represents a requirement is very helpful in design process. The present thesis does not intend to propose a new modeling language for requirements. Instead, it uses existent modeling language and tries to find a way to automatically transform natural language into graphic language. The essential step of this approach is to derive or obtain the semantic meaning of a text from the ROM diagram. In

order to generate UML diagrams or FBS models from natural language, generation rules will be proposed in the present thesis to meet such an objective.

As discussed above, the problem of systematic specification language has been well researched and many theories or models have been proposed. To achieve our objective in the present research and to simplify our task, we have selected two typical graphic specification languages – UML diagrams and FBS models - for the final output of our research.

UML has been selected mainly because UML is one of the most popular requirement specification models, especially in software engineering fields. UML has been used as an important tool in the requirements engineering fields. It can identify the use cases of a requirement and can form a class diagram to assist with further detailed design. FBS modeling has been selected as the other formal structural language mainly because it is widely accepted in the field of mechanical design as an important conceptual design methodology to assist product design. Also, FBS is one kind of function modeling language that analyzes the product requirements and forms FBS models. Our specific objective in this thesis is to propose a methodology that transforms natural language into UML and FBS models.

1.3 Challenge

The modeling of requirements is a process of formalizing the ambiguous natural language description of the customer's needs into a more precise structured representation. With

the assistance of such structured representation, a system designer can better model the system requirements and the architecture.

Consider a complex engineering project such as the design of a product or a manufacturing procedure. There should be a series of functions to complete the project, and each function will be affected by the environment and former actions. The process of formulating the conceptual model is a human activity in analysis and determination. However, misunderstanding the customer's real needs is a major issue that may lead to incorrect structural models. Furthermore, as the product requirements become more and more complex, more and more industrial fields may be involved. In addition, the limited background of a designer can focus only on one specific industry. All of the above may limit the design activities and may eventually cause the whole project to fail.

To generate representation models automatically from product design documents described in natural language, the following problems have to be solved:

- How to capture the meaning of a text automatically.
- How to define the representation scheme of a product function described in a specific modeling language and by specific modeling mapping with a ROM element that captures the semantic meaning of the design text.
- How to derive the right conceptual models by simulating human analytic logic to decompose a design text.

Obviously, the first problem must be solved through Natural Language Processing (NLP) algorithms. The tools based on NLP are able to improve the quality of communications throughout the design process, to facilitate the understanding of the customer's real intention, and to elicit precise and complete product requirements [7]. Once the meaning of the requirement text is precisely captured, structural models can be generated automatically.

For the second problem, current engineering practice is to generate the modeling language of requirement documents from the original customer requirements manually through communicating iteratively with the customer. This is often a recursive process: gathering and formulating customer requirements, generating preliminary solutions, and refining customer requirements [2, 10]. The final requirement specification comes from such a brainstorming process. However, as business becomes more and more complex, multiple customers, with different backgrounds, are usually involved in the requirement modeling process. Misunderstanding the customers' real needs is a major issue that may lead to incorrect requirements specifications. There exists a contradiction between product requirements description based on ambiguous natural language and the precisely structured language used to model the product requirements.

Furthermore, for complex engineering projects, the design document includes a great amount of information, the human processing of which is extremely tedious. Efforts have been made to develop automatic or semi-automatic processes that bridge these two extremes: an unrestricted natural language text and a structured formal representation [11,

12]. Still, due to the difficulties in the processing of unrestricted natural language, the success of these efforts is limited [13-15].

1.4 Approach

To overcome the two challenges proposed in Section 1.3, a series of steps is proposed. First, to bridge the gap between unrestricted natural language and formal modeling language, an intermediate representation is useful. The approach proposed in this present thesis is based on such an intermediate representation: Recursive Object Model (ROM) [16]. ROM can represent all linguistic elements in natural language [16] whereas it is derived from a mathematical theory [9]. The semantics of a text can be derived from the ROM diagram. The proposed approach first generates the ROM diagram of a text describing the product requirements, from which use case diagrams and class diagrams are extracted.

Then, generation rules should be derived to extract the information from natural language requirements and to map the information to existing modeling languages such as UML and FBS.

The proposed approach in the present thesis is characterized in the following list:

- 1) Generates the ROM diagram for the product requirements in natural language.
- 2) Generates the key element of the requirements that fulfills the product objective.

- 3) Extracts the semantic meaning from the ROM diagram and maps the key element into certain modeling language elements to automatically generate a graphical representation of the requirements such as UML and FBS models.

1.5 Literature Review

The objective of the present thesis is to automatically generate a structural graphic representation such as UML diagrams and FBS models from natural language. Furthermore, the present thesis basically uses Natural Language Processing techniques to transform textual requirements into existent requirement models. To achieve this research goal, the literature review includes the following fields:

- Graphic specification language for conceptual design
- Natural language process in requirement engineering

1.5.1 Graphic specification language

Many researchers have attempted to develop algorithms for understanding the semantics of a natural language text and for translating the text into some types of graphic language. Zeng and Mehdi have developed a software prototype called 3DSV to generate a VRNL graphic representation from a simplified story-based description of a scene [17]. Ma and McKeivitt have attempted to automatically generate a semantic representation of events in 3D animation by using a semantic representation as a bridge between linguistic inputs and visual knowledge [18]. Jesen *et al* have developed an approach to automatically generate the UML diagrams from XML DTDs [19]. However, due to the lack of semantic

analysis, none of these approaches can process unrestricted natural language texts. Based on the semantic representation of a text through the ROM diagram, our proposed approach can generate UML diagrams and FBS models from the design text described in natural language.

1.5.2 NLP in requirements engineering

Our goal to automatically transform a design text described in natural language into a specific modeling language is also an Artificial Intelligence (AI) approach. Natural language processing is the foundation for this approach. Since Natural Language Processing is a complex research field that may touch on many categories, we focus mainly on Natural Language Processing in the area of requirements engineering fields.

There are several researchers who have attempted to use a linguistic approach to support requirements engineering, especially in the modeling of the requirements specification. MacDonell *et al.* have proposed an approach – autonomous requirements specification processing, using natural language processing [20]. Mich has proposed CASE (Computer Aided Software Engineering) tools, which are called NL-OOPS, using a Natural Language Processing System LOLITA to support the transformation of natural language to object-oriented models [21]. Rolland and Proix also think that natural language plays an important role in the conceptual specification stage in the development of computerized systems. They have proposed a CASE tool based on linguistic approach to support requirements engineering [22]. Moreover, in database design fields, Omar *et al.*

have proposed a new heuristics that assists the semi-automated generation of Entity-Relation (ER) diagrams for database modeling [23]. All these approaches just mentioned are similar to our automatic generation from a natural language design text to a specification modeling approach.

Rolland's approach focuses mainly on requirements engineering in database and information system development and provides a CASE tool called OICSI (a French acronym for "intelligent tool for information system design"). OICSI is a system prototype that exploits knowledge-based paradigms to provide an active aid to database and information system analysts during the Requirements Engineering process. Problem-statements in OICSI are expressed in French natural language and are automatically interpreted in terms of the OICSI conceptual model. Similarly, OICSI uses a text generation technique to give feed back to the user on information about the specification (i.e. the conceptual schema). Figure 2 illustrates the analysis process and modeling process.

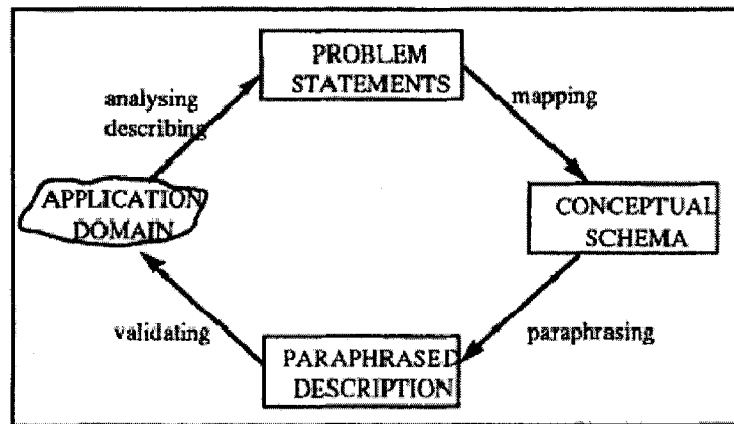


Figure 2 Analysis process and modeling process [22].

The OICSI process has two main parts – conceptual modeling and conceptual schema validation. Conceptual modeling in OICSI is based on a linguistic approach that tries to formalize the linguistic mechanisms through which analysts are able to abstract observed phenomena into concepts.

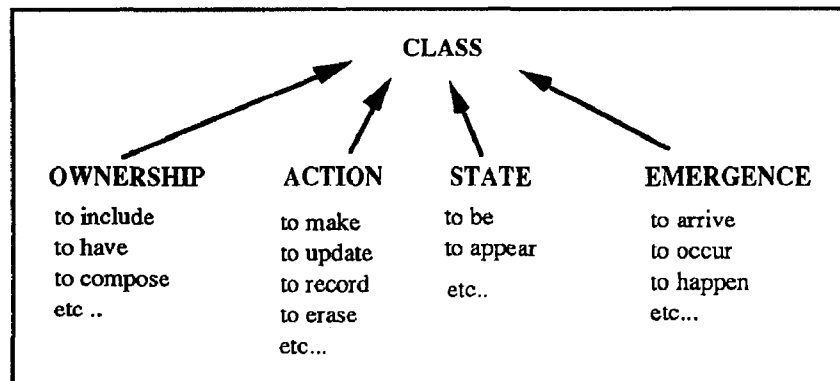


Figure 3 Hierarchy and instances of classes of verbs [22]

The approach begins with Fillmore’s case system and considers the cases to be types of relationships that groups of words have with the verb in any clause of a sentence. By classifying the case as a class and hierarchy of words and patterns such as sentence patterns and elementary patterns, a conceptual schema of a certain requirement can be generated. Figure 3 shows one case in natural language for a class of verbs. The conceptual schema generation process in OICSI is based on rules that map cases into concepts. These rules are dependent on the target conceptual model. Conversely, the linguistic patterns are independent of a particular modeling technique and can be used within any design methodology. Figure 4 shows the mapping rules used in the schema generation process.

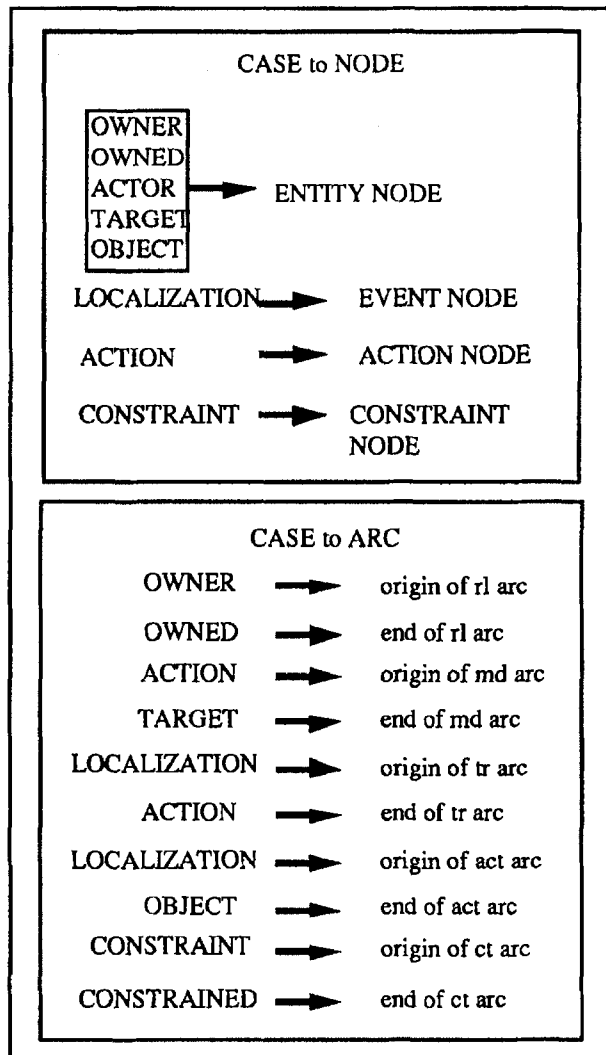


Figure 4 Mapping rules [22]

The schema validation process in OICSI is basically a reverse approach when compared with the schema generation process. The schema validation process converts the specification schema back to natural language sentences to verify the accuracy of a specification schema. Figure 5 shows the detailed process of generating sentences.

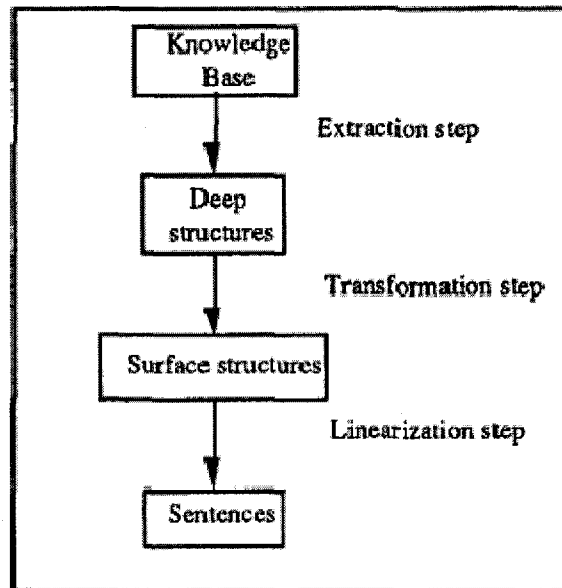


Figure 5 Process of generating natural language texts [22]

Rolland's process is very similar to our approach. It gives us the ideas that Requirements Engineering should be supported by a case tool based on a linguistic approach and that validation of specifications must be performed by means of a text generation technique. These ideas have already been presented.

Mich [21] also proposed a CASE tool that supports requirements analysis by generating object oriented models from natural language requirements documents, in a procedure called NL-OOPS. NL-OOPS is an acronym for Natural Language – Object Oriented Production System that supports natural language requirements analysis by extracting the objects and their associations for use in creating object models. It uses the natural language processing system LOLITA (Large-scale Object-based Linguistic Interactor

Translator Analyser) as an NLP tool, which has been developed at Durham University [24]. Mich thinks requirements analysis includes two main activities – the identification of requirements and the modeling of requirements. The LOLITA assists the requirement identification process by simulating the requirements eliciting process and by performing linguistic analysis such as correcting the requirements, completing the requirements text and eliminating the style difference. Figure 6 shows the NL_OOPS requirements elicitation and modeling process. After pre-processing the requirements, the NL-OOPS system models the pre-processed requirements into object-oriented models through an OO analysis, which contains the following logic:

- finding the objects
- organising the objects
- describing how the objects interact
- defining the operations of the objects
- defining the objects internally

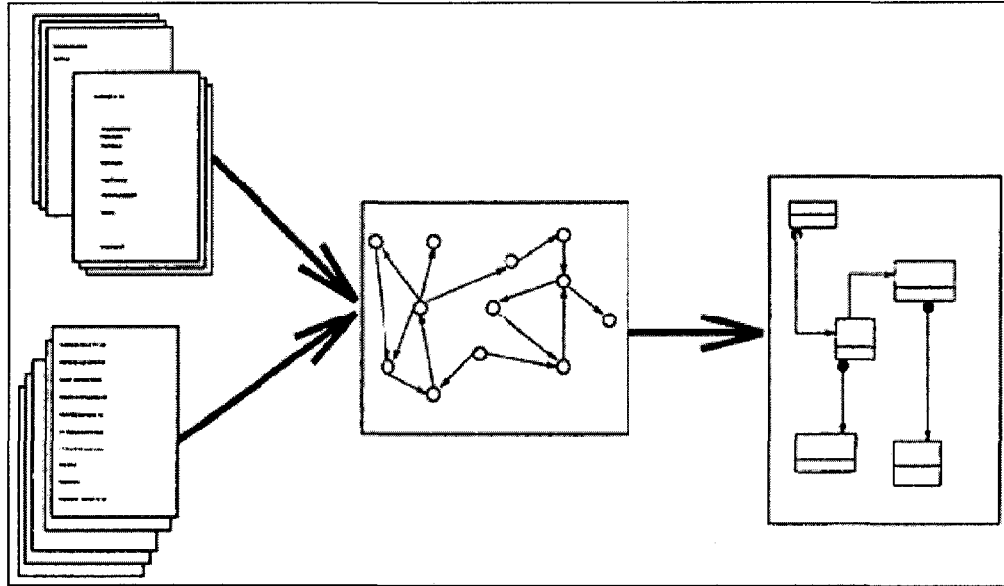


Figure 6 Requirements elicitation and modeling [21]

The NL-OOPS approach proposes the brilliant idea of pre-processing the requirements before the modeling process starts, thereby ensuring the grammar correctness and requirement complement. In our approach, we use the ROM diagram as the intermediate step to transform natural language into modeling language. The ROM diagram is also a Natural Language Processing output of design text that can also ensure the correctness of the natural language input. Moreover, our approach uses a question-asking strategy proposed by Zeng and Wang to ensure the complement of the design text [25].

As in the case of the previous two approaches, Omar *et al.* propose a heuristics-based ER modeling process that tries to automatically formalize the design documents to specification models. The heuristics-based ER modeling process provides a semi-automatic transformation process tool called an ER-Converter. This approach mainly

focuses on database modeling using an ER diagram theory as the implementation model. Also, this approach uses a heuristics method to identify the ER elements. Figure 7 shows the architecture of the ER-Converter tool. The heuristic method can provide a good solution but not necessarily an optimal solution for the identification of the ER elements. These methods can quickly extract the ER elements from the design requirements by using certain selection rules. The following is one part of the heuristic methods for the selection rules:

Heuristics to determine entities:

1. Heuristic HE2: A common noun may indicate an entity type.
2. Heuristic HE3: A proper noun may indicate an entity.
3. Heuristic HE7: If consecutive nouns are present, check the last noun. If it is not one of the words in set S where $S = \{\text{number, no, code, date, type, volume, birth, id, address, name}\}$, most likely it is an entity type. Otherwise, it may indicate an attribute type.

Heuristics to exclude non-potential entity type candidates:

1. Heuristic HEX: A noun such as “record”, “database”, “company”, “system”, “information” and “organization” may not be a suitable candidate for an entity type. For example, “company” may indicate the business environment and should not be included as part of the entity types. Examples:
 - a) “An insurance company wishes to create a database to keep track of its operations.”
 - b) “An organization purchases items from a number of suppliers.”

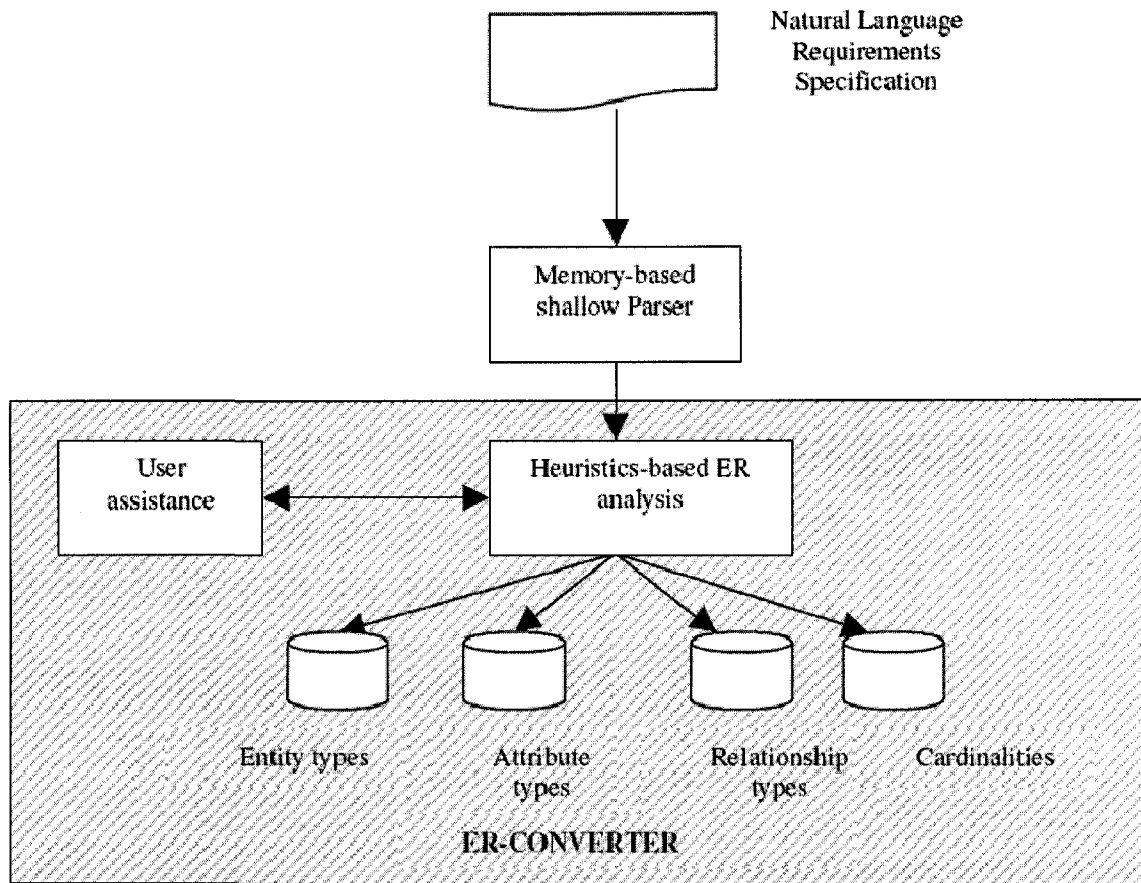


Figure 7 Architecture of ER-Converter tool [23]

Unlike the three previous approaches in transforming natural language into specification models, the autonomous requirements specification process proposed by MacDonell *et al.* tries to find a way to automatically generate the system design specification by using natural language processing, which is also the same goal for our approach. This autonomous process contains a natural language parsing system and a term management system. The parsing system starts after a set of token is extracted from the requirement specification documents, an approach which is similar to our own in that it defines a set

of criteria for extraction from a design text. Figure 8 shows the architecture of this approach.

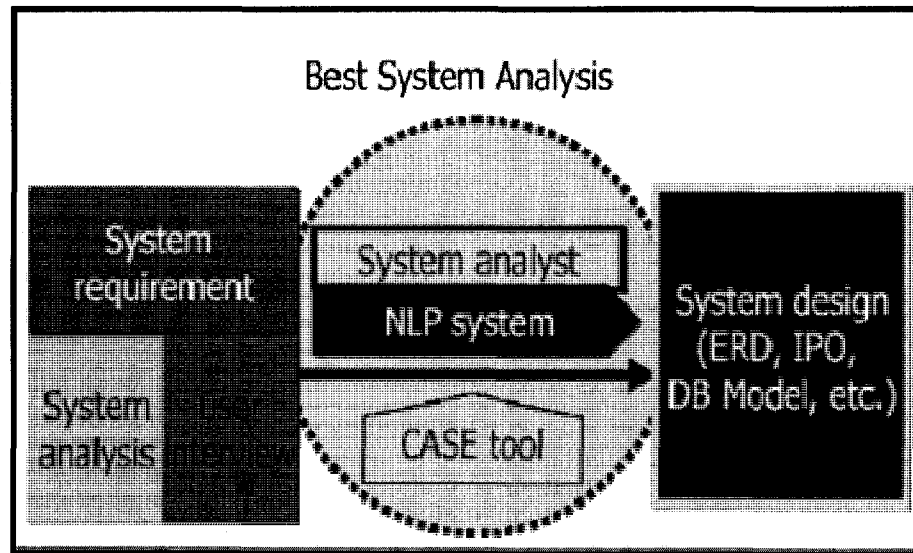


Figure 8 Assisted requirements analysis process [20]

The difference between our approach and the other processes previously mentioned is that all of the four other approaches extract specification elements directly from natural language documents after defining a set of criteria. Considering the complexity and unrestricted of natural language, the extracting process from natural language is difficult to secure the complement generation from natural language. By contrast, our approach uses an ROM diagram as an intermediate between natural language and modeling language. Moreover, all the approaches referred to above rely heavily on the grammar parsing process and they lack semantic meaning analysis which often leads to lack of utility in real-time implementations.

1.6 Thesis Organization

Chapter 1

This chapter introduces the scope and object of the present thesis and compares our research with several similar approaches to the generation of the design specification models from natural language using Natural Language Processing.

Chapter 2

This chapter discusses the theoretical foundations of the present thesis: the axiomatic theory, ROM theory, formalization methods, and FBS modeling theory.

Chapter 3

This chapter provides the general framework for semantic analysis based on the ROM diagram generated from ROMA software. Then, two frameworks that will be introduced in this thesis have been proposed for two kinds of specification models – UML diagrams and FBS models.

To validate the theory, a software prototype system is presented in Chapter 4 and Chapter 5. An example of a traditional POS management system in the software engineering fields is chosen as a case study to illustrate the theory for the transformation from natural language requirements to UML diagrams. Another two examples of design documents are selected to show our research on the formalization of FBS models based on FBS modeling theory and ROM diagram analysis.

Finally, Chapter 6, Conclusion and Future Work, summarizes the main research results of the present thesis and points out directions for future research.

Chapter 2

Theoretical Foundations

2.1 Axiomatic Theory of Design Modeling

Zeng has proposed an axiomatic theory as the logic tool to represent and to reason about object structures [9]. This is also the basic theoretical foundation for the present thesis. This axiomatic theory gives the designer a logical approach to human thought after defining axioms dealing with objects. The basic concept rests on two definitions of axioms:

- 1) Everything in the universe is an object
- 2) There are relations between objects

Based on these axioms, a requirement can itself be seen as an object, which is defined as O . This being the case, then the structure of the requirement object should be $\oplus O$. The following formula shows Axioms 1:

$$\oplus O = O \cup (O \otimes O) \quad (1)$$

where $O \otimes O$ shows the relation between objects.

Since a requirement can be seen as an object, the elements of natural language requirements – paragraphs, phrases, and words – can also be seen as objects. Take the basic elements of a requirements text – words – as the objects. The requirements can be

decomposed into a set of word objects and into another set of relations within these objects. This observation has led to the Recursive Object Model theory by Zeng [16].

2.2 Semantic Analysis of a Requirements Text

It is widely accepted that graphics are the best means for carrying the semantic message. Several graphic languages are used to support the representation of human thoughts, including formalized concept maps, entity-relationship diagrams, conceptual graphs, topic maps and system modeling languages (OMG SysML) [26-30]. Though they have been instrumental in either modeling systems or in supporting brainstorming, they suffer from a major problem when they are used for processing natural language: the final diagram depends heavily on the person who draws the diagram. That is to say, the semantics underlying the text is an issue of individual experience and knowledge. This fact makes it difficult to apply these diagrams to finding out the precise meaning of a given text.




Based on the axiomatic theory of design modeling [9], Zeng has proposed a new graphic language called the Recursive Object Model (ROM) [16]. Corresponding to each text, there is only one correct ROM diagram, from which other diagrams, such as concept maps and topic maps, can be derived. The semantics of a text can be obtained by applying mathematical algorithms to the ROM diagram.



In the present section, we will first give a brief introduction to the Recursive Object Model (ROM) and its computer representation. Then the basic idea for semantics extraction will be presented.

2.2.1 Recursive Object Model (ROM): introduction

The Recursive Object Model (ROM) [16] is a part of a general design theory: Environment-Based Design (EBD) [6, 9, 10, 31, 32]. In the context of this research, the ROM provides an intermediate medium between natural language and structured modeling language. The ROM theory treats each word in a sentence as an object and considers that every object may have one or more relations to other objects. Furthermore, each sentence also forms an object and has a relation to other sentences in the text. Table 1 shows the elements of ROM.

Table 1 Elements of Recursive Object Model (ROM) [16]

Type		Graphic Representation	Definition
Object	Object		Everything in the universe is an object.
	Compound Object		It is an object that includes at least two other objects in it.
Relations	Constraint		It is a descriptive, limiting, or particularizing relation of one object to

			another.
	Connection		It is to connect two objects that do not constrain each other.
	Predicate		It describes the act of an object on another or that describes the states of an object.

2.2.2 ROMA: ROM analysis

ROMA, the abbreviation of Recursive Object Model Analysis, transforms a text described in natural language into an ROM diagram, which is represented in the XRD format, an extension of XML for ROM diagram. An XML format is mainly focused on data integration at the logical level of the data model, creating a need for techniques that work at the conceptual level, which is more suitable for use by system designers and end users [19]. XRD carries semantic information implied in the text. Figure 9 shows the ROMA process.

For example, we have the following requirement scenario:

Design a tool for riveting brake linings onto brake shoes for internal drum brakes. The user of this tool is a car mechanic. The working height of the user 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. It will be manufactured in a specific workshop, which has specified equipments. The cost of this tool cannot be over \$190.00.

This scenario describes a typical mechanical design problem. Figure 10 shows the ROMA snapshot when analyzing the scenario above.

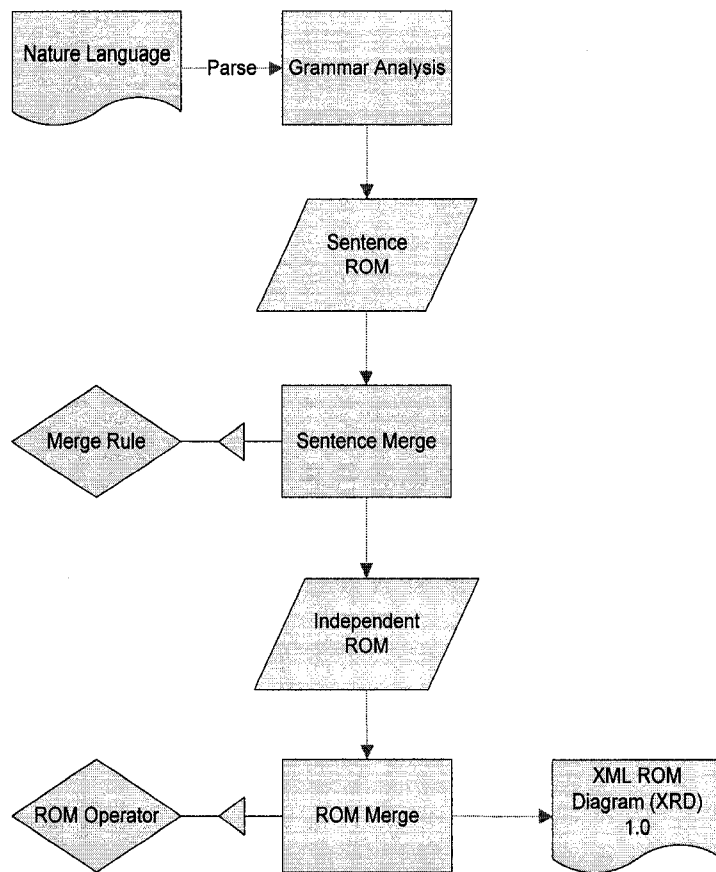


Figure 9 ROMA process

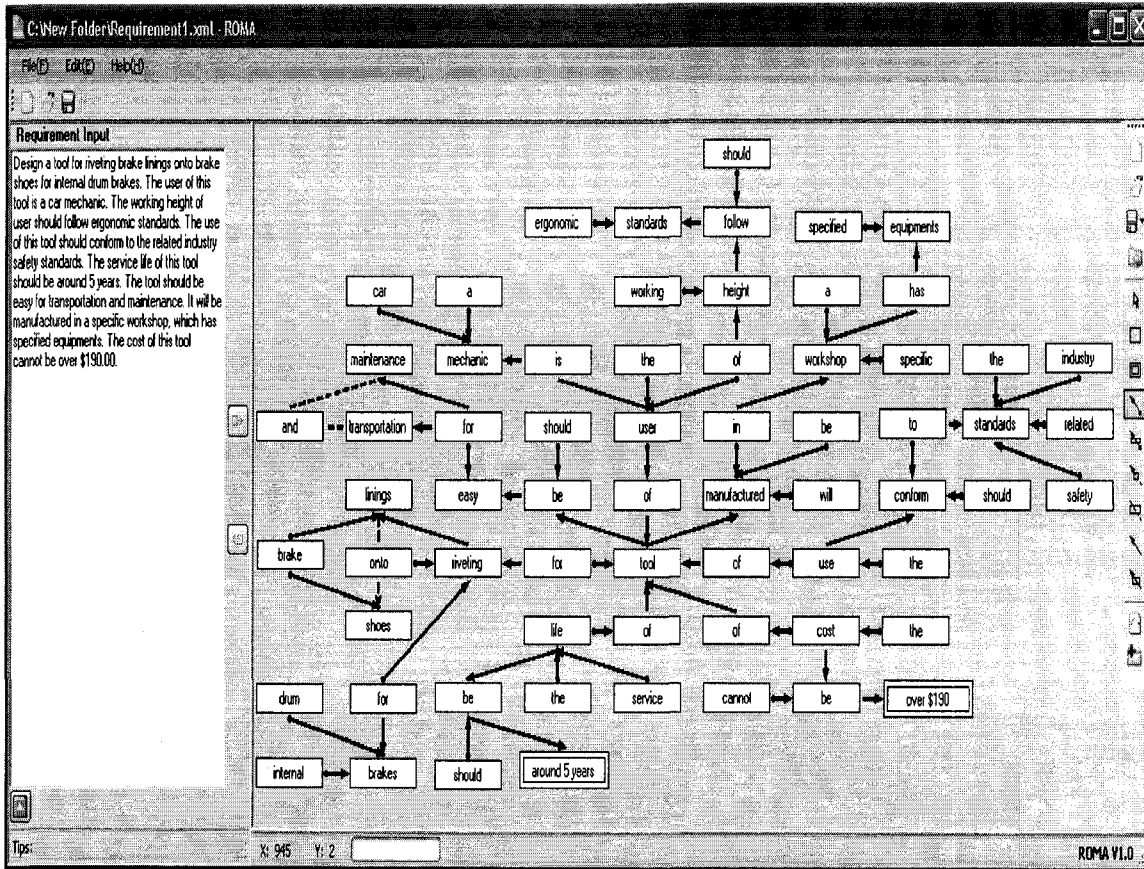


Figure 10 ROMA snapshot

Table 2 is the terminology of XRD

Table 2 XRD terminology

Term in XRD	Explanation	Symbol
The root of XRD	It's the root of the whole xml, which can be recognized with certain traversal method	<rom_root>
ROM	One XRD can have several isolated ROM sub-diagrams	<rom>
Name of ROM	The unique name of each ROM sub-diagram, normally named as "rom" plus number	<rom name="rom1">
Version of ROM	As the ROM sub-diagram may be changed through certain layout or merging algorithms, the version identifies the current status of each ROM sub-diagram	<rom version="0">
Position of ROM	The position of the ROM sub-diagram is the result of layout algorithms of ROMA, which store the position information corresponding to	<rom top left width height>

	its position when in display	
Object	Object is the basic element of the ROM diagram; it can be a simple word, a phrase, or even a sentence regarding the type of object. The object can also be seen as the basic element of text because it stores the words of the text.	<object> <object name romname>
Type of object	Defines the type of object. The single object is the word of a sentence while a compound object can be a phrase or a sentence of a text	<type>
Text carried by the object	The word element of a text	<text>
Position of the object	This position is stored relative to the layout information within a ROM sub-diagram, which is not affected when the sub-diagram position is changed.	<x> <y>
Neighbor information	A neighbor stores all objects connected to the current object. It not only stores the name of the connected object, but it also stores the type of connection and the direction of that connection. This element is very convenient when mining the semantic meaning of an object through relations.	<neighbors> <neighbor neighborname relationtype>
Words class	This element stores the real word class of each single object. It can be “n” for noun, “v” for verb etc. This element can help identify the role of an object in a text	<class>
Relation	Relation is the element that stores the relationship of an object within a ROM diagram. The relation can also be seen as a connection of each element for a text. It is more useful in the traversal method of the ROM diagram because it mainly carries on the semantic meaning of a text.	<relation> <relation name romname>
Type of relation	There are three types of relations corresponding to the definition of ROM in Table 1	<type>
Position of relation	This element stores the information of the relation from which object and to which object	<fobject> <tobject>

The following part is a segment of XRD from the above application.


```

<?xml version="1.0" ?>
<!DOCTYPE XRD1.0 (View Source for full doctype...)>
- <rom_root>
- <rom name="rom0" version="0" left="0" top="0" width="725" height="225">
- <object name="object2" romname="rom0">
  <type>0</type>
  - <position>
    <x>-125</x>
    <y>0</y>
    <width>100</width>
  </position>
  <text>machine</text>
  - <neighbors>
    <neighbor neighborname="Object200" relationtype="6" />
    <neighbor neighborname="object3" relationtype="1" />
    <neighbor neighborname="object4" relationtype="1" />
  </neighbors>
  <class>n</class>
</object>
...
- <relation name="relation1" romname="rom0">
  <type>6</type>
  - <position>
    <fobject>Object200</fobject>
    <fposition>1</fposition>
    <tobject>object2</tobject>
    <tposition>3</tposition>
  </position>
  <status>0</status>
</relation>
...

```

This shows the basic relation between word objects, which can be represented as a mathematical structure for the automatic derivation of semantics. The XRD file can be used by various applications that need the semantic information of a text.

2.2.3 Semantics from ROM diagram

To get the semantics underlying a text through its ROM diagram, the graph theory has been used to process the diagram, where each object in an ROM diagram is viewed as a vertex and each relation is viewed as an edge. An ROM diagram is a directed graph.

Therefore, the ROM diagram has a mathematical structure defined as follows:

Object in ROM diagram as Vertex v

Relation in ROM diagram as Edge e

Then ROM diagram can be defined as $\text{Rom} = \langle v, e \rangle$

The size of the ROM diagram can be represented by a matrix M

$$M = \begin{pmatrix} v_{0,0} & \cdots & v_{0,i} \\ \vdots & \ddots & \vdots \\ v_{i,0} & \cdots & v_{ii} \end{pmatrix}$$

where

matrix M is $i \times i$ matrix;

i is the max number of objects;

is relation to the following value $v_i = \begin{cases} 0 & \text{if no relation between objects} \\ 1 & \text{if constrain relation} \\ 2 & \text{if connection relation} \\ 3 & \text{if predicate relation} \end{cases}$

According to the definition of the matrix representation of ROM diagrams, the matrix of a ROM diagram is a symmetric matrix when the direction of relation is not specified.

However, the elements of the matrix are different from those of a normal mathematical matrix; these elements carry the semantic meaning of the text. Then some general algorithms can be derived from the matrix and are listed below:

Let a Role in a design text be R . The function f_{count} counts the element numbers of rows or columns of a matrix, then

$$R = \max(f_{count}(v_{x,y})) \quad (2)$$

While $v_{x,y} = 3$, i is any number between 0 to the max number of objects in a ROM diagram

Let the Action in a design text be Act , the function f_{act} is the number of predicate relations of R

$$Act = \max(f_{act}(R)) \quad (3)$$

By using the algorithms in graph theory, we can traverse the ROM diagram. This means that we can look up all the objects, which are words or phrases in the original natural language format, following the sequence of their semantic connections rather than the grammatical sequence in the original text. As a result, the final XRD keeps not only the vertices and edges but also the adjacency list (list of objects connected to current objects) in each object segment. This representation can thus assist with the identification of the priority order of each object from the semantic perspective. Furthermore, the number of relations to each object can also be treated as a property or weight of this object [33].

2.3 Formalization of Design Requirements

Since the ROM diagram forms a graphic representation of natural language, it can be used to store the semantic information of the design requirements. However, an ROM diagram performs only a linguistic analysis of the requirements text; it lacks the method for extracting modeling information from the diagram. Zeng has proposed a general methodology for the process to formalize design requirement. The input of the process is the design requirements text and the output is the formulation of these requirements.

In the formalization process, Zeng defines Ω as the structure of the engineering system, E as its environment, and S as the product. The engineering system is then decomposed into the following formula:

$$\oplus \Omega = \oplus (E \cup S) = (\oplus E) \cup (\oplus S) \cup (E \otimes S) \cup (S \otimes E) \quad (4)$$

Where $\oplus E$ is the structure of the environment and $\oplus S$ is the structure of the product.

Figure 11 illustrates the engineering system.

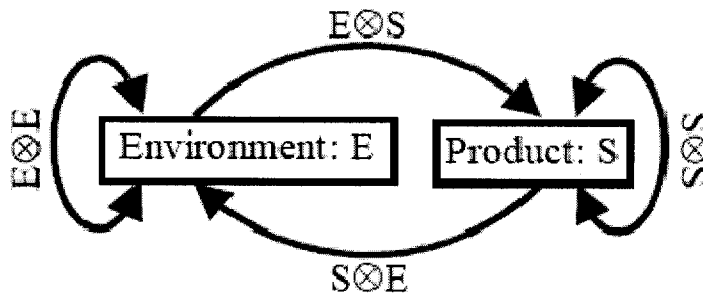


Figure 11 Engineering system [34]

By using linguistic analysis in the formalization process, the formula of the engineering system is derived from the natural language requirement. Figure 12 shows the formalization process for the requirements.

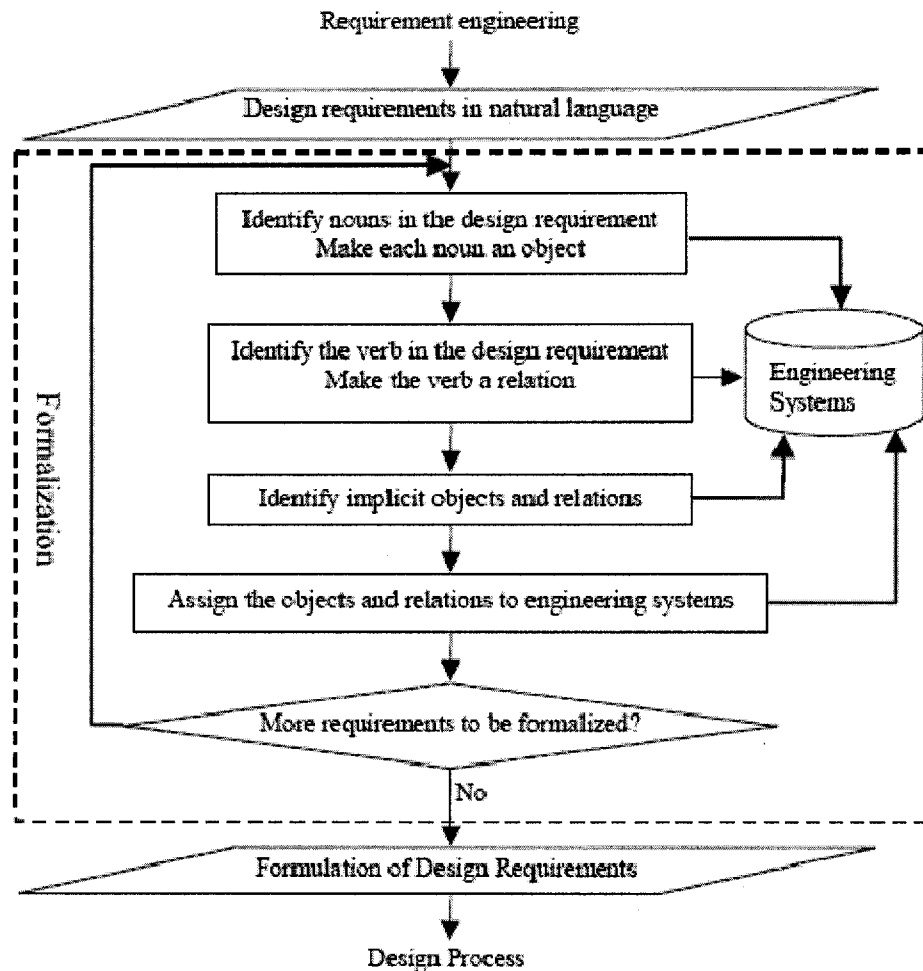


Figure 12 Formalization processes of design requirements [34]

2.4 Function-Behavior-State (FBS) Modeling

Although there is no precise and standard definition of function from the product requirement point of view, we can simply consider a function as some kind of actions that

fulfills the objective or part of the objective of one product. During a design process, the designer needs to specify the product structure with a function definition. This is the most important part in the early stage of design process.

Since the functions of the product and their relationships are the fundamental elements in product architecture design, it is crucial to model the functions during the design activities. Erden *et al.* [35] define the function modeling as the name given to the activity of developing the models of devices, products, objects, and processes based on their functionalities and the functionalities of their subcomponents. As one kind of framework to represent the Functions, Yasushi [3] has proposed a new scheme for functions: Function-Behavior-State (FBS) modeling, that defines a function as an association of human intentions and behavior and represents a design object hierarchically.

FBS Modeling represents Functions that are generated from product documents. The documents can be the design problem and the design solution. FBS modeling theory proposes a knowledge representation scheme for functions which define a set of representation definitions – function, F-B relationship, state, behavior, physical phenomena, and aspect. Figure 13 shows the relations in the FBS function scheme.

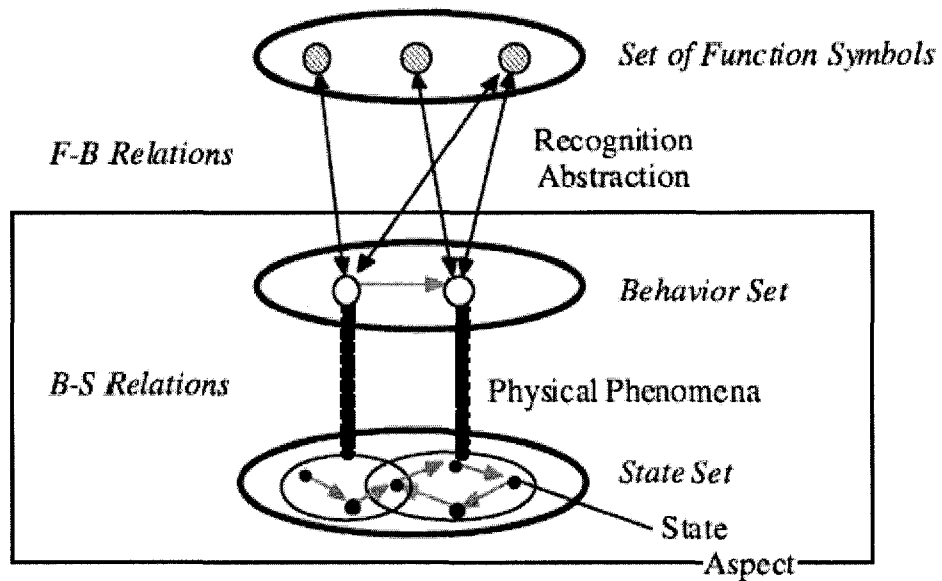


Figure 13 The relationship between function, behavior, and state[3]

FBS modeling theory is a systematical method of function modeling and gives a computer the necessary tools – the FBS modeler – to support conceptual design. The FBS modeler provides a function decomposition method. Umeda and Tomiyama divide this decomposition process into two categories – causal decomposition and task decomposition [3] and use these two decomposition methods in two different design phases. The task decomposition occurs in the first design phase and is used by the end user and the designer to decompose the design specification into each of the detail tasks with the assistance of a function knowledge base. The causal decomposition occurs in the second phase of the design process. The designer uses the causal decomposition method to decompose the behavior and the structure of the product with the assistance of a Behavior knowledge base. Figure 14 illustrates the architecture of the FBS modeler.

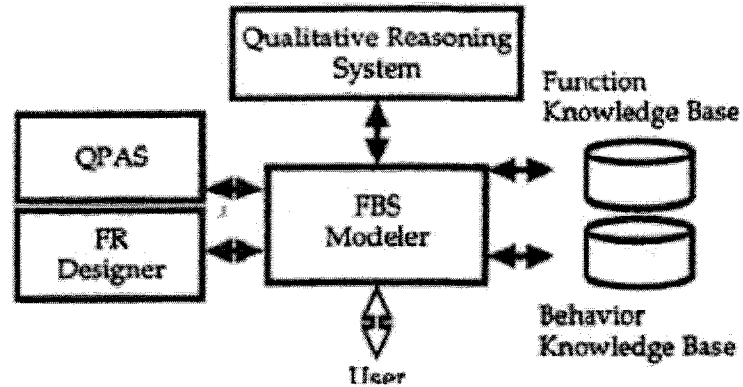


Figure 14 Architecture of the an FBS modeler [3]

As the product system becomes more and more complex, the interactions of the components within the system increase. Some unpredictable interactions may result in undesirable behavior. To deal with this problem, a system to detect the possible behavior is proposed for large product-system design, which is called a design interferences detector (DID). Tomiyama has proposed a general model – function-behavior-physical phenomenon-state model (FBPhPhS) - which extends the FBS model by including the physical phenomena between the state and behavior to support DID [35]. The FBPhPhS model is basically an integration of the FBS model with a qualitative reasoning system (QRS). Figure 15 shows the differences between FBS and FBPhPhS.

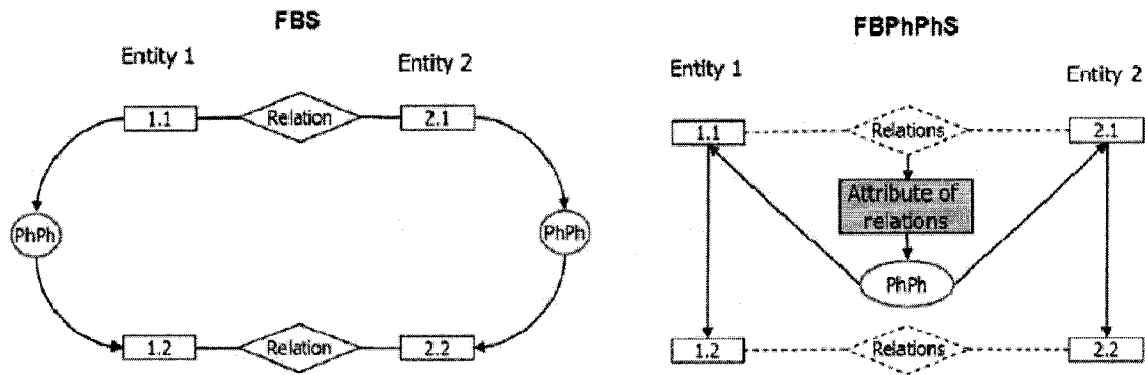


Figure 15 Differences between an FBS and an FBPhPhS model [35].

In Figure 15, we can see that the FBS model focuses only on the physical phenomena between each entity connected in a product system. While there exists some interaction between two entities that do not actually connect with each other – which means no relations between them. The FBPhPhS model includes these physical phenomena in the models by defining a set of attributes of relations and by using QRS to detect the interactions. Obviously, the FBPhPhS model is more suitable for dealing with the complexity of a modern product system. However, as our approach is mainly to target the automatic generation methodology, due to the limitation of the semantic mining of design documents, the interaction between unconnected components may not be easy to derive directly. Under such conditions, our approach will mainly focus on automatically formalizing the FBS models.

FBS modeling, especially the FBS modeler, gives a great methodology for function modeling. However, FBS modeling process basically is a brainstorm activity because it heavily relies on human interference. Due to the uncertainty of human thought (such as

different designer background, different design focus, etc.), it is impossible to get the standard FBS diagram from different designers. Our purpose is to follow the methodology provided by FBS modeling and finds a way to formalize FBS models automatically, thereby increasing the correctness and efficiency in the FBS conceptual design process. Finally with the assistance of those automatically formalized conceptual design diagrams, the system designer can better model product functions and architecture.

Chapter 3

Framework for Automatic Transformation from ROM Diagram to Modeling Languages

In requirements engineering, the formalization process for design documents is performed by engineer using semantic analysis and function decomposition methods. In function modeling process, engineers play the key role in the analysis of the documents. They find the role and functions of the product, decompose the functions and define the characterization of the representation definition of the function. To automatically formalize representation models from design documents, simulating the human activities of design process is a reasonable solution, especially for the requirements analysis and the decomposition process. However, unlike human activities, the automatic formalization of a design text relies on the understanding of natural language based product requirements. This thesis aims to perform the semantic analysis using the ROM diagram through simulating the manual modeling procedure and finally formalize UML and FBS models automatically.

3.1 General Framework for the Semantic Analysis of ROM Diagrams

3.1.1 Semantic analysis of ROM diagrams

Since the FBS or UML models are one kind of specification modeling method, the basic idea of our approach is to describe the key role and function correctly and completely. We give a general schema to support the analysis of the ROM diagram. The UML tries to

define the actor and action of a requirement, while FBS uses a function schema to represent function with function, behavior, state, aspect, F-B relationship, and B-S relationship. In our approach, it is hard to generate the above representation schema directly from the ROM diagram. After studying the theory of FBS modeling, especially for the FBS modeler decomposition method, we found that it is a reasonable method for decomposing the document with a set of schema to identify the role and action in the ROM diagram. FBS also gives some methods such as causal decomposition and task decomposition for dividing the function knowledge levels and clarifying their relationship. We found it is an intuitive way to represent product function and it is especially useful for conceptual design in design engineering. FBS modeling can be one fundamental theory in our project. We follow exactly the same logic in our approach.

With the support of Zeng's Environment Based Design theory [36], the product requirements can be categorized into eight levels: natural laws; social laws and regulations; technical limitations; cost, time, and human resource; basic functions; extended functions; exception control level; and human-machine interface, which is shown in Figure 16 [36]. Each level of requirements should have different syntactic criteria. For example, level of natural laws and rules may have the syntactic criteria like weather, wind and rain. On the contrary, level of basic function may only involve of actor and action. Classifying the requirements documents [37] and mining the aspects of requirements documents [38] are the necessary step in gaining the syntactic criteria from the design text. This step is also the pre-requirement in the automatic formalization of the

UML and FBS models.

From Zeng's Axiomatic Theory of Design Modeling[39], we have the mathematical foundation of requirement classification. Furthermore, based on the paper Classification of Product Requirements Based on Product Environment as the method, our research needs to find the algorithm to analyze the design text and to divide the text using syntactic criteria. This pre-processing of the design text will simplify our research and narrow our work on text semantic mining and model formalization.

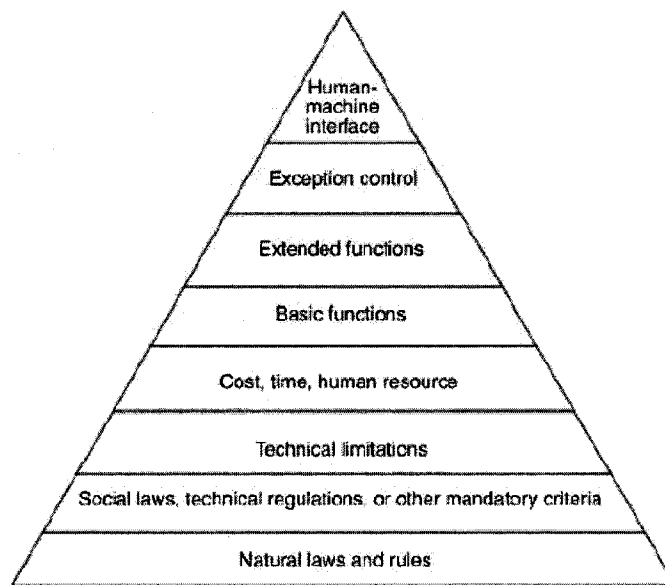


Figure 16 Eight levels of requirements[36]

3.1.2 Knowledge base for the ROM diagram

From Environment Based Design Theory, especially from the ROM diagram, the semantic meaning of a design text can be extracted and transformed into a mathematical data structure – normally forming a graphic map structure. This pre-research gives us the

43

opportunity to understand the natural language text and to lead to an ROM semantic decomposition process.

To achieve the automatic formalization of conceptual design models such as a FBS modeling, we need to first solve the following problem mentioned in Chapter 1 - How can the meaning of a text be captured automatically?

As discussed before, the ROM diagram contains the semantic meaning of the product requirements. Based on Zeng's EBD theory, all the design products can be treated as objects, meaning they have an environment in which they perform their functions. In this case, capturing the meaning of the design documents – or a scenario of procedures – can not leave the environment behind. These environments can be system working conditions, system working environments or the resources of a system, etc. All the environments can be seen as limitations or constraints when we mine the semantic meaning of product design documents. Developing a knowledge base for the ROM diagram is crucial in the automatic formalization process.

The knowledge base can also be seen as a criterion that testifies about all the objects from the ROM diagrams to find out the semantic meaning of the objects and their roles in original natural language requirements. This knowledge base can be a data set and a series of principles. The knowledge base reflects the definition of Aspect in FBS modeling, meaning predefining the limitations before modeling.

In this approach, we define the following criteria for semantic analysis in ROM diagram:

1. Product (center object)

No matter how complex a design text is, only one product will be described in each requirement document. Through ROM theory and linguistic analysis, the output of the ROM diagram should follow the same principles. Let *freq* be the function of counting the frequency of the objects in the ROM diagram. Let *act* be the actions sent out from the objects where *o* is the appearance of each object in the ROM diagram and *prod* is the product of a requirement:

Before merging the ROM diagram

$$Prod = \max (freq (o))$$

The most frequent object is the main object of a requirement.

After merging the ROM diagram

$$Prod = \max (count (act (o)))$$

By counting the number of actions, the largest number of actions should come from the main object.

By examining these two principles, we first verify the correctness of the ROM diagram and then get the central object of the whole requirement, which should be the product (a machine in this case).

2. Main actions

How to describe a product requirement correctly in documentation? The bottom line is to describe the product – or we can say a machine in the manufacture field – functioning completely and correctly. Let *Act* be the set of actions from any object, v be the function derived from each object. *Verbs* connect the actor and the object of the actions. From the ROM diagram and common sense, we get the following criteria for functions:

$$\begin{array}{l} \textit{Act} = v(o) \\ \\ \textit{Where } v(o) = \left\{ \begin{array}{l} \textit{true} \\ \textit{(if word class = verb and} \\ \textit{relation type is predicate)} \\ \\ \textit{false} \\ \textit{(otherwise)} \end{array} \right. \end{array}$$

(5)

We can then define the main function with the following formula:

$$\textit{Main Act} = v(\textit{prod})$$

(6)

Main Act is also a set of functions containing all the actions sent out from the product. By verifying the action criteria, we can examine the ROM diagram.

3. State changing actions

The product design requirements focus mainly on describing the procedure of a product such as a machine procedure. Using FBS modeling, the F-B relationship and the F-S relationship show the relationship between Function, Behavior and State. Zeng defines the behavior as performance and gives a scheme of performance in figure 17.[40]

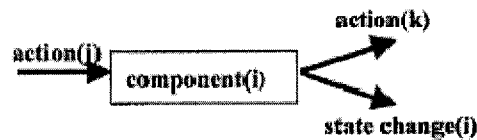


Figure 17 Performance scheme[40]

Since the action can be seen as a function in the design process, the performance (behavior) can be seen as the combination of the action result and the product state change.

To capture the semantic meaning of the ROM diagram especially for the product design problem, our approach needs to define the criteria for changing the state. Based on the performance scheme, the component can also derive from the performance after catching the key words that indicate the state changing. Let the f_{state} be the function that indicates the state changing of the product. Let $prep$ be the function to test the preposition object in ROM diagram, r_t be the relation from ROM diagram and t_x be the relation type from ROM diagram.

$$\begin{aligned}
 f_{state}(act(o), prep, r_t) &= \begin{cases} true \\ (if\ prep\ is\ true) \\ false \\ (if\ prep\ is\ false) \end{cases} \\
 prep &= r_t(t_1) \cup r_t(t_2) \\
 comp(f_{state}, act(o)) &= \begin{cases} true \\ (if\ f_{state}\ is\ true\ and\ act(o)\ is\ true) \\ false \\ (otherwise) \end{cases}
 \end{aligned}$$

(7)

From the formula, the changing of state is determined by the preposition criteria in a ROM diagram. This approach follows the natural law of human speech. For example, a machine performs some function can be seen as an action and reaction to the destination. As defined in Figure 17, the behavior is the combination of action and change of state, and then the change of state usually occurs after an action. After the change of state point is retrieved from ROM diagram, the component of the product can be selected if it connects to an action. Using this logic, the change of state point and the product component can be retrieved from ROM diagram.

4. Environment

From Zeng's EBD theory, Environment is a general term. As inside the engineering system, the component and the workflow are the environments for system functions. At the component level, the attribute and actions are the inside environment. Moreover,

outside the whole system, human interaction, nature law, etc., are the environment for the system. For the sake of the simplicity of our research, we focus on the environment that acting directly on the product. So we define the environment *Env* criteria as follows, where *Comp*, *Attr* and *OutEn* denote sets of components inside the system, Attribute of the Component and Environment outside the system:

$$Env = \{Comp, Attr, OutEn\}$$

(8)

The detailed mechanism for the generation of the proper environment definition will be discussed in the parts of this paper dealing with the formalization rules.

3.2 Framework for Automatic Generation from ROM Diagram to UML

Figure 18 shows the general procedure of how engineers understand a requirements text. First, an effective reader tends to find keywords when s/he tries to understand a sentence and to find key sentences when s/he tries to understand a paragraph. Secondly, the main verb seems to carry the most important message in a sentence. Thirdly, a key sentence (or a topic sentence) in a paragraph is always carries the most amount of relations inside a paragraph. Finally, every actor in any requirement text must be a noun and every action in any requirement text must be a verb.

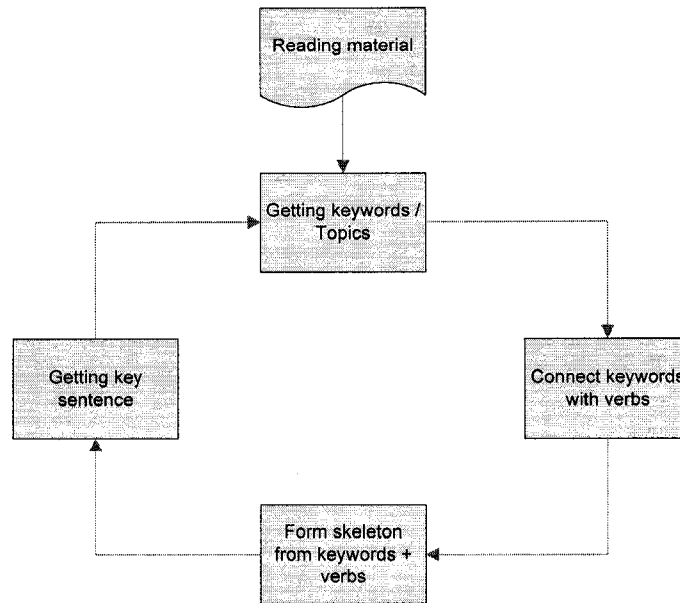


Figure 18 Representations for describing design information

In the ROM diagram, each object contains a word or a phrase that has a part of speech or its equivalent, based on which type of object can be analyzed. An ROM diagram also has three types of relations that can hold the semantic meaning between two objects. By analyzing the type of relation, we can easily get the semantic meaning of each relation between two objects and finally get the keywords in a sentence.

Through the priority of each object (the number of adjacency list for each object), the object's importance can be ranked in a sentence or a paragraph. As the priority indicate the action send out from a object and based on human analytic logic, those having a higher priority carry more meaning in a given text; thus, they should be taken as the candidate for actor and can be further used for generating class diagrams. As the predicate relation indicate the action send out from one object and our approach is intent

to find out the actor and action of a design documents, then only a predicate relation is selected to determine the priority in each object.

3.3 Framework for the Automatic Formalization from ROM Diagram to FBS Models

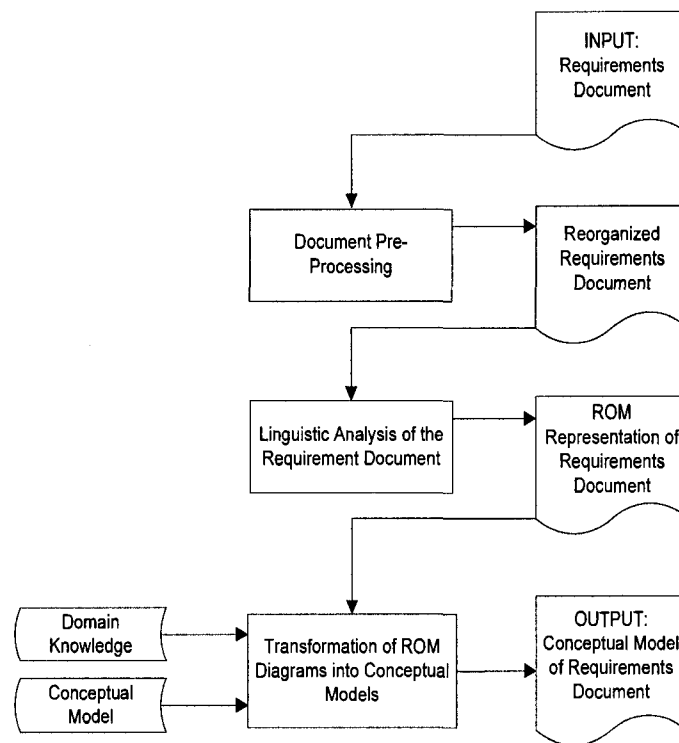


Figure 19 Framework for the transformation of a requirements document into conceptual models

Figure 19 shows the framework of the automatic formalization process. First, the input requirement document described in natural language will be processed with some document refining process such as asking the right question as proposed by Zeng and Min [2]. If the question-asking strategy is used, the document can be pre-processed and

the original document can be made more standard and complete. After the pre-processing, the document will go through a linguistic analysis process by using the computer tools from the ROM project, tools called ROMA tools. The output of the ROMA is the ROM diagram, which is discussed above. It is also the input of our automatic formalization process to the FBS models. The domain knowledge in the framework can be the function knowledge of the FBS model and the criteria for finding the FBS scheme, which is discussed above. Using the FBS modeler as the reference for function decomposition will lead to the FBS models.

In the ROM diagram, each object contains a word or a phrase that is a part of speech or its equivalent. An ROM diagram also has three types of relations that have the semantic meaning between two objects. By analyzing the type of relation, we can easily get the semantic meaning of each relation between two objects and finally get the keywords in a sentence.

Based on the priority of each object, the object's importance can be ranked in a sentence or a paragraph. The object having a higher priority has more meaning in a given text; thus, they should be taken as a product and can be further used for generating actions, environment, component and attribute. The computer tools can determine the function scheme that is suitable for an ROM diagram by automatically using the criteria from the function knowledge base. Then, using an FBS decomposition strategy with certain traversal algorithms, the main action can be decomposed into sub functions. By keeping the same decomposition procedure, the level of functions and the relation of function and

behavior will be clarified until the decomposition level reaches the physical level such as the component and outer environment. It should be emphasized that the relation type in an ROM diagram plays the key role in identifying the function scheme and in decomposing the function structure. Moreover, as mentioned above, the preposition and the predicate relation are the most important in our approach.

Chapter 4

Transformation from ROM Diagram to UML Diagram - R2U

The automatic generation of a UML model relies on the full understanding of the natural language based requirements description. For example, if an engineer wants to draw a use case diagram, he or she needs to understand the requirement at first and then get the actor and actions related to the UML standard. Our research aims to simulate the human activities in the requirement analysis process and to automatically generate UML diagrams by using a software system.

4.1 Generation Rules

Based on the previous discussions, it is possible to get the semantic structure of a requirement text and then automatically generate UML models based on that semantic structure. This subsection describes the procedures and rules for the automatic generation of UML models from the ROM diagram representing a text.

Our current research is mainly focused on Use Case Diagrams and Class Diagrams. Use Case diagrams have two types of objects – actor and action whereas Class diagrams have class name, method and property. For the Use Case diagram, an actor corresponds to an object that is a noun with the highest priority. The action should be a verb object related to the keywords of the identified actor. The class diagram also comes from the object's ranking of priority. The only difference between actor and class is the meaning of the keywords that will be discussed further below.

Generation rules:

Pre-condition:

- Role = Object in ROM diagram with noun class property
 - ***Class* \in *Role***
 - ***Actor* \in *Role***
 - Action = verb phrase
 - Method = verb from noun phrase
-

Generation Rules:

- Actor and Class = Highest priority Object in ROM Diagram
 - Priority of Object = Number of predicate relations to the object
 - Actor = outside of the system
 - Class = inside the system
 - Action = System External User's action
 - System External User's action = Actor Object in ROM diagram with verb class + Object in ROM diagram with preposition related
 - Method = System Internal action
 - System Internal action = Class Object in ROM diagram with verb class + Object in ROM diagram with preposition related
-

Below are the UML generation procedures:

Generation Procedure:

- 1 Get a set of the object with the highest number of neighbors on its adjacency list in each sub-ROM diagram
 - 2 Check word class to select the noun Object as Class object
 - 3 Iterate the list of Class objects. For each class object search the dictionary to determine if the object is external or internal to the system (for example, customer is always outside the system while a database may be inside a system)
 - 4 Traverse ROM diagram using a graph search algorithm
 - 4.1 Get the first Object in the ROM Diagram correlated to the class object (also considered as keywords in paragraph) with the first half predicate relation (From ROM, the first half predicate relation connects its subject and verb in a sentence)
 - 4.2 This object forms a method object in the class diagram
-

-
- 4.3 Traverse from this object to reach an object correlated to the current object
 - 4.4 If a relation is the second half of a predicate relation (acts as a verb to an object in English grammar), then it forms an action (skeleton of the sentence) to the actor (keywords)
 - 4.5 If a relation is a connection relation (acts as a verb to a noun phrase in grammar), and the related object is one of the keywords, then it forms as inter actor action (skeleton of the sentence) from one actor to another actor (keywords)
 - 4.6 Repeat steps 4.1 to 4.5 while iterating the adjacency list in current class object.
 - 5 From each keyword, find any connection relation to an object in the ROM diagram which contains noun word class
 - 6 Verify the object found in step 5 in the class object list. If they are equal, then go to step 7.
 - 7 This relation forms the connection between the class object in the UML class diagram.
-

Using the generation rules, the human semantic capturing process can be simulated by catching the keywords, finding the preposition connect to the keywords, finding relation of each keyword and getting the semantic meaning of ROM diagram.

This procedure takes the input ROM diagram (in XRD form) as a directed graph and uses graph traversal algorithms during the analysis process. For the semantic part, this procedure uses the meaning of relation defined in the ROM to rank the priority of an object in a ROM diagram and identifies the semantic meaning of each sentence by catching its skeleton.

It should be pointed out that the rules above are still preliminary. Further experiments are needed in order to deal with a broader range of problems. A software prototype has been developed for this purpose.

4.2 Software Prototype

The software prototype R2U has been developed based on the generation rules given in the previous section. The prototype is implemented in the windows environment by using C#. The input of this software is an XRD file and the output is UML diagrams. Figure 20 shows the data flow of the R2U application.

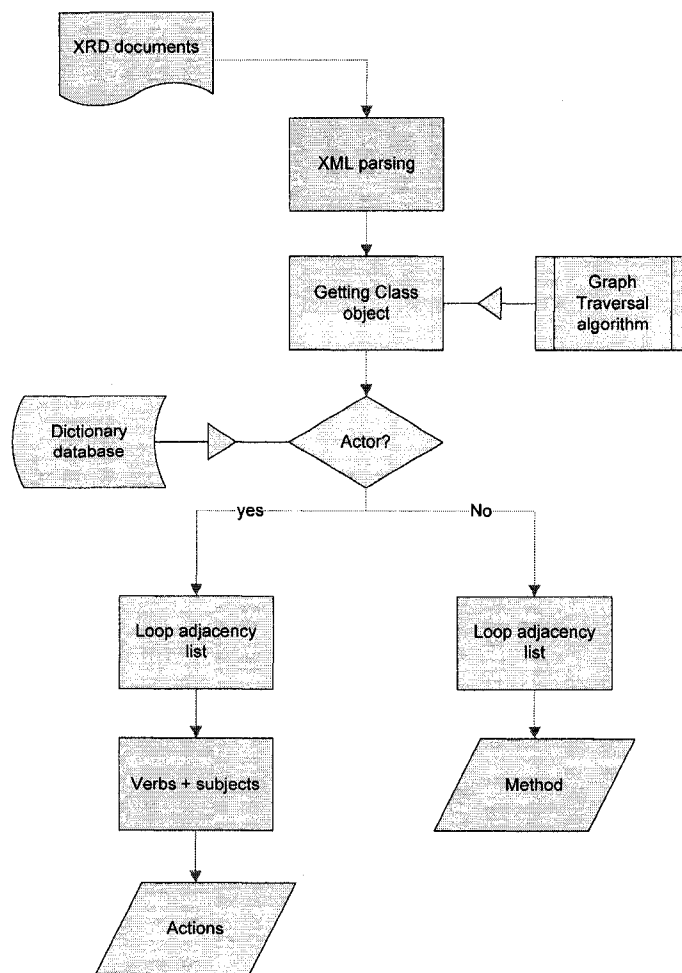


Figure 20 Data flow diagram of an R2U application

R2U has two critical functional parts. One is the XML parsing, which is combined with graph traversal algorithms to ensure certain objects in a ROM diagram will be visited. The relations in the ROM diagram have the most important information to actually determine the traversal sequence. The traversal algorithm simulates the logic of the human analysis process.

The following pseudo code shows the basic function of the ROM diagram traversal process:

```
Start  
Declare XMLReader Variable  
Declare Graph data structure  
Reading XRD  
Fill out Graph structure  
Foreach Node  
    Current_Priority = Amount (relation in adjacency list = predicate)  
    If Current_Priority > Max_Priority  
        Max_Priority = Current_Priority  
    End If  
End Foreach  
Fill out Class list  
Foreach Class list  
    Lookup dictionary  
    If Class = External User  
        Actor_List.Add(Class)  
    End If  
End Foreach  
Stop
```

The following pseudo code shows the basic function of the semantic analysis system:

```
Start  
Declare Action list  
Declare Method list  
Foreach Class in Class list  
    If relation type = predicate  
        Method = predicate->Object  
        Fill out Method List  
    End If  
End Foreach  
Foreach Actor in Actor list  
    Traversal (Actor)  
        If relation type = predicate  
            Actor = predicate->Object  
            Traversal (Actor)  
        Else  
            Fill out Action List  
        End If  
    End Traversal  
End Foreach  
Stop
```

4.3 Case Study

To test the approach proposed in this thesis, some experimental results from a case study are presented in this section. The test case is a small requirement text describing the scenario of a POS management system in a common supermarket environment. As discussed above, the original input of our project is in a natural language based requirement description given below.

1. *The customer arrives at a POS checkout with goods.*
2. *The cashier starts a new sale.*
3. *The cashier enters the item identifier.*
4. *The system records the sale-line's item and presents an item description.*
5. *The cashier repeats steps 2-3 until it indicates it's done.*
6. *The system presents the total price with taxes.*

7. The cashier tells the total price to the customer and asks for payment.
8. The customer pays and the system handles the payment.
9. The system logs the completed sale.
10. The system sends the sale and payment information to the external Inventory Systems.
11. The system presents the receipt.
12. The customer leaves with the receipt and the goods.

This requirement text shows an entire check-out process that occurs in most stores. From the system design point of view, the product of this requirement is a system and there is also an external actor and an internal actor in the system. By analyzing this requirement, the designer can identify the customers and some basic functions of this system.

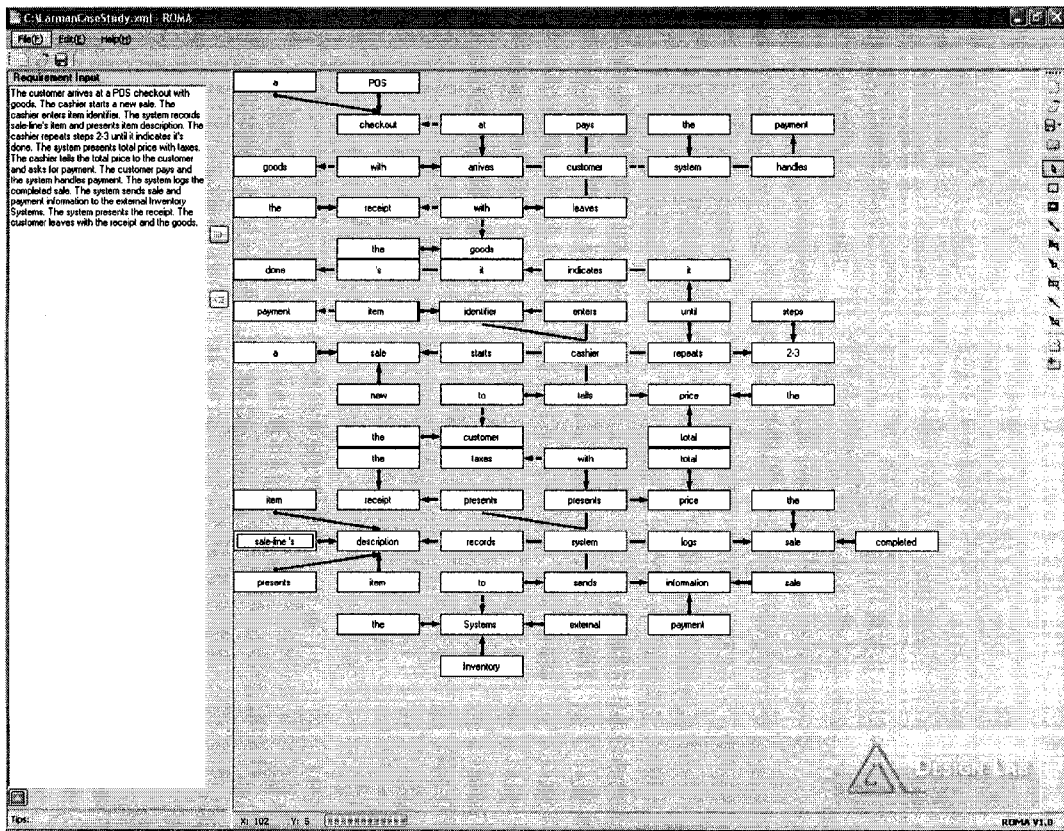


Figure 21 ROM diagram of the test requirement text

Figure 21 shows the ROM diagram of the test requirement text, which was generated by the ROMA system. Internally, the ROM diagram is represented in XRD format, which is used to generate the UML diagrams by the R2U system.

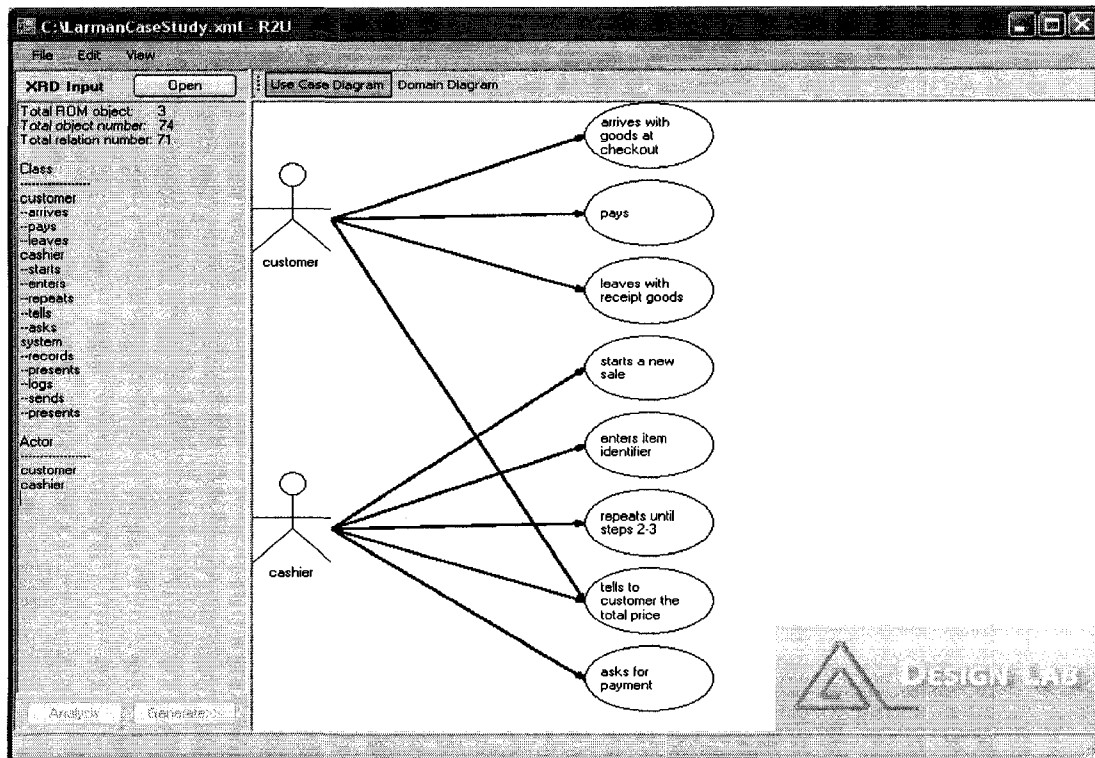


Figure 22 Use Case diagram output of R2U 1.0

From the XRD file generated by the ROMA system, the R2U software automatically generates and displays the UML diagrams, based on the generation rules introduced in the previous section. Figure 22 and Figure 23 show the use case and class diagrams of the test case, respectively.

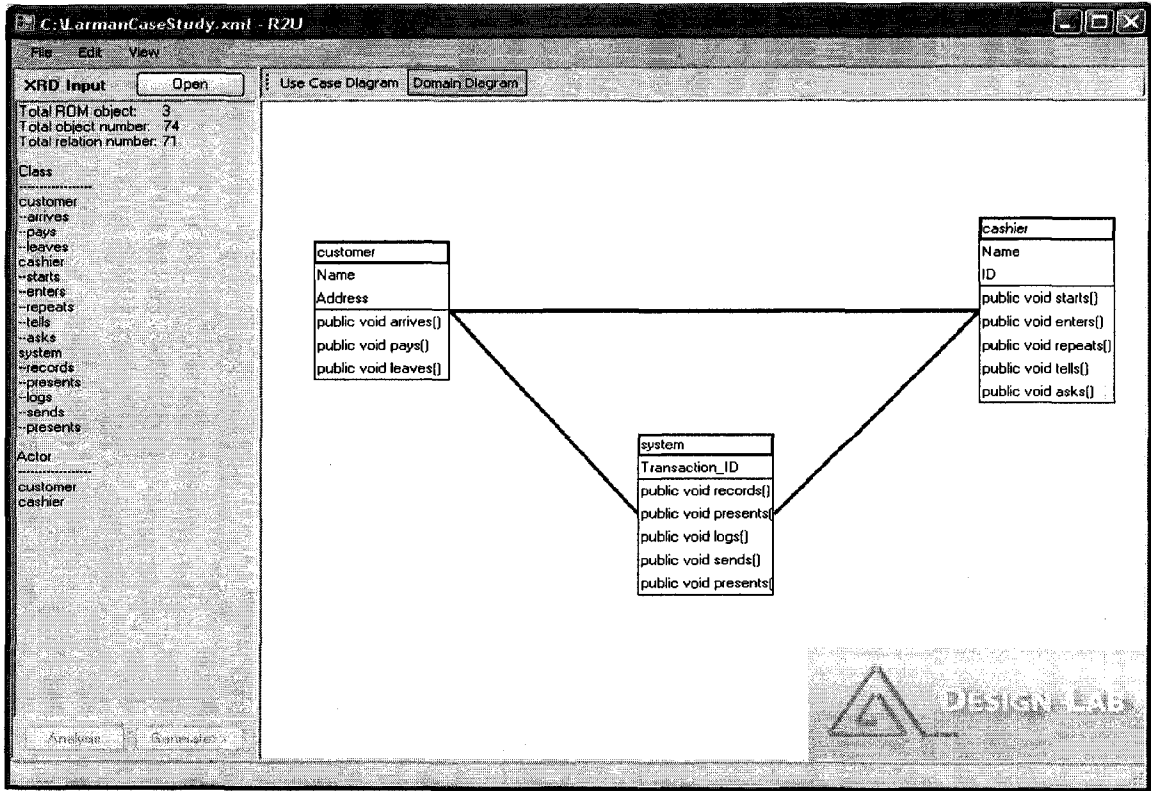


Figure 23 Domain Diagram (Class Diagram) output of R2U 1.0

Chapter 5

Transformation from ROM Diagram to FBS Models - R2FBS

Based on the previous discussions, it is possible to get the semantic structure of a requirement text and then automatically formalize FBS models based on the semantic structure. This subsection describes the procedures and rules for the automatic formalization of FBS models from the ROM diagram representing a text.

5.1 Identify the FBS Schema from ROM Diagram

Our research simulates the FBS modeling procedure. The FBS modeling procedure deals with function modeling by first defining the function scheme and then by decomposing the function. Our automatic formalization process follows the same logic and divides our approach into three parts – locating the function key words in the ROM diagram, which is described in the framework chapter, decomposing the function level and mapping to FBS scheme to finally form the FBS models. This decomposition process answers the following question proposed in Chapter 1: How to define the function representation scheme of product function described by FBS modeling and mapping with ROM element to capture the semantic meaning of functions?

5.1.1 Function decomposition rules for ROM diagram

The FBS modeling proposes a representation schema by giving the following definitions: Functions, F-B relationship, state, behavior, physical phenomena and aspect. Since our research focuses on the computer-assisted formalization of FBS models, getting the

scheme automatically is crucial to our approach.

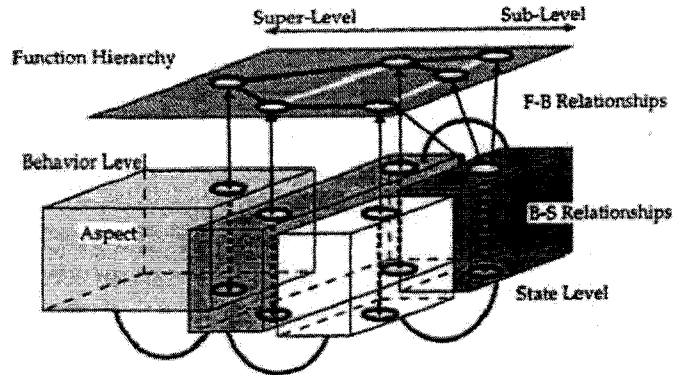


Figure 24 A FBS diagram [3]

FBS modeling uses the FBS modeler to assist a function decomposition process to simulate the human activities by decomposing the function into lower-level sub-functions and finally by reaching the substantial components as shown in Figure 24.

By solving the first question about the capture of the semantic meaning of ROM diagram, we have established the knowledge base of our approach. By using these criteria, we found a set of product elements that can be extracted from a ROM diagram and describe a product function completely. These elements can also be seen as another scheme suitable for ROM diagrams. As discussed above, an ROM diagram is not a systematic modeling language. We can treat an ROM diagram only as a data structure to restore the semantic meaning of a text or to treat it as an output after performing a linguistic analysis. We can't derive the FBS scheme directly from an ROM diagram. We need to examine or traversal the ROM diagram and find the key element following the criteria discussed

above. Let FBS be the set of schemes defined in FBS modeling and RC be the set of schemes extracted from the ROM diagram. The following formula shows the definition:

$$FBS = \{E, F, F - B, S, B, B - S, PP, A\}$$

(9)

Where E is the entity, F is the function, $F-B$ is the F-B relationship, S is the state, B is the Behavior, $B-S$ is the B-S relationship, PP is the physical phenomena and A is the Aspect:

$$RC = \{P, A, En, Comp, Attr\}$$

$$En = En_{inside} \cup En_{outside}$$

$$En_{inside} = Comp \cup Attr$$

(10)

Where P is the product, A is the action, En is the environments including those outside system environment and those inside the system environment, $Comp$ is the component, and $Attr$ is the attribute.

After getting the RC from the ROM diagram, mapping it into the FBS modeling scheme is the next approach. Part of the mapping mechanism is shown below:

$$E = P \cup Comp$$

$$F = A(o) \cap v(P)$$

...

$$A = E_n$$

(11)

We can use the formula given before as the bridge from the ROM diagram to the FBS scheme, making it possible to finally formalize the FBS models automatically. The only

problem is the F-B relationship. Also the B-S relationship can not be derived by using the mapping rules above. Therefore, we need to find another methodology to clarify the hierarchy of the functions level and relationship. This approach also intends to figure out the following question proposed in Chapter 1 - How can the right FBS models be derived by simulating the function decomposition procedure provided through the FBS modeler?

The Function-Behavior relationship and the Behavior-State relationship illuminate the connection inside the FBS models. In another words, these two relationships divide the whole function model into different levels hierarchically. As shown in Figure 24, after solving the first two problems, we get the different parts of the function. However, we still have no idea about the hierarchy of the function models, and the entire element or schemes are separated. This section of the present paper presents a possible solution to the problem and finally gives a method to finish FBS modeling procedure.

After matching the criteria and extracting the key function element from a ROM diagram using the approach discussed before, we know the whole frame of the function from the requirement document described in natural language. To formalize a function model such as an FBS model, knowing the scheme is not sufficient. We still need to explore the structure and relationship of the function and then to formalize the function models.

Tomiyama has proposed a function hierarchical decomposition method for FBS modeling. This decomposition method simulates human activities in function modeling. For example, when an engineer designs the product, the traditional way is to find the

main function by understanding the requirement documents. Then the designer goes through the documents and finds more information to try to decompose the main function into sub-functions. This decomposition process should be a recursive process by which the designer keeps exploring the lower level of sub-functions and going back to the upper level to verify the sub function's correctness. This decomposition process is normally affected by the designer's understanding and experience, which usually differs from designer to designer. Moreover, only the designer can determine when the decomposition process should stop and at what level of sub-function. In FBS, it provides the FBS modeler to simulate this manual decomposition process discussed above. Consider the FBS modeler is a computer tool that supports conceptual modeling. It uses the knowledge base to store the function knowledge that can support the function deposition. Also, the FBS modeler divides the decomposition process into two categories: causal decomposition and task decomposition. Task decomposition results in the sub-functions related to the knowledge while casual decomposition results in the sub-function related to the physical features in the system.

In our approach, we have tried to follow the same logic as that of the FBS modeler. However, considering that our approach is trying to formalize the FBS models automatically which means minimizing human interaction, the decomposition method may be slightly different from that of the FBS modeler. With the help of the linguistic analysis tools mentioned above, ROMA and its output ROM diagram, it is possible to find a methodology to perform the function decomposition automatically. The basic idea

of our approach comes from the traversal method in computer science. Like the FBS function decomposition method, the starting point of our approach in function decomposition is the main function in the requirement. Unlike FBS modeling, the input of our approach is not the requirement document but a graphic representation of the requirements, that is, the ROM diagram. As discussed above, this starting point should be a unique one: the product of a requirement. To simplify our algorithm in traversal of the ROM diagram, we do not select the FBS modeling element such as function, behavior, etc. but select the function element we created for the ROM diagram – actions, environment, component, etc. – to work as the node during the traversal process and to connect all the elements using FBS decomposition principles. Then after we reach some node indicating a component or outside system environment, the traversal stopped and recursively goes back to the start node to verify this sub-function’s correctness. By using this traversal logic, we simulate a recursive function decomposition process that is similar to human actions and that follows the FBS modeler principle. The following formula has been derived to support our function decomposition process:

$$\begin{aligned}
 &function_{main} = prod + (\exists a(prod): a(prod) \cap r_t \text{ is predicate}) \\
 &function_{decp} = function_{sub} + en
 \end{aligned}$$

(12)

Using the logic to go through the ROM diagram and to combine with mapping algorithms from the ROM function element to the FBS function scheme, we can define the sub-function. Then through the recursive traversal algorithms we can verify the sub function.

In the mean time, after verifying the sub function, we can get the behavior from the action and we can indicate the state change point of the requirement description. Figure 25 shows the logic of function decomposition:

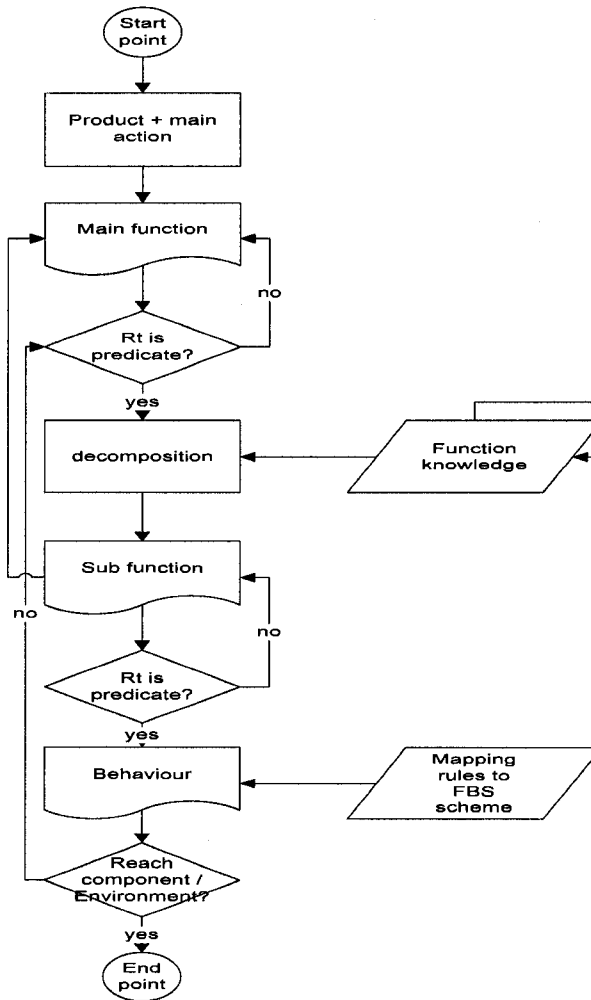


Figure 25 Function decomposition logic

5.1.2 Mapping function elements to FBS schema

The mapping process can actually start simultaneously with the decomposition process.

As the decomposition process begins with the main function identified from the first step, traversal algorithms may follow the FBS decomposition method. Until the process reaches the component, a behavior can be found, and in the mean time mapping the sub function to the FBS modeling scheme may save time. Then the formalization process can be divided into two main procedures: locate the function element and the function decomposition. The following parts of this section explain the rules we researched to assist the two processes.

Below are the function elements locating rules:

Location rule:

Pre-condition:

- Function Element definition
 - a) Product is a unique noun object with highest priority.
 - b) Environment is object connected with preposition object
 - c) Environment can be an inner environment and an outer environment for a product
 - d) Inner environment can be a component or an attribute of the component depending on the induct object's word class
 - e) Performance is the action and reaction on a component combined with the component itself.
 - Function Element searching Criteria (see research methodology part of this paper)
-

Procedures:

- Product = Highest priority Object in ROM Diagram
 - Priority of Object = Number of predicate relations to the object
 - Action = verb object connected with predicate relation as input relation
 - Main action = Action sent out from the product
 - Preposition object = object connected to a predicate relation as an output relation and constrain relation as output relation
 - Environment = connected to preposition object with predicate relation connect to preposition object
 - Physical phenomenon = environment object which connected preposition object and verb object
 - Component = physical phenomenon – outside system physical
-

-
- Outer Environment = physical phenomenon - component
 - Inner Environment = environment – physical phenomenon
 - Attribute = inner environment – unrelated to component object
-

Below are the function decomposition rules:

Decomposition rules:

Pre-condition:

- Function Element gain from first step
 - Mapping to FBS scheme algorithms
 - FBS modeler decomposition method
-

Procedure:

- Start point = product
 - Main function = main actions
 - Action object = component (if predicate relation indicates the next adjacent object is a verb)
 - Action object = $\left\{ \begin{array}{l} \text{physical phenomenon} \\ \text{(if predicate relation} \\ \text{indicate the next} \\ \text{adjacent object is verb)} \\ \text{others} \end{array} \right.$
 - Manufacture = most frequent physical phenomenon + inside the system
 - Behavior = actions + component
-

Below are the function formalization procedures:

Formalization Procedure:

- 8 Get the highest adjacency list with predicate relation type amount Object in ROM diagram
 - 9 Save the position of the selected object in ROM diagram and set it as the product of system
 - 10 Fill the main action list with traversal function where relation type from product is predicate
-

-
- 11 Fill the preposition list with traversal function where all objects with relation type suitable to preposition criteria proposed above
 - 12 Fill Environment list by go through the preposition list follow certain criteria
 - 13 Iterate the object of environment list, for each environment object lookup a dictionary and function knowledge base to determine if it's outside the system.
 - 14 Traverse ROM diagram using FBS function decomposition method and graph search algorithm
 - 14.1 Set the traversal start point with product
 - 14.2 Traverse from the start point and fill the action list
 - 14.3 Iterate the action list to the action object
 - 14.4 Identify the object of an action is a physical phenomenon or not
 - 14.5 If not physical phenomenon, go to step 7.6. Otherwise, go to step 7.7
 - 14.6 Restore the action object as new start point and then repeat step 7.2 to 7.5
 - 14.7 Traverse back to gather behavior information using formalization rules and mapping rules
 - 15 Using mapping strategy to map all the stored FBS scheme and output the structure of function models.
-

Using the formalization rules, the human semantic capturing process can be simulated by catching the product, finding the function element scheme, finding the preposition of keywords, decomposition of finding actions, mapping to existed FBS models and getting the function design structure and relationship of a product from ROM diagram.

This procedure takes the input ROM diagram (in XRD form) as a directed graph and identifies the element of function design and then uses graph traversal algorithms during the function decomposition process. For the semantic part, this procedure uses the different meaning of relation defined in the ROM as the semantic tool to rank the priority of an object in a ROM diagram and identifies the semantic meaning of each sentence by catching its skeleton.

It should be pointed out that the rules above are still preliminary. Further experiments are

needed in order to deal with a broader range of problems. A computer tools implementing the derived rules has been developed for this purpose.

5.2 Transformation Algorithm

The automatic FBS models formalization computer tool has been developed based on the formalization rules given in the previous section. The software prototype is implemented in the windows environment by using C#. The input of this software is an XRD file and the output is FBS models and function design analysis documents. Figure 26 shows the data flow of this approach.

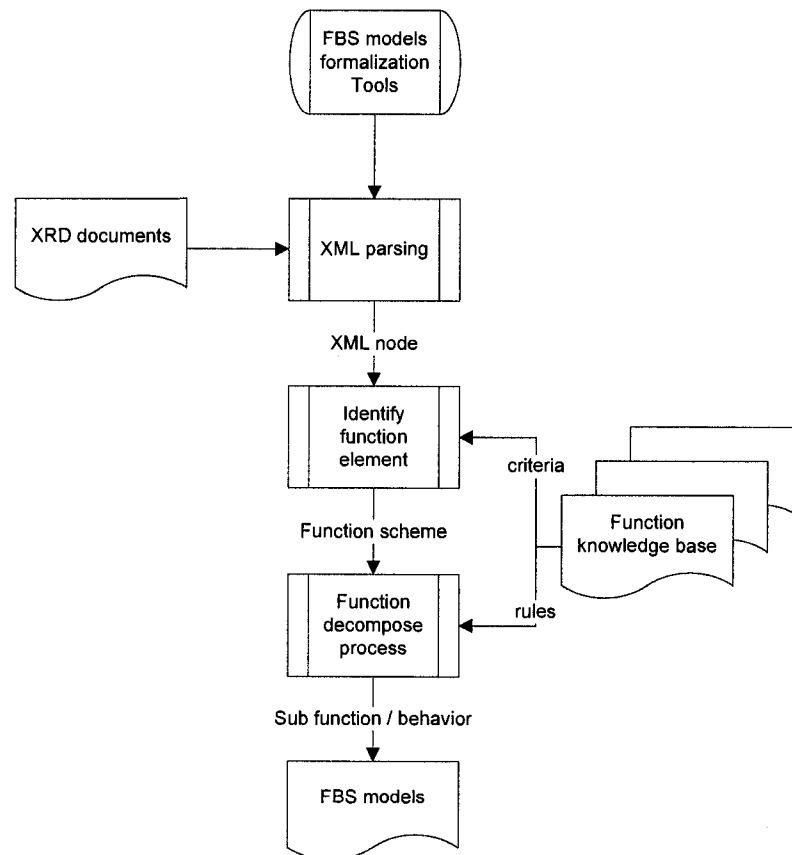


Figure 26 Data flow diagram of automatic formalization tools

The computer tool called R2FBS has two critical functional parts. One is the XML parsing, which is combined with graph traversal algorithms combined with function criteria. Relation in the ROM diagram is the most important information for the traversal algorithm, which actually determines the traversal sequence. The other is an algorithm simulating the logic of the FBS function decomposition process.

The following pseudo codes show the basic function of identifying the function element with some ROM diagram traversal algorithms:

```
Start  
  
Declare XMLReader Variable  
  
Declare Graph data structure  
  
Reading XRD  
  
Fill out Graph structure  
  
Foreach Node  
  
Current_Priority = Amount (relation in adjacency list  
= predicate)  
  
If Curent_Priority > Max_Priority  
  
Max_Priority = Current_Priority  
  
End If  
  
End Foreach  
  
Product = Max_Priority  
  
Action list = action (Product)  
  
Foreach Node
```

```

If (relationtype (Node) =predicate)
  && relationtype(Node)=constrain
    Fill preposition list = Node
  End If
End Foreach
Foreach preposition
  If neighbor (preposition) = true
    &&
    relationtype(neighbor(preposition))=predicate
    Fill Environment list
  End If
End Foreach
Foreach neighbor
  If wordclass (neighbor (preposition)) =v
    Fill component list
  End If
End Foreach
Stop

```

5.3 Case Studies

5.3.1 Formalization of requirements text

To test the approach proposed in this paper, some experimental results from a case study are presented in this section. The test case is a small requirement text describing a

product requirement for coating small food such as donuts with powder. As was discussed above, the original input of our project is a natural language-based requirement description given below.

The coating machine will coat a small, flat, cylindrical product with a powder. The semi-finished product is made by extrusion, at the rate of about one piece per second. The product can fall directly from the extruder on to the coating machine. After coating the product, the coating machine will deliver the product to a conveyor belt and transport the product for wrapping and packaging.

--select from case 3 design coating machine, Practical Studies in Systematic Design [41]

This requirement text shows a simple auto-coating system. From the system design point of view, the product of this requirement is a machine. There are also external and internal environments to the system. By analyzing this requirement, the designer can identify the main function and some basic components of this system.

Figure 27 shows the ROM diagram of the test requirement text, which was generated by the ROMA system. Internally, the ROM diagram is represented in XRD format, which is used to formalize the FBS models by the use of R2FBS computer tools.

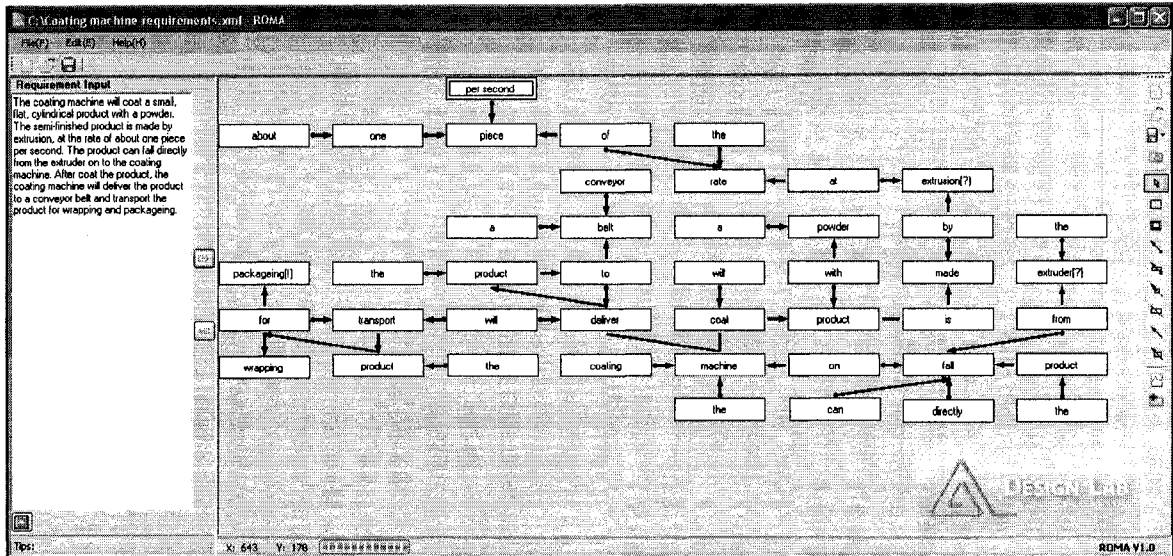


Figure 27 ROM diagram of the test requirement text

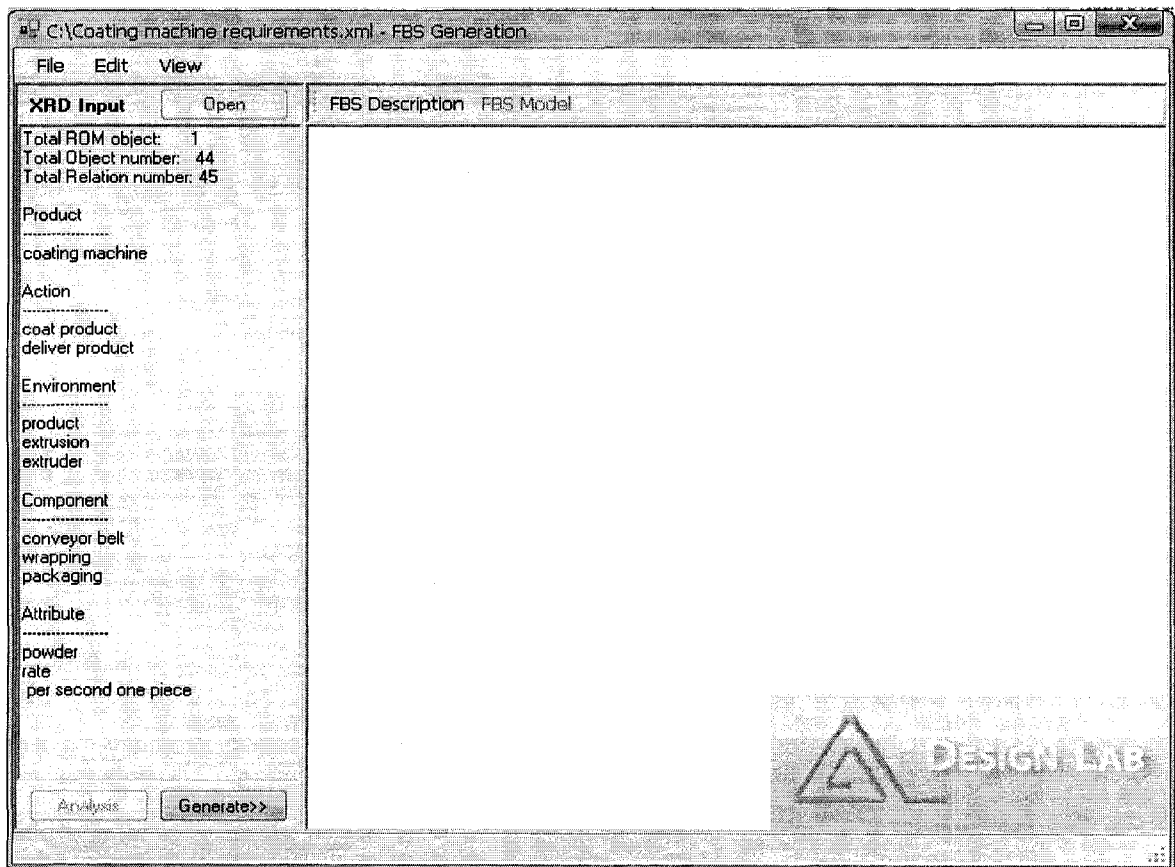


Figure 28 Function schema gain from a ROM diagram by R2FBS 1.0 for requirements

From the XRD file generated by the ROMA system, R2FBS software will automatically identify the function elements through certain criteria proposed on this paper as shown in Figure 28, based on the function elements generated from the ROM diagram and the FBS decomposition method, our automatic formalization rules will form the final output, namely the FBS models. Figure 29 shows the FBS models output of the test case, respectively.

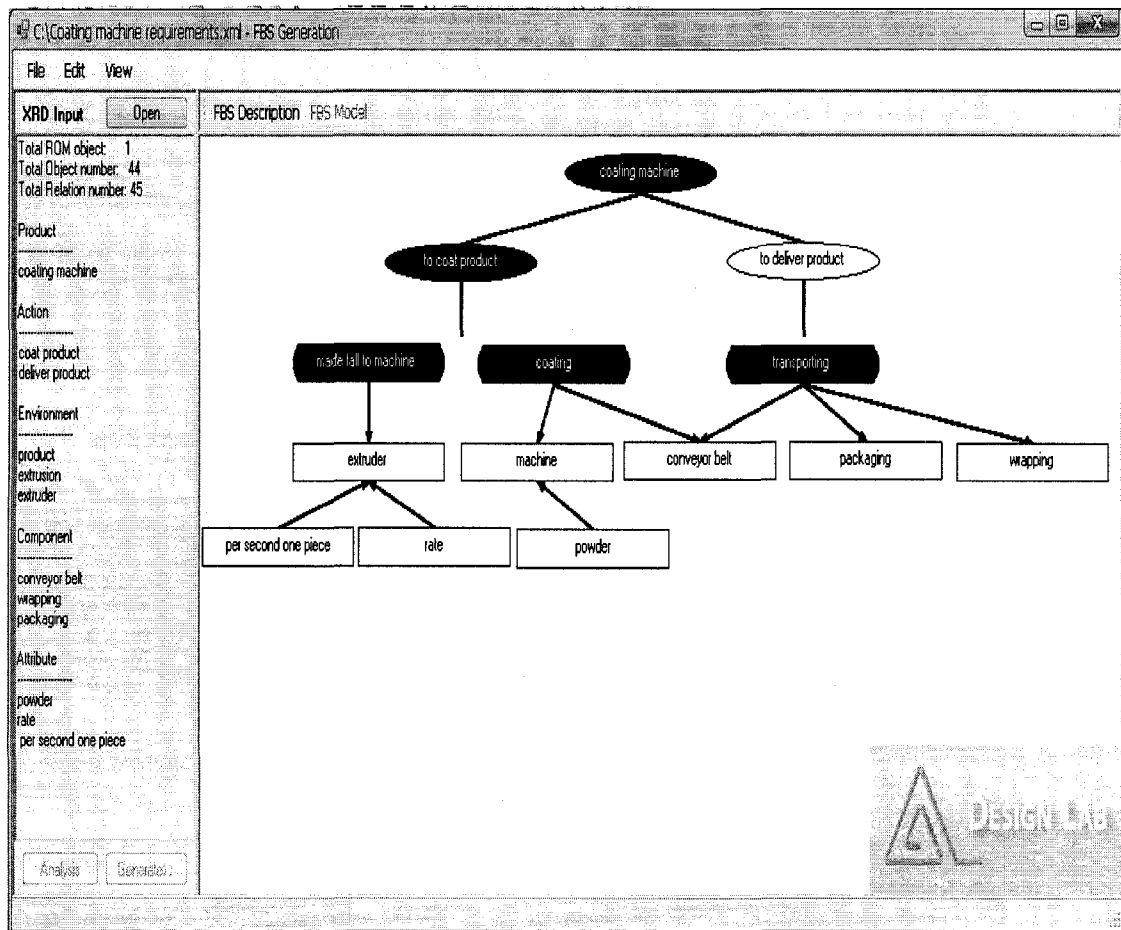


Figure 29 A FBS model output of R2FBS 1.0 for design requirements

5.3.2 Formalization of design patents

FBS models deal with design documents that may include many types of text such as the product requirements illustrated above or design patents, which will be discussed below.

We chose the design patents as a study case because patents contain more accurate and restricted text styles, making them different from requirement documents. Also, most design patents focus on describing the component and functionality of a product system. This focus is very suitable to our approach. The following design patent describes a low temperature clothes dryer based on a real United States Patent:

A low temperature clothes dryer having a drying chamber provides removable horizontal screens supporting clothing items and a hanging bar for hanging clothes to be dried. A timing control allows setting the time of operation of the drying cabinet. An electric heater with thermostat is provided to initially raise and maintain the air temperature within the drying chamber to at least about 90 degrees F. The dehumidifier is then operated, providing for circulation through the ducts and drying cabinet by an internal fan. The dehumidifier has an evaporator, through which warm, humid air is passed, thereby cooling the air and condensing water there from, the water being collected in a removable container or drained through a drain hose. The fan forces the cooled, dried air through a condenser which heats the dried air for recirculation through the drying chamber by means of ducts, thereby drying the clothing therein.

-From United States Patent, Patent n No.: US 7,377,052 B2; Date of Patent: May 27, 2008

The text in the design patent of this low temperature clothes dryer clearly describes the components of the dryer and functionality of each component. For the designer, this document is more accurate than a regular product design requirements text. So far as our

approach is concerned, the patent text is easier to extract information from after performing the linguistic analysis of our approach: ROMA. Figure 31 shows the results of the ROMA process: a ROM diagram. This ROM diagram is also the input of our approach to automatically generate a function representation schema and finally the formalized FBS models.

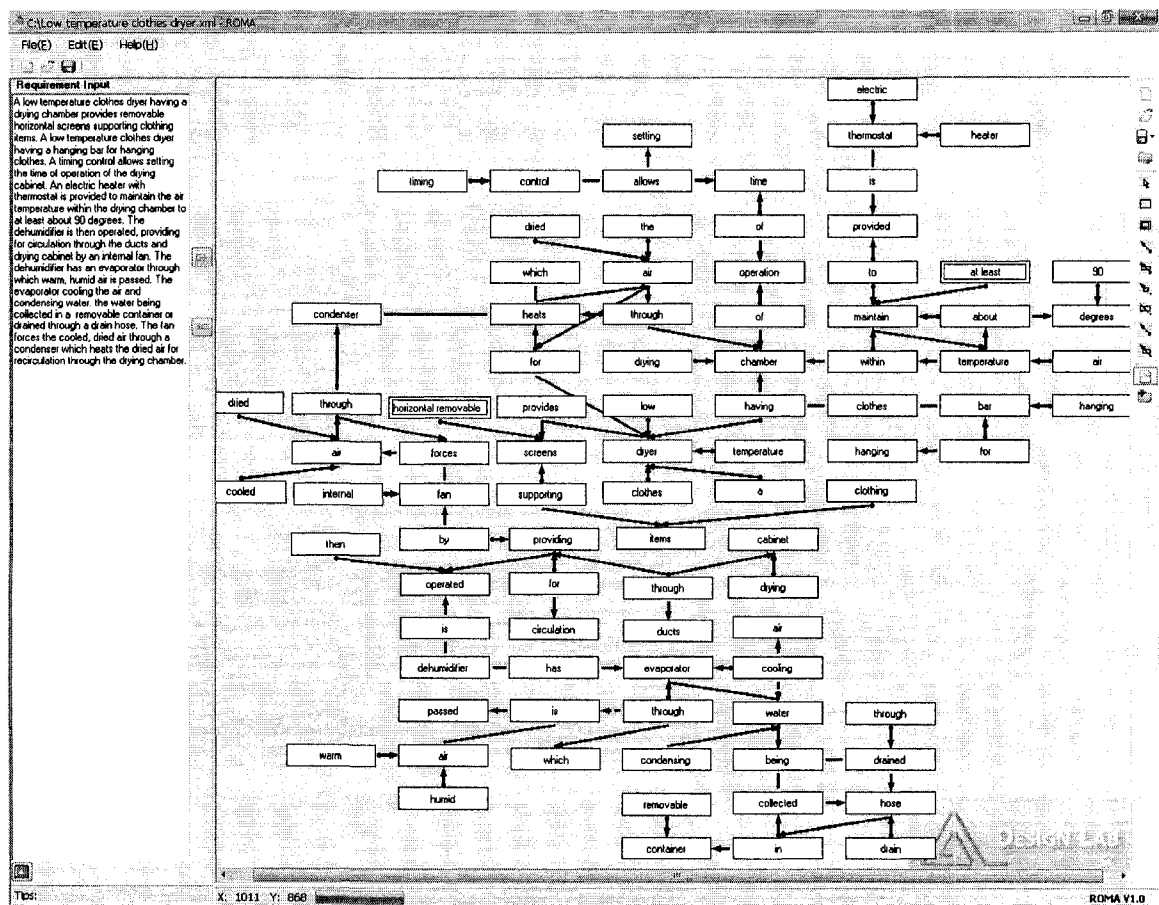


Figure 30 A ROM diagram for the low temperature clothes dryer

Unlike the previous test case, the design patent of this low temperature clothes dryer holds more information about functions and components because of the property of the patent text. Therefore, the function schema for the design patent has become more

80

complex compared with the previous example of design requirements. The schema generated by R2FBS is shown in Figure 32. Figure 33 shows the formalized FBS models based on the schema and traversal algorithms discussed above in this chapter.

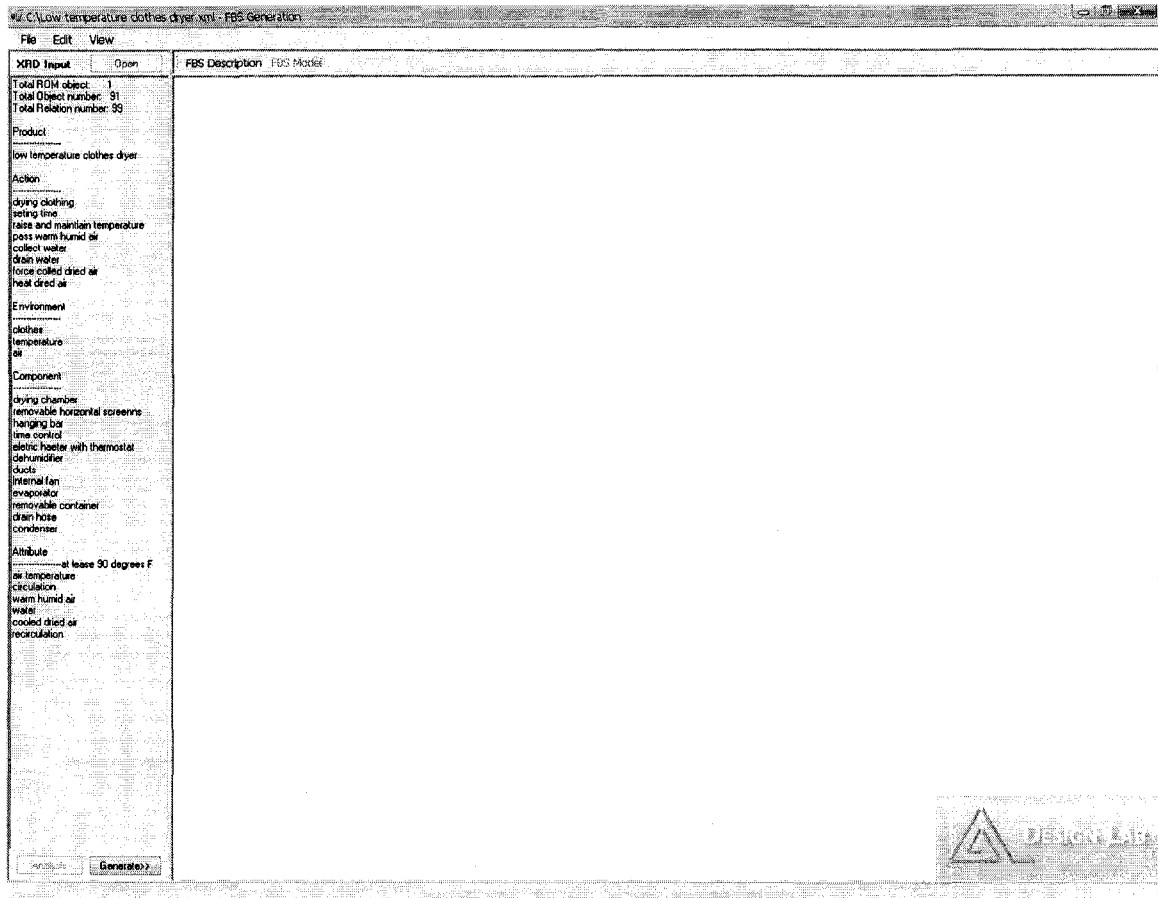


Figure 31 Function schema of the low temperature clothes dryer generated by R2FBS 1.0

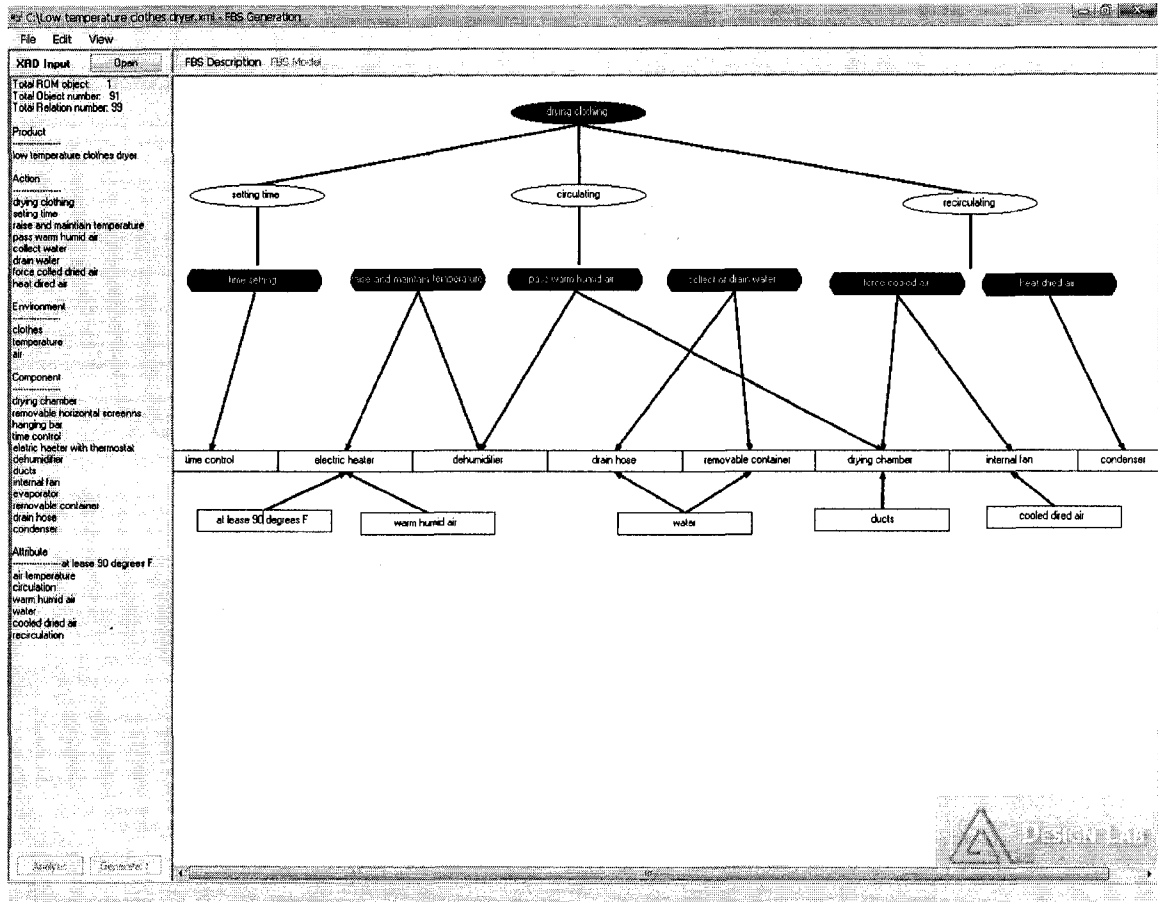


Figure 32 FBS models of low-temperature clothes dryer

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Design document modeling is important at the early stage of product design and in the engineering fields. A correct and complete design model is extremely useful for the engineer.

The present thesis proposes a novel approach in the automatic transformation of design documents from a natural language description to structural modeling languages. With the help of linguistic analysis and modeling specification languages, UML and FBS modeling, this approach generates UML diagrams and FBS models as two outputs.

The ROM diagram corresponding to a text includes the main semantic information implied in the text. Based on the semantics in the ROM diagram, a knowledge based proposal includes the criteria of function element and function decomposition rules. A software prototype is presented as a proof of concept for this approach. A case study shows that the proposed approach is feasible.

Though the automatic transformation of design documents shows a great potential in the product design of manufacturing engineering, our proposed approach does not intend to exclude the human users from the loop. On the contrary, this approach will help engineers better understand requirements, especially in a large project, by reducing the ambiguities of human understanding in analyzing the requirements and by increasing the consistency of the final function models when multiple engineers may be involved.

6.2 Future Work

As can be seen in the present thesis, our current approach largely depends on the capability and capacity of the ROMA system, which captures the semantics of a natural language text. Therefore, the accuracy of ROMA is of critical importance. Currently, ROMA is still under further development though it is already very robust. Another problem that needs to be dealt with is the study of the structure of design documents so that they can be preprocessed in terms of the ROMA system. The rules for the generation of UML diagrams and FBS models from a ROM diagram should also be refined. Lastly, it is important to perform a more comprehensive system test based on statistical analysis rather than on a limited number of case studies. These tasks are being implemented in a collaborative project on a PLM system for the automotive and aerospace industries.

Reference

1. IEEE, *IEEE Standard Glossary of Software Engineering Terminology*. 1990.
2. Wang, M. and Y. Zeng, *Asking the right questions to elicit product requirements*, in *Electrical and computer engineering*. 2007, Concordia University: Montreal. p. 90.
3. Tomiyama, Y.U.a.T., *FBS Modeling: Modeling Scheme of Function for Conceptual Design*. 1995.
4. Rumbaugh, J., I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. 1998: Addison-Wesley.
5. OMG. *OMG Systems Modeling Language, v1.0*. [cited; Available from: <http://www.omg.org/docs/formal/07-09-01.pdf>].
6. Zeng, Y., *Environment-based formulation of design problem*. Transaction of SDPS: Journal of Integrated Design and Process Science, 2004. **8**(4): p. 45-63.
7. Zeng, Y., *Recursive Object Model (ROM) - Modeling of Linguistic Information in Engineering Design*. Computers in Industry, 2008. **59**(6): p. 612-625.
8. Browne, G.F.S.a.G.J., *Conceptual foundations of design problem solving*. IEEE Transactions on Systems, Man, and Cybernetics, 1993. **23**: p. 1209-1219.
9. Zeng, Y., *Axiomatic theory of design modeling*. Transaction of SDPS: Journal of Integrated Design and Process Science, 2002. **6**(3): p. 1-28.
10. Zeng, Y. and G.D. Cheng, *On the logic of design*. Design Studies, 1991. **12**(3): p. 137-141.
11. Mala, G.S.A. and G.V. Uma, *Automatic construction of object oriented design models [UML diagrams] from natural language requirements specification*, in *Pricai 2006: Trends in Artificial Intelligence, Proceedings*. 2006. p. 1155-1159.
12. Nuseibeh, B. and S. Easterbrook. *Requirements engineering: a roadmap*. in *Proceedings of the Conference on The Future of Software Engineering 2000*. Limerick, Ireland ACM Press
13. Fantechi, A., et al., *Assisting requirement formalization by means of natural language translation*. Formal Methods in System Design, 1994. **4**: p. 243–263.
14. Gnesi, S., et al., *An Automatic Tool for the Analysis of Natural Language Requirements*. International Journal of Computer Systems Science and Engineering, 2005. **20**(1).

15. Miles Osborne, C.K.M., *Processing Natural Language Software Requirement Specifications*. icre, Second International Conference on Requirements Engineering (ICRE'96), 1996: p. 229.
16. Zeng, Y., *Recursive Object Model (ROM) - Modeling of Linguistic Information in Engineering Design*. Computers in Industry (submitted), 2007.
17. Xin, Z., Q. H. Mehdi and N. E. Gough, *From Visual Semantic Parameterization to Graphic Visualization*. Proceedings of the Ninth International Conference on Information Visualisation (IV'05), 2005.
18. Ma, M. and P. McKeivitt. *Semantic representaion of events in 3D animation*. in *Proceedings of The Fifth International Workshop on Computational Semantics (IWCS-5)*. 2003. Tilburg, The Netherlands.
19. Mikael R. Jensen, T.H.M., Torben Bach Pedersen, *Converting XML DTDs to UML diagrams for conceptual data integration*. Data & Knowledge Engineering, 2003. **44**: p. 323 - 346.
20. MacDonell, P.S.G., D.K. Min, and D.A.M. Connor, *AUTONOMOUS REQUIREMENTS SPECIFICATION PROCESSING USING NATURAL LANGUAGE PROCESSING*. Proceedings of the 14th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, 2005.
21. MICH, L., *NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA*. Natural Language Engineering, 1996. **2**(2): p. 161-187.
22. ROLLAND, C. and C. PROIX, *A NATURAL LANGUAGE APPROACH FOR REQUIREMENTS ENGINEERING*. Advanced information systems engineering, ed. P. Loucopoulos. 1992: Springer.
23. Omar, N., P. Hanna, and P. Mc Kevitt, *Heuristics-based entity-relationship modelling through natural language processing*. Fifteenth Irish Conference on Artificial Intelligence and Cognitive Science (AICS-04), 2004: p. 302-313.
24. Long, D. and R. Garigliano, *Reasoning by Analogy and Causality: Model and Applications*. 1994, Chichester, UK: Ellis Horwood.
25. Wang, M. and Y. Zeng, *Asking the right questions to elicit product requirements*. International Journal of Computer Integrated Manufacturing, 2007. **in press**.
26. Anderson, J.R., Byrne, M. D., Douglass, S., Lebiere, C., *An Integrated Theory of the Mind*. Psychological Review, 2004. **111**(4): p. 1036-1050.
27. Buzan, T., *The Mind Map Book, Chapter "Mind Mapping Guidelines"* 1991, New York: Penguin.

28. Chen, P.P., *The Entity-Relationship Model - Toward a Unified View of Data*. ACM Transactions on Database Systems 1976. **1**(1): p. 9-36.
29. Fowler, M., *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Third Edition ed. Object technology series. 2003: Addison-Wesley.
30. Lutz Maicher and Jack Park: , I.-.-.-. *Charting the Topic Maps Research and Applications Landscape*. 2006, Berlin: Springer.
31. Zeng, Y. *Formalization of design requirements*. in *Integrated Design and Process Technologies, IDPT-2003, Austin, Texas, December 3-6, 2003*. 2003.
32. Zeng, Y. and P. Gu. *An environment decomposition-based approach to design concept generation*. in *Proceedings of International Conference on Engineering Design'01*. 2001.
33. Zhu, S., S.J. Yao, and Y. Zeng. *A novel approach to quantifying designer's mental stress in the conceptual design process in ASME DETC/CIE 2007*. 2007. Las Vegas, Nevada, USA.
34. Zeng, Y. *Formalization of design requirements*. in *Integrated Design and Process Technologies, IDPT-2003, Austin, Texas, December 3-6*. 2003.
35. M.S. ERDEN, H.K., T.J. VAN BEEK, V. D'AMELIO, E. ECHAVARRIA, AND T. TOMIYAMA, *A review of function modeling: Approaches and applications*. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2007.
36. Zeng, Y., *Environment-based design: process model*. 2004, Concordia Institute for Information Systems Engineering, Concordia University: Montreal. p. 40.
37. Hochmüller, E., *Requirements Classification as a First Step to Grasp Quality Requirements*.
38. Américo Sampaio, N.L., Awais Rashid and Paul Rayso, *Mining Aspects in Requirements*.
39. Zeng, Y., *Axiomatic Approach to the Modeling of Product Conceptual Design Processes Using Set Theory*, in *Department of Mechanical and Manufacturing Engineering*. 2001, University of calgary: Calgary, Alberta, Canada. p. 229.
40. Zeng, Y. and P. Gu, *A science-based approach to product design theory Part II: Formulation of design requirements and products*. Robotics and Computer Integrated Manufacturing, 1999. **14**(4): p. 341-352.
41. Vladimir Hubka, M.M.A., W Ernst Eder, *Paractical Studies in Systematic Design*. 1988.