Requirements Modeling: from Natural Language to Conceptual Models Using Recursive Object Model (ROM) Analysis

Min Wang

A Thesis

In the Department

of

Electrical and Computer Engineering

Concordia University

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Electrical and Computer Engineering) at
Concordia University
Montreal, Quebec, Canada

July, 2013

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Ms. Min Wang

Entitled: "Requirements Modeling: from Natural Language to Conceptual Models Using ROM Analysis"

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

____________________________	Chair
Dr. Weiming Shen	External Examiner
Dr. Olga S. Ormandjieva	External Examiner
Dr. Abdelwahab Hamou-Lhadj	Examiner
Dr. Benjamin Fung	Examiner
Dr. Yong Zeng	Supervisor

Approved by ____

Chair of Department or Graduate Program Director

___________20___ ____

Dean of Faculty of Engineering & Computer Science

ABSTRACT

Requirements Modeling: from Natural Language to Conceptual Models Using Recursive Object Model (ROM) Analysis

Min Wang Ph.D.

Concordia University, 2013

Requirements elicitation and modeling are critical for the success of product development not only in software engineering but also in other engineering fields. Collecting the right requirements at each stage and transforming them into conceptual models are essential in delivering a successful product. In most cases, original requirements are represented by natural language in engineering. However, a key challenge faced by industries is to transform existing loosely structured legacy requirements document into the structured representations. This transformation process is extremely time-consuming and prone-to-error. Some efforts in research have been made to develop automatic or semi-automatic processes to bridge natural language and formal representation. Motivated by both the strong industrial need to automatically formalize natural language based requirements (NLR) and the research breakthrough in product requirements modeling, this present thesis proposes a new approach to transforming product requirements from their unrestricted natural language representation to structured conceptual models by using Recursive Object Model (ROM).

The proposed approach includes the following three main aspects: 1) developing criteria for the completeness and necessity of design requirements corresponding to certain design stage, 2) developing a dynamic requirements elicitation approach to refine requirements, and 3) developing algorithms for transforming design requirements from natural language to conceptual models, such as Use Case Model by Universal Modeling Language (UML) and Function-Behavior-State (FBS) model. This presented research involves Natural Language Processing (NLP) techniques, in conjunction with question asking (QA) strategy and conceptual modeling algorithms. The significant tasks include defining the scope of the right requirements, automatically question asking to elicit requirements, formulating the transformation of requirements text into conceptual models, generating the rules for the conceptual modeling, developing algorithms based on the transformation rules, and finally automating the requirements modeling process through software prototypes.

The research foundation of this thesis is the Environment Based Design (EBD) methodology which is derived from axiomatic theory of design modeling (ATDM). To bridge the gap between unrestricted natural language and formal conceptual models, an intermediate representation, ROM, is the core for representing the semantics of design requirements throughout the requirements evolution process.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Y. Zeng for providing me with the opportunity to do this challenging and rewarding research and for his invaluable guidance, advice, encouragement and financial support throughout my PhD study. His passion and dedication in the pursuit of the truth in research, his insightful sense of the best research directions, his logical and creative thinking about research and life have influenced every aspect of my research and life. I would also like to thank my thesis committee members, Dr. O. Ormandjieva, Dr. W. Hamou-Lhadj, Dr. B. Fung, and Dr. W. Shen for their constructive suggestions and comments on my research.

I would like to extend my thanks to my colleagues Mr. Lei Chen, Mr. Suo Tan, Ms. Thanh An Nguyen, Ms. Yijing Zeng, Dr. Xiaoguang Deng, Dr. Wei Liu and other members in the Design Lab for their comments on my research and the friendship throughout the past years. I would also express my appreciation to the professors and staffs in ECE and CIISE of Concordia University, who have provided a wonderful academic surroundings for me to complete this study.

Last but not least, I wish to express my appreciation to my beloved family. I am very grateful towards my parents, my brother, and my sister. I owe a lot to my husband, Wei, for his extraordinary patience, love, and support over so many years. My son Eric always brings joyfulness and hope for my life. I cannot imagine being able to reach this point without their love.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

EBD: Environment-Based Design

ROM: Recursive Object Model

PES: Product-Environment System

UML: Universal Modeling Language

OMG SysML: OMG Systems Modeling Language

UCD: Use Case Diagram

DM: Domain Model

FBS: Function-Behavior-State

QA: Question Asking

R2UML: ROM to UML

R2FBS: ROM to FBS

NLR: Natural Language based Requirements

Chapter 1
Introduction

1.1 Background and Motivation

Requirements are a foundational aspect of system engineering, software engineering, or enterprise engineering. Along with the increasing scale of product development, the significance and effects of requirements are becoming more outstanding. They play a driving role and provide all the necessary information to implement product development or process optimization on time within budget. The requirement quality is the key for the success of the whole product lifecycle. It is imperative that requirements must be gathered and extracted into necessary, specific, complete dimension, and understood by all stakeholders of development project. Requirements can be described in a natural language, sketches, equations, and some other forms like multimedia. They are the starting point and basis for the design phase. Design information can be contained in various representations, such as text, verbal statements, graphic models and mathematical expressions. Among all the representations, natural language is the most flexible yet ambiguous means even it is the most widely used; graphic models are the more effective and the more efficient; mathematical language is the most precise.

Along with the advancement of computing technologies, more and more design tasks are directly or indirectly aided by computers. Directly, some design tasks are automated such as geometric modeling, structural analysis and optimization. Indirectly, some design tasks are being conducted through collaboration between human and computers such as drafting, innovation, and requirements elicitation. In order to support the entire design process, emerging CAD/E systems must be able to support the smooth integration of systems with other systems and with their human users. The basis of this integration is a semantic model that can accommodate the communication of systems with each other and with their human users (Luh et al., 2012).

Technologies supporting this communication range from early efforts on geometric reasoning (Marefat and Kashyap, 1990; Wilson and Latombe, 1994) to recent progresses on data/knowledge mining (Fayyad et al., 1996; Lin and Katz, 2003). Another application in CAD/E closely related to semantics is the acquisition of design knowledge from existing design requirements. It is useful to extract critical design information relevant from the design requirements to a new design or redesign. Critical information can only be identified if the semantics of the requirements is understood. It must also be pointed out that a design requirement by itself and in its textual form is only a piece of passive knowledge. Its application depends on how the designer understands and digests the active design knowledge implied in the document so that they can be logically associated with a design situation. By definition, the understanding of requirements is the transformation from requirements into a formal representation constituted by a set of semantic components and their relationships (Wiggins and McTighe, 2005), which are conceptual models actually. Therefore, the transformation from requirements into a formal or semi-formal representation is necessary and essential for design, industry, and business. This representation provides a shared view of the product throughout the entire design process, enabling a design team to detect issues early and prevent problems that would otherwise delay development and degrade design quality.

For representing semantic information appearing in the design process, some formal and structured conceptual models are developed and applied in different fields. For example, in software engineering, Universal Modeling Language (UML) (OMG-UML, 2011) is a widely adopted software modeling notation to specify, construct and document the artifacts of systems. A few semantic approaches have been developed to process class diagrams (Chen and Zeng, 2009), state machines (Lano and Clark, 2007), interactions (Lano, 2007), use cases (Chen and Zeng, 2009), OCL (Markovic and Barbosa, 2008) and activity diagrams (Storrle and Hausmann, 2005). However, UML is a software-specific language, which does not support the general needs of design in other domains (Wölkl and Shea, 2009). SysML is used to specify, analyze, and design systems that may include hardware, software, and personnel (Friedenthal et al., 2008; OMG-SysML, 2011; Soares and Vrancken, 2008; Weilkiens, 2007). Since SysML is based on UML, it also facilitates integration between systems and software development.

In the design of engineering systems, especially mechanical and architectural systems, function is recognized as the bridge between human desires and physical behavior of artifacts. Among various function-based models (Deng, 2002; Erden et al., 2008; Gero and Kannengiesser, 2007; Goel et al., 2009; Umeda and Tomiyama, 1995), the Function-Behaviour-State (FBS) model was proposed by Umeda and Tomiyama as a framework to represent a design object hierarchically and to define a function as an association of human intention and behaviour (Umeda and Tomiyama, 1995). The FBS model has drawn a lot of attention in design research as it provides a knowledge representational scheme for conceptual design, and for the knowledge intensive engineering framework (Yoshioka et al., 2004).

A lot of efforts in research have been made to develop automatic or semi-automatic transformation from natural language to conceptual models. Tjoa and Berger proposed an approach to transforming natural language based requirements specifications into an EER model (Tjoa and Berger, 1993). Mala and Uma present an approach to extracting the object-oriented elements of the required system (Mala and Uma, 2006). Gnesi et. al. developed an automatic tool for the analysis of natural language requirements (Gnesi et al., 2005). Liu et al. proposed a methodology with Use-case language schemas to automate natural language requirements analysis and class model generation based on the Rational Unified Process (RUP) (Liu et al., 2004). Due to the difficulties in natural language processing (Kanda et al., 2008) and the huge gap between natural language and structured models (Fantechi et al., 1994; Gnesi et al., 2005; Osborne and MacNish, 1996), those efforts have achieved very limited success.

This thesis attempt to deal with the gap between unrestricted natural language based requirements and structured conceptual model through an intermediate model - Recursive Object Model (ROM) (Zeng, 2008), which captures the semantic information of the concerned natural language. Through study and practice in engineering, the ROM based transformation can help to extract system dynamics during the earlier design stage (Medyna et al., 2012) and facilitate the general modelling process (Ozaydin and Tanik, 2011) and specific design methods such as TRIZ (Cascini, 2012). The proposed approach first generates the ROM diagram of product requirements. Then the key elements included in the requirement text, and all the design information such as product components, product environment, and relations between them are extracted based on predefined rules. Finally, the key elements are transformed into a conceptual model.

1.2 **Objectives**

In most cases, customers describe their intent using natural language. For various reasons, customers may not be able to describe their needs accurately. The scope of the research presented in this thesis is limited in the initial requirements described in natural language. This research aims to present new approaches to developing requirements and transforming requirements into conceptual models in a systematic manner. Developing requirements means talking to all stakeholders involved to produce a robust set of requirements. Among the characteristics of high quality requirements, necessity and completeness are firstly needed for the success of a product development. Without them, you may end up spending more money to fix a problem or pay for costly customizing because you didn't identify all the components at the beginning. Therefore, requirements elicitation through clarifying and explicating the initial requirements to refined requirements is necessary preparation for this aim.

On the other hand, the fierce market competition enforces industries to keep improving and innovating their products. Catching market trends is the art of requirement collection and analysis. Environment-Based Design (EBD) methodology (Zeng, 2004a; Zeng, 2004b; Zeng, 2011) provides us with a feasible and systematic solution. The environment analysis approach in EBD can help us overview the requirements from a wider perspective, collecting and analyzing them more systematically and all-round. Moreover, proved by practice, EBD guides both routine and creative design naturally not only for engineering but also for personal development, even for research.

Motivated by EBD, some questions should be asked for effective eliciting necessary and complete requirements. The first question is what are the criteria of necessary and complete requirements? Criteria are needed to evaluate the requirements, at the same time; the criteria can direct a roadmap to collect the requirements. The second question is how to get necessary and complete requirements in a more systematic manner? However, natural language may easily lead to ambiguous or distorted understanding of the user's original intents (Oxman, 2004). Moreover, within most product development frameworks requirement generations are some of the more ill-defined and least structured activities (Arthur and Gröner, 2005). Therefore, the use of more precise NLP system or linguistic tools will help to support the product development in general and requirements analysis in particular.

The second aim of this research is transforming natural language based requirements into conceptual models, during which semantic information needs to be analyzed, extracted and formulated through transformation mechanism. In the same way, the NLP system will be utilized in semantic analysis. In addition, an automatic and structured framework will help to reduce the misconception and improve the efficiency and quality.

To achieve above aims, we propose the research framework shown in Figure 1-1. This framework illustrates that the transformation process from initial requirements to conceptual models consists of two steps: 1) clarification and/or explication from initial requirements to refined requirements, and 2) transformation from refined requirements to conceptual models through extracting the semantic meaning of the requirements by performing a series of systematic procedures. The requirements refinement process is not necessarily linear in nature, but an iterative process that constantly refines existing requirements and identifies new ones, which matches the whole design activities iteration process accordingly.

Based on our research, an automatic question-asking strategy based on proper requirements roadmap is effective and feasible in the communication process for collecting and refining requirements purpose. The research foundation of requirements roadmap and Question-Asking strategy is the Environment Based Design (EBD) methodology while Recursive Object Model (ROM) has been found to be a valuable tool for representing the semantics of design requirements in both question asking and semantic analysis.

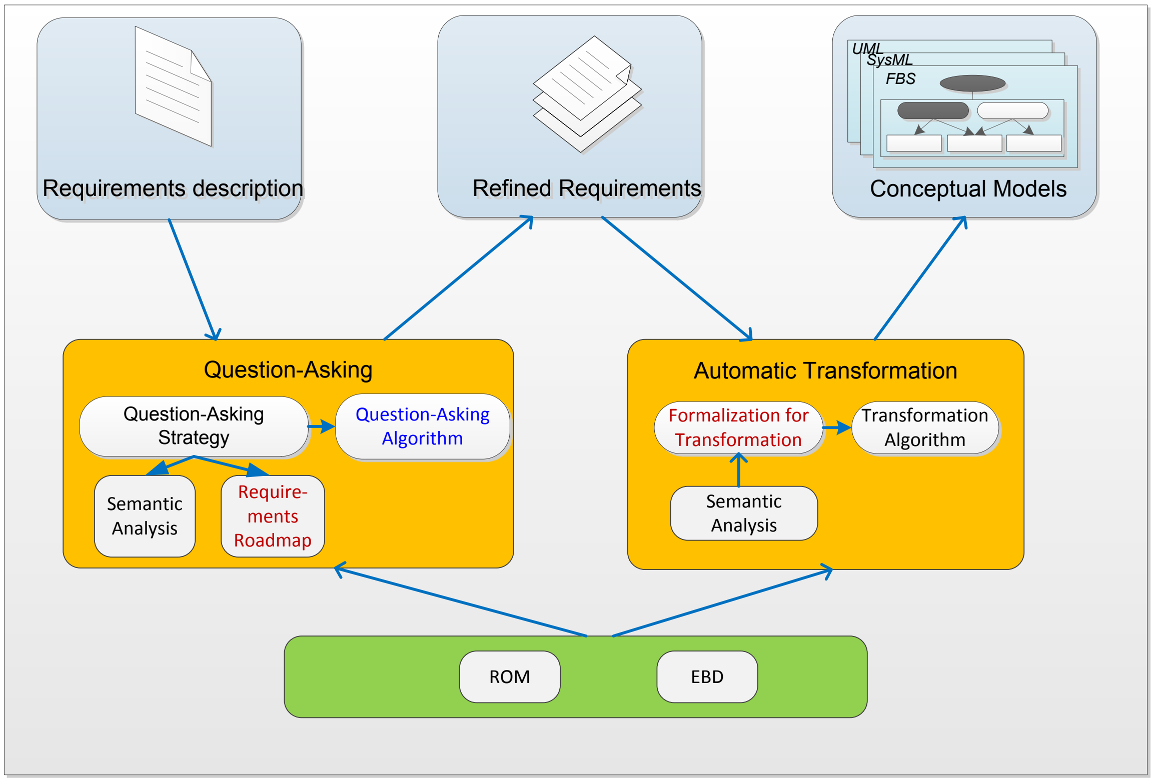


Figure 1-1 Research framework

Overall, the objectives of this research are extracted as follows based on the framework:

- Development of criteria/roadmap for the completeness and necessity of requirements based on EBD,
- Development of dynamic requirements elicitation strategy based on semantics and requirements roadmap,
- Formalization of the transformation from natural language based design requirements to conceptual models,

- Development of transformation algorithm from ROM to conceptual models (such as Domain Model and FBS) and prototypes, and
- Validation of proposed approach by case studies.

1.3 **Challenges**

Natural language forms a majority of design requirements, which reflects customer's needs. The difficulties in requirements elicitation and modeling lie in the understanding of such natural language based requirements accurately and thoroughly and capturing of semantics implied in an interested text and the identification of missing information. Great challenges still exist in this attempt, among which is the capturing of semantics from product design requirements. An extreme in this front is the processing of existing unrestricted natural language and transforming into the structured representations. Furthermore, design requirements for a complex product or process may include a great amount of information, which is extremely tedious for human processing. This transformation process is extremely time-consuming and prone-to-error. Therefore, a systematic and computer assisted requirements conceptual modeling is on demand definitely.

1.4 **Research Contributions**

The research in this thesis has made the following major contributions:

1) An environment-based requirements roadmap is proposed to direct requirements elicitation.

2) Question asking strategy is developed to effectively elicit complete requirements,

3) A general framework is proposed for formulizing the transformation from design requirements into conceptual models (UML, FBS) ,

4) Algorithms, including transformation rules and procedures, are developed for transforming design requirements into conceptual FBS, Use Case Model and Domain Model,

5) Software prototypes of Question Asking, R2UML and R2FBS are developed to implement the algorithms.

6) Three case studies from different engineering fields have been performed to demonstrate how the proposed algorithms work.

1.5 **Thesis Structure**

Chapter 1 presents the motivation, objectives, significance and overview of the present thesis.

Chapter 2 examines the previous research dealing with the requirements classification, requirements elicitation and requirements modeling from the fields of design science and computer science.

Chapter 3 introduces the theoretical foundations of this thesis, including the concepts of design thinking, design, Environment-Based Design (EBD) and Recursive Object Modeling (ROM).

Chapter 4 elaborates an environment-based roadmap for developing complete requirements, and dynamic requirement elicitation based on the proposed roadmap and question asking strategy. An example of energy trading system is chosen to illustrate the approach.

Chapter 5 analyzes the structure of conceptual models, such as FBS model and domain model.

Chapter 6 presents the formalization for transforming requirements into conceptual models.

Chapter 7 presents the algorithm of transformation from requirements text into FBS.

Chapter 8 presents the algorithm of transformation from requirements text into Use Case Model and Domain Model.

Chapter 9 evaluates proposed transformation from design text to FBS, Use Case Model, and Domain Model respectively through three case studies of design patent, energy trading system requirements, and POS system requirements.

Chapter 10 summarizes the research work and gives suggestions for future work.

Chapter 2
Literature Review

The objective of the present thesis is to develop the creteira for complete requirements and elicit and model the product requirements, which will facilitate design process and enhance the quality of design. To achieve this objective, this literature review will cover the following areas:

- Design and design process which provide the context for the present research,
- Product requirements and classification,
- Requirements elicitation methods,
- Requirements modeling,
- Conceptual models including UML, SysML and FBS, and
- Linguistic analysis for requirements

2.1 **Design and Design Process**

Researchers have provided various descriptions of the term "design" such as: design activities are generally considered to be a form of complex problem solving (Simon, 1969); design begins with a needs-analysis (Asimow, 1962); design is a social activity (Minneman, 1991). In some design studies, the objectives usually focus on finding common characteristics from different engineering domains, within the framework of cognitive science (Prabhakar and Goel, 1998). Therefore, design process can be regarded as a cognitive process intended to produce a solution to a design task. As a matter of fact, the design process varies from product to product and from industry to industry. In a generic framework, any product must go through five phases: project definition, specification definition, conceptual design, product development and product support (Ullman, 2002).

The nature of design requirements and the design process have been the subject of a wide variety of research. Recently, some approaches have been proposed in this field, such as methodology-based design and language-based design (Darlington and Culley, 2002). In the category of methodology-based design, some methods for the development of design support mechanisms have been applied, such as quality function deployment (QFD) (Akao and Glenn, 2003; Clausing, 1998), a taxonomic approach (Gershenson and Stauffer, 1999), key characteristics (Verstijnen et al., 1998), and functional decomposition. Darlington and Culley classified the research in this category into two kinds of noticeable design theories (Darlington and Culley, 2002). One comes from Wootton who made an analysis of the design requirement process in terms of the stakeholders and information sources involved in the complexity of developing new products as corporate activity (Wootton et al., 1998). The theory provides the foundation of a prescriptive guide to the process of requirement capturing for industrial use. The other one, a science-based approach to product design theory, comes from Zeng and Gu (Zeng and Gu, 1999). They proposed a set theory-based representation scheme for the representation of the design objects that evolve during the design process. As the continuation of the efforts in the science-based approach to product design theory, Zeng proposed a new design methodology, environment based design (EBD), which is a step-by-step approach to solving a poorly defined problem and which can assist the designers in delivering creative and innovative design solutions (Zeng et al., 2004).

In recent years, many researchers study, compare, and apply the EBD theory as well as applications in their research. For example, Maletz applies the formalization process of product requirements into an integrated requirements modeling approach in his PhD research, which is a contribution towards the integration of requirements into a holistic product lifecycle management strategy (Maletz, 2008). Weissman et al. adapt the concepts from EBD such as defining a product in terms of the elements of its environment, the use of requirement categories based on the product's life stages, and mapping natural language to a standardized representation, for their computational framework for authoring and searching product design specifications (Weissman et al., 2011). Another research in software engineering is Moroz's thesis. As the project manager of a software company, Moroz leads his group applying the EBD of Software (EBD-S) into agile software development by providing a light-weight and flexible framework for the architecture and design documentation, formalized design concept generation and effective system evolution control. This integration of EBD-S to the real-world Scrum development process is demonstrated on the example of Telecom Expense Management software development. Based on their work, Moroz concludes that the EBD-S approach resulted in 25% project time saving due to more accurate estimations, higher code quality and lower error rate (Moroz, 2011). All these research work testifies the EBD theory and approach are feasible and promising in theory and practice from different engineering perspective.

2.2 Requirements Engineering

In engineering, a requirement is a singular documented need of what a particular product or service should be or perform. It is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system in order for it to have value and utility to a user (Wikipedia, 2011). Requirements are most commonly used in systems engineering, software engineering, or enterprise engineering. Developing requirements means talking to all stakeholders involved to produce a robust set of requirements.

A requirement needs to meet several criteria to be considered a "good requirement" (Hull et al., 2005; Leffingwell and Widrig., 2003; Young, 2001; Zielczynski, 2007). The following are characteristics of a Good Requirement:

- Unambiguous: there should be only one way to interpret the requirement.
- Testable (verifiable): testers should be able to verify whether the requirement is implemented correctly. To be testable, requirements should be clear, precise, and unambiguous. Some words can make a requirement untestable (Hull et al., 2005):
    - Some adjectives: robust, safe, accurate, effective, efficient, expandable, flexible, maintainable, reliable, user-friendly, adequate
    - Some adverbs and adverbial phrases: quickly, safely, in a timely manner
    - Nonspecific words or acronyms: etc., and/or, TBD

    - Modifying phrases: as appropriate, as required, if necessary, shall be considered
    - Vague words: manage, handle
    - Passive voice: the subject of the sentence receives the action of the verb rather than performing it
    - Indefinite pronouns: few, many, most, much, several, any, anybody, anything, some, somebody, someone, etc.

- Clear (concise, terse, simple, precise): requirements should not contain unnecessary verbiage or information. They should be stated clearly and simply.

- Correct: If a requirement contains facts, these facts should be true.

- Understandable: requirements should be grammatically correct and written in a consistent style. Standard conventions should be used. The word “shall” should be used instead of “will,” “must,” or “may.”

- Feasible (realistic, possible): the requirement should be doable within existing constraints such as time, money, and available resources.

- Independent: to understand the requirement, there should not be a need to know any other requirement.

- Atomic: the requirement should contain a single traceable element.

- Necessary: a requirement is unnecessary if none of the stakeholders needs the requirement, or removing the requirement will not affect the system.
- Implementation-free (abstract): Requirements should not contain unnecessary design and implementation information.

Besides these criteria for individual requirements, three criteria apply to the set of requirements. The set should be

- Consistent: There should not be any conflicts between the requirements.
- Nonredundant: Each requirement should be expressed only once and should not overlap with another requirement.
- Complete: A requirement should be specified for all conditions that can occur.

In software engineering, it is widely recognized that requirements can be classified into: functional and nonfunctional requirements; nonfunctional requirements are classified further as performance/reliability, interfaces and design constraints (Southwell et al., 1987). In security requirements engineering (SRE), security requirements are split from function and nonfunctional requirements with more detailed classification (Fabian et al., 2010). Cleland-Huang et al. proposed an automated classification of non-functional requirements (Cleland-Huang et al., 2007). Casamayor et al. proposed a semi-supervised classification of non-functional requirements (Casamayor et al., 2009). In design lab of Concordia University, Chen and Zeng gave a useful classification of requirements in generic engineering point of view, shown in Figure 2-1 , the product requirements are categorized into eight levels according to the environments the product resides in: natural laws, social law and regulations, technical limitation, cost, time and human resource, basic functions, extended functions, exception control level, and human-machine interface (Chen and Zeng, 2006). In this model, the priority is determined that the requirements at the lower levels have higher priority in developing a design solution. Other classification framework for software requirements prioritization approaches are proposed on emphasizing differences and similarities among eleven selected approaches (Carod and Cechich, 2009).

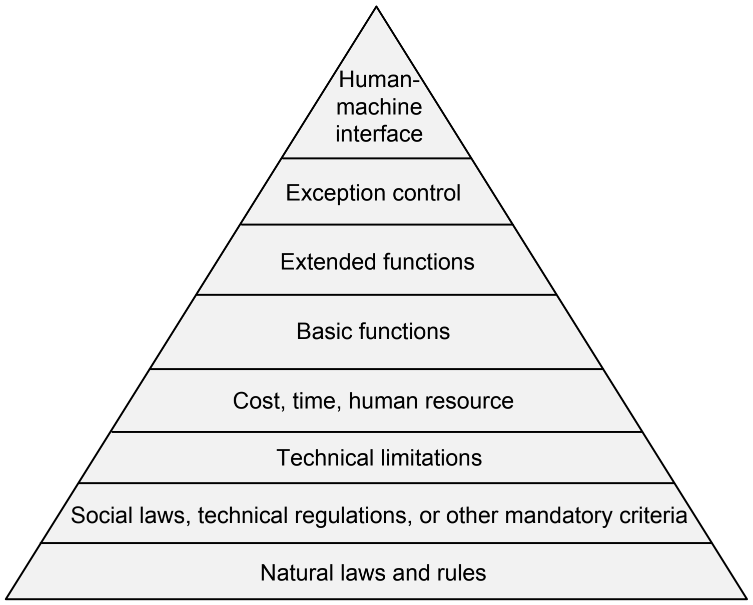


Figure 2-1 Eight levels of requirements (Chen and Zeng, 2006)

From practice, research and business analysis points of view, VOLERE (Volere, 2010), a famous requirements specification template, is widely used by organizations for discovering, organizing, and communicating their requirements. VOLERE classifies requirements into functional requirements, non-functional requirements, project constraints, design constraints, project drivers, and project issues. This template with detailed further classifications in each of category provides comprehensive support for understanding a product to be designed. Besides, this template can be used with some popular tools, such as DOORS (IBM, 2011a), Requisite (IBM, 2011b), Caliber RM (Borland, 2011), etc. In practice, different requirements classification may lead to different design method (Amyot, 2003).

Requirements engineering (RE) is the systematic approach of developing requirements through an iterative cooperative process of analyzing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained (Loucopoulos and Karakostas, 1995).

The software systems RE is constituted by five core activities of eliciting requirements, modeling and analyzing requirements, communicating requirements, agreeing requirements, and evolving requirements. In practice as other design process, these core activities are interleaved, iterative, and may span the entire software system development life cycle (Nuseibeh and Easterbrook, 2000). Many methods and approaches have been proposed and applied, also endlessly research efforts are being conducted in these activities in requirements engineering.

2.3 **Requirements Elicitation**

Requirements elicitation is the first step and critical activity in the early phases of requirements engineering. It is a process of interactions between customers, designers, project managers, and other partners of the product development. The aims of requirements elicitation are set up to identify the system boundaries, stakeholders, business goals, technical goals, and tasks in a project. It is reported that more bugs occur in requirements specification than in coding (56% vs. 7%) and furthermore bugs in requirements specification are more expensive to correct (82% vs. 1%) (Martin, 1987). Macaulay identified five possible causes of system failures, presented below in descending order of effect: 1) Poor communication between people; 2) Lack of appropriate knowledge or shared understanding; 3) Inappropriate, incomplete or inaccurate documentation; 4) Lack of a systematic process; and 5) Poor management of people or resources (Macaulay, 1996). To deal with these problems, a framework of methodological approaches to requirements elicitation is proposed with four-dimension view: user participation and selection, user-designer interaction, communication activities, and techniques (Coughlan and Macredie, 2002). Another important issue is due to the iterative nature of design, requirements will evolve and change (Morkos et al., 2012). A good requirements elicitation method should predict the requirement change propagation.

Accordingly, it can be seen that effective communication and accurate statements of requirements are the key factors in the design of successful systems. However, achieving a shared understanding of requirements is difficult in any situation, obtaining the right requirements therefore implies efforts in software development process (Aranda et al., 2010). Natural language is usually a major means of communication during the elicitation process (Lecoeuche et al., 1998). Natural language allows design requirements to be discussed with enormous semantic richness easily and naturally by non-specialists. However, natural language descriptions carry lots of noises, ambiguities, and contradictions, as pointed out by Meyer (Meyer, 1985). Therefore, requirements elicitation has to deal with informality, incompleteness and inconsistency (Leite and Cesar, 1987).

Collecting information is the basic approach in requirements elicitation by in-site observation, formal interviews, informal discussions, questionnaires, and history utilization (Andreou, 2003). Abundant elicitation techniques are presented and adopted in numerous projects. Traditional techniques include questionnaires, surveys, interviews, analysis of existing documentation such as organizational charts, process models or standards, and user or other manuals of existing systems. Group elicitation techniques have brainstorming and focus groups, as well as RAD/JAD workshops. Others classes techniques like prototyping techniques, model-driven techniques, cognitive techniques, and contextual techniques may be integrated in use with traditional and groups techniques (Sajid et al., 2010). We summarized some major methods as follows:

- Interviews: interviews are the most common technique used for gathering information during requirements elicitation as other traditional methods. However there are no standardized procedures for structuring information received from interviews (Zeroual, 1989). It is also challenging to integrate different interpretations, goals, objectives, communication styles, and use of terminology into a single set of requirements (Hickey and Davis, 2004).

- Issue-based information system (IBIS) (Christel and Kang, 1992; Conklin et al., 1991): IBIS provides an integrated approach to organizing information from interviews, though it does not support automated checking of consistency, nor support for types outside of issues, positions, and arguments.
- Joint application design (JAD): JAD, a team technique, focuses on improving the group process and getting the right people involved from the beginning (Zahniser, 1990). It promotes the cooperation, understanding, and teamwork. Meanwhile, JAD enhances idea generation and evaluation, communication, and consensus generation. JAD is specifically designed for the development of large computer systems and it has been used successfully by IBM since the late 1970s (Wood and Silver, 1995).
- Misuse cases: Misuse cases apply the concept of a negative scenario in a use-case context. One significant characteristic of misuse cases is that they seem to lead to quality requirements, such as those for safety and security, whereas other elicitation methods are focused on end-user requirements (Alexander, 2003).

In order to promote understanding and gathering of information in elicitation, many elicitation approaches represent the requirements from different viewpoints such as:

- Controlled requirements expression (CORE): CORE provides a framework for analyzing and expressing requirements in a structured diagrammatic notation (Christel and Kang, 1992; Mullery, 1979). However, CORE does not effectively represent timing behavior and reuse; the support of complex data descriptions remains to be a problem.

- Feature-oriented domain analysis (FODA) (Kang et al., 1990): FODA is a domain analysis method that focuses on developing reusable assets. The FODA method, founded on two modeling concepts: abstraction and refinement (Kean, 1997), abstracts different applications to the level where no differences exist between the applications. Specific applications in the domain are developed as refinements of the domain products.
- Critical discourse analysis (CDA) (Schiffrin, 1994): CDA uses sociolinguistic methods to analyze verbal and written discourse. Sociolinguistics assigns special significance to the structure of speech and texts; it also provides methods for specifying the linguistic features of different types of discourse units and the way they are tied together into larger units of meaning (Alvarez, 2002). In particular, CDA can be used to analyze interviews from requirements elicitation and to understand the narratives and "stories" that emerge during the interviews.
- Accelerated Requirements Method (ARM) (Hubbard et al., 2000): The ARM process is a facilitated requirements elicitation and description activity. Overall, there are three phases of the process: preparation phase, facilitated session phase and deliverable closure phase. During the preparation phase, planning and preparation are completed to ensure an effective session. During the session phase, a trained--and content neutral-

-facilitator leads the selected participants through a structured process to collect the functional requirements of the project under consideration. And in the closure phase, the key deliverables, such as a requirements collection, are polished.

- Quality Function Deployment QFD: QFD is "an overall concept that provides a means of translating customer requirements into the appropriate technical requirements for each stage of product development and production (QFD-Institute, 2005). The distinguishing attribute of QFD is the focus on customer needs throughout all product development activities. By using QFD, organizations can promote teamwork, prioritize action items, define clear objectives, and reduce development time (QFD-Institute, 2005). Although QFD covers a broad portion of the product development life cycle, the earlier stages of the QFD process are applicable to requirements elicitation for software engineering (Mead, 2006). These stages include: 1) identifying the customer (stakeholders), 2) gathering high-level customer requirements, 3) constructing a set of system features that can satisfy customer needs, and 4) creating a matrix to evaluate system features against satisfaction of customer needs.

As we observe, each technique has trade-offs between strength and weakness. In practice, proper one or more techniques can be applied according to the type and volume of a project. Among those techniques, question-asking approach no matter in questionnaires, surveys, electronic interviews, face-to-face interviews or others may be adopted widely. Though a lot of efforts have been made to address the problems in requirements elicitation, not much research results have been reported regarding the approaches based on questioning and answering. Hands et al. propose a computer-based interviewing tool which may enhance the requirements gathering process and conducting user evaluation, however it was executed by the predefined questions (Hands et al., 2004). Another investigation was Eris's work on the role of effective inquiry in the innovative engineering design process (Eris, 2004). But his work falls short of a methodology on how to make effective inquiries. Tom and Sitte presented a formal approach named Requirements Elicitation of Future Users by Systems Scenarios (REFUSS) to derive future user requirements (Tom and Sitte, 2009). Wang and Zeng proposed a systematic iterative question-asking approach to elicit product requirements (Wang and Zeng, 2009). This approach aims at identifying the customer’s real intent and at capturing the complete product requirements by asking questions based on a semantic analysis of the requirements text, which is represented by ROM diagrams. The question asking approach is feasible and promising by the initial experiments. However, the algorithms and generation rules for question asking need to be improved before it can be put into industrial applications.

2.4 **Requirements Modeling**

Requirements modeling is a fundamental activity in RE and it is the construction of abstract descriptions that are amenable to interpretation. Models can be used to represent a whole range of products of the RE process.

Some general categories of RE modeling approaches are described below:

- Enterprise modeling: Enterprise modeling and analysis deals with understanding and organization’s structure; the business rules, the goals, tasks and responsibilities of its constituent members, and the data that it needs, generates and manipulates. Enterprise modeling is often used to capture the purpose of a system by describing the behavior of the organization in which that system will operate (Loucopoulos and Kavakli, 1995), or model an enterprise in terms of its business rules, workflows and the services that it will provide (Greenspan and Feblowitz, 1993).
- Data modeling: Data modeling is used in large computer-based systems, especially information systems to understand, manipulate and manage information data. Traditionally, Entity-Relationship-Attribute (ERA) (Johnson and Henderson, 2011) modeling is used for data modeling and analysis; Nowadays, Object-Oriented modeling is increasingly supplanting ERA techniques by using class and object hierarchies.
- Behavioral modeling: Modeling the dynamic or functional behavior of stakeholders and systems, both existing and required, is often involved in modeling requirements process. A suggested way is to start by modeling the current physical system, and analyze this to determine the current logical system, and finally build the model of new logical system. Structured, object-oriented or formal modeling methods can be used in behavioral modeling.
- Domain modeling: It a significant proportion in RE process, because domain model provides an abstract description of the world in which a designed system will operate

and interacting with its environment. Explicit domain models permit detailed reasoning about the domain, and provide opportunities for requirements reuse within a domain.

- Modeling Non-Functional Requirements (NFRs): Modeling NFRs is more difficult since it is not easy to express and analyze NFRs as measurable way. Besides, NFRs as properties of a system as a whole cannot be verified for individual components. Recent investigations show that Xu et al. proposed a grouping mechanism to model NFRs in software architectures directly and explicitly (Xu et al., 2005). Saleh and Al-Zarouni proposed an approach to capturing non-functional software requirements using the user requirements notation (Saleh and Al-Zarouni, 2004). Cysneiros and Leite present a process to elicit NFRs, analyze their interdependencies, and trace them to functional conceptual models expressed by UML (Cysneiros et al., 2001).
- Analyzing requirements models: Modeling requirements provides opportunity for analyzing them. Investigated analysis techniques include requirements animation, automated reasoning, case-based reasoning and knowledge-based critiquing, consistency checking, and a variety of techniques for validation and verification (Nuseibeh and Easterbrook, 2000).

In software engineering process, a sequence of transformations is performed starting from requirements and ending with implementation to build a software system. Many researches devoted into the transformation between user requirements and analysis models in recent years. Tjoa and Berger proposed an approach to transform natural language based requirements specifications into an Extended Entity Relation (EER) model (Tjoa and Berger, 1993). Subramaniam et al. presented an approach to automating the transition from stakeholders' requests to use cases in OOADK (Subramaniam et al., 2004). Yue et al. presented a conceptual framework to provide common concepts and terminology and to define a unified transformation process (Tseng et al., 2005). Gorschek and Wohlin developed a Requirements Abstraction Model to response to the industrial need (Gorschek and Wohlin, 2006). This model consists of four abstraction levels: product level (goal), feature level (features), function level (functions/actions), and component level (details-consists of). However, thesis tasks are still mainly manually accomplished through iterative communication with the customer, which is often a recursive brainstorming process: gathering and formulating customer requirements, generating preliminary solutions, and refining customer requirements.

2.5 **Conceptual Models**

This research's goal is to propose a computer-aided modeling approach from natural language based design requirements to conceptual models. A conceptual model is a high-level description of an application. It enumerates all concepts in the application that users can encounter, describes how those concepts relate to each other, and explains how those concepts fit into tasks that users perform with the application (Johnson and Henderson, 2011).

The conceptual model is explicitly chosen to be independent of design or implementation concerns. The aim of a conceptual model is to express the meaning of terms and concepts used by users such as domain experts to discuss the problem, and to find the correct relationships between different concepts. The conceptual model attempts to clarify the meaning of various, usually ambiguous terms, and ensure that problems with different interpretations of the terms and concepts cannot occur. Such differing interpretations could easily cause confusion amongst stakeholders, especially those responsible for designing and implementing a solution, where the conceptual model provides a key artifact of business understanding and clarity. Once the concepts have been modeled, the model becomes a stable basis for subsequent development of applications. The concepts of the conceptual model can be mapped into physical design or implementation constructs.

A lot of conceptual models were developed or being developed in various engineering fields such as domain model, Entity-Relationship (ER) model, and Function-Behaviour-State (FBS). A conceptual model can be described using various notations, such as UML (OMG-UML, 2011) for object modelling, or Information Engineering (IE) or IDEF1X for Entity Relationship Modelling.

In software engineering, Universal Modeling Language (UML) (Fowler, 2003; Lano, 2009; Rumbaugh et al., 1998) is a widely adopted software modeling notation to specify, construct and document the artefacts of systems (OMG-UML, 2011). A large number of semantic approaches have been developed as subjects including Use Cases, class diagrams, state machines, interactions, OCL, and activity diagrams and so on. Each category has its own advantages and disadvantages, such as, Use Cases are popular due to their simplicity, acting as a bridge between technical and business stakeholders, the compact graphical nature to represent requirements, and even as a basis for managers when doing project estimation (Diev, 2006). However, Use Cases are helpful mainly to model functional requirements, but not for others like non-functional requirements. Also Use Case diagrams lack well-defined semantics, which may lead to differences in interpretations by stakeholders. Another typical conceptual model in UML notation is a class diagram in which classes represent concepts, associations represent relationships between concepts and role types of an association represent role types taken by instances of the modelled concepts in various situations. In ER notation, the conceptual model is described with an ER Diagram in which entities represent concepts, whereas cardinality and optionality represent relationships between concepts.

Many researches are conducted on the requirements modeling by UML. Liu et al. proposed a methodology with Use-case language schemas to automate natural language requirements analysis and class model generation based on the Rational Unified Process (RUP). They developed a CASE (Computer aided Software Engineering) tool, Use-Case driven Development Assistant (UCDA) to support their approach (Liu et al., 2004).

However, UML is a software-specific language, and does not support the general needs of designing in broader fields. Therefore, OMG Systems Modeling Language (OMG SysML™) (Friedenthal et al., 2008; OMG-SysML, 2011; Weilkiens, 2007) was created and has been steadily gaining popularity in different areas like Wölkl and Shea's work (Wölkl and Shea, 2009).

The SysML is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. In particular, the language provides graphical representations with a semantic foundation for modeling system requirements, behavior, structure, and parametrics, which is used to integrate with other engineering analysis models (OMG-UML, 2011). SysML represents a subset of UML 2 with extensions needed to satisfy the requirements of the UML™ for Systems Engineering RFP as indicated in Figure 2-2. SysML is the response to the UML for systems engineers' request for proposal, therefore, SysML was designed with "real" systems in mind, whereas UML is software oriented.

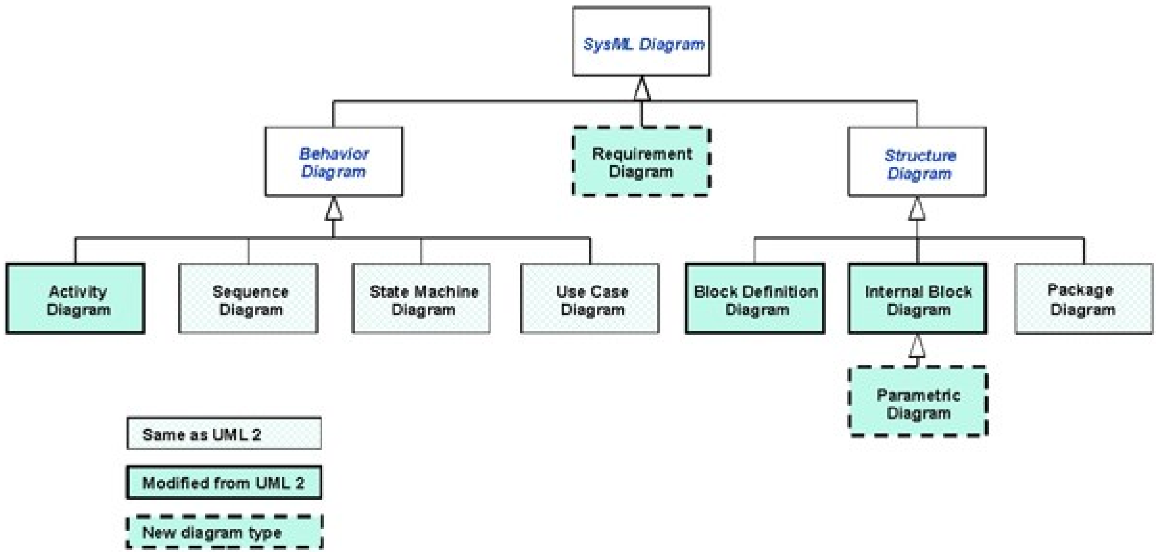


Figure 2-2 SysML Diagram Types (OMG-SysML, 2011)

SysML allows engineers to describe how a system interacts with its environment, and how its parts must interact to achieve the desired system behaviour and performance. The

SysML model provides a shared view of the system, enabling a design team to surface issues early and prevent problems that would otherwise delay development and degrade design quality. Since SysML is based on UML, it also facilitates integration between systems and software development. SysML can be used in many important activities during the system life cycle, such as in communication with stakeholders, improving system knowledge, model execution and verification, documentation for maintenance. The SysML Requirements diagram, the SysML Use Cases diagram, and the SysML Requirements table are applied to specify and model a list of user requirements for a road traffic management system (Soares and Vrancken, 2008).

SysML is a precise language, including support for constraints and parametric analysis, which allows models to be analyzed and simulated, greatly improving the value of the systems model, compared to textual system descriptions. SysML improves communication across team members and between teams by providing a formal language for sharing systems information among all project stakeholders. And SysML helps reduce errors and ambiguities during systems development processes by offering a more complete representation of systems. Therefore SysML is more and more adopted in model-based systems engineering.

In engineering, more function modeling oriented approaches are proposed to construct a basis for solving the representation problems of complex products and their complex development processes, such as, Gero proposed a dynamic design model using the concepts of function, behaviour, and structure with the transformation between these (Gero and Kannengiesser, 2007). Gero's Function-Behaviour-Structure (FBStr) is useful to demonstrate the conceptual relation between function, behaviour, and structure. Structure–behavior–function (SBF) (Bhatta and Goel, 1994; Bhatta and Goel, 1997) provides an ontology for teleological modeling, and SBF models of engineering systems have been used in computer programs for automated design and problem solving (Goel et al., 2009). Another conceptual model, Function-Behaviour-State (FBS) model of Umeda and Tomiyama provides a systematic method for decomposition and embodiment of functional design. FBS model as a framework represents a design object hierarchically and defines a function as an association of human intention and behaviour (Umeda and Tomiyama, 1995). With supports of the developed computer tool, FBS Modeller, the FBS modeling was extended and applied by researchers in the design research field (Chase and Liew, 2001; Deng, 2002; Erden et al., 2008; Gero and Kannengiesser, 2007; Umeda et al., 1990; Umeda and Tomiyama, 1995). Overall, the function modeling bridges the gap between the high-level requirements and the low-level details (Erden et al., 2008).

In the FBS modeling theory, function is defined as the bridge between human intention and physical behaviour of artifacts whereas the structure of a design object is represented hierarchically (Umeda and Tomiyama, 1995). Figure 2-3 shows the relationship among function, behaviour, and state.

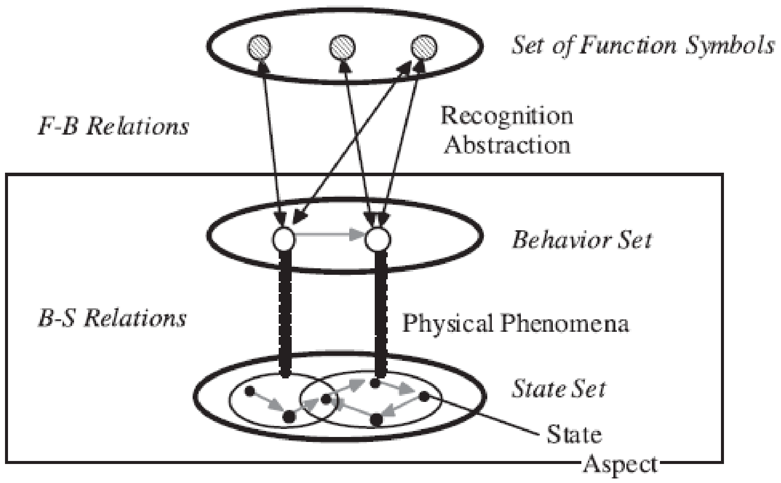


Figure 2-3 Relationship between function, behaviour and state (Umeda and Tomiyama, 1995)

FBS modeling requires that its users understand the product requirements thoroughly and distinguish the different functional stages and relationship between the functions. This could be a challenging task for a complex engineering project. Design document for a complex engineering product or process may include a great amount of information, which is extremely tedious for human processing. To support the application of the FBS modeling theory, a software tool – the FBS modeler – is developed to support the conceptual design. The FBS modeler provides a function decomposition method, which includes causal and task decompositions (Umeda and Tomiyama, 1995). The decomposition process largely depends on the designer’s knowledge and experience with the FBS theory, which may result in different FBS models for the same design problem. Furthermore, design text mainly focus on describing the components and functionality of a product system, however, it is described by natural language, which may easily lead to different understanding due to its ambiguity. While, FBS modeling are more appropriate and helpful for design with more accurate and restricted styles.

2.6 **Linguistic Analysis**

A robust requirements engineering approach should have a robust notation system for modeling and documentation of user and system requirements or rationales, also for analysis of business and architecture. The traditional User Requirements Notation (URN) (Amyot, 2003) is a semi-formal, lightweight graphical language for modeling and analyzing requirements in the form of goals and scenarios. URN combines two existing notations: Goal-oriented Requirement Language (GRL) and Use Case Maps (UCMs). The URN aims to support the elicitation, analysis, specification, and validation of requirements. And it is the first standardization effort to address explicitly, in a graphical way and in one unified language, goals (non-functional requirements - GRL) and scenarios (functional requirements - UCMs), and the links between them (Amyot, 2003).

In order to support the smooth flow of the design process, it is critical to identify the semantic structure underlying in design requirements. Therefore, linguistic analysis is essential for extracting semantics from design text. A systematic online market research for requirements analysis using linguistic tools indicates that the use of linguistic instruments (Mich et al., 2004).

Chen studied the correspondence between English sentence structure and ER (Entity-Relationship) diagrams, and proposes eleven rules for translation of information requirements into ER model. The basic constructs of English, such as noun, verb, adjective, adverb, gerund, and clause, are found to have counterparts in ER diagrammatic technique (Chen, 1983).

Several CASE tools have been developed for research to supply the functionality of NL requirements. CoGenTex Inc. developed a prototype LIDA (Linguistic assistant for Domain Analysis), which provides linguistic assistance in the model development process (Overmyer et al., 2001). UCDA by Liu et al. can assist the software developer to geneate use-case diagrams, use-case specifications, robustness diagrams, collaboration diagrams and calss diagrams in IBM Rational Rose (Liu et al., 2004); CIRCE is used in systematic analysis of natural language requirements (Ambriola and Gervasi, 2006); Al-Safadi proposed a semi-automated approach by constructing of CASE tool, named DBDT (database-designing tool), to transform a natural language description into a conceptual data model of enhanced-ER model (Al-Safadi, 2009).

Zeng proposed a new graphic language called Recursive Object Model (ROM) to present natural language used in engineering. The ROM is not only the linguistic tool for capturing the semantics of the requirements text, but also a notation system to specify and discover requirements for a proposed system or an evolving system, and review such requirements for correctness and completeness (Zeng, 2008). Other NLP systems are developed in natural language based requirements modeling, Such as RELAX to address uncertainty in self-adaptive systems requirement (Whittle et al., 2010). Among these NLP systems, recently ROM has drawn a lot of attention, since it has proven sufficient to represent the technical English text. Seresht and Ormandjieva propose an automated assistance for use cases elicitation from user requirements text by applying ROM to represent software requirements (Seresht and Ormandjieva, 2008). Maletz applies our formalization process of product requirements, in which ROM serves the core foundation, into an integrated requirements modeling approach in product lifecycle management for his PhD research (Maletz, 2008). Christophe et al. propose a combination of both ROM and semantic disambiguation approaches for the refinement of the requirements, which presents the possibilities of both approaches in terms of formalizing requirements in order to enhance the entire design process by providing relevant and well-formed information on the initial conditions of the design problem. In this thesis, ROM and ROM Q/A setting the framework for requirements and formalizing their structure whereas the semantic disambiguation approach searches for the essence of each concept used in the description of the design problem (Christophe et al., 2011). It has been tested that ROM is feasible and effective for dealing with instruct natural language used in engineering documents where only statements are involved. Therefore, ROM provides foundation for our research.

Chapter 3
Research Foundations

This chapter introduces the theoretical foundations of the present thesis: axiomatic theory of design modeling (ATDM) (Zeng, 2002), environment-based design (EBD) (Zeng, 2004a; Zeng, 2004b), and Recursive Object Model (ROM) (Zeng, 2008). ATDM is a logical tool for representing and reasoning about object structure (Zeng, 2002). It provides a formal approach that allows for the development of design theories following logical steps based on mathematical concepts and axioms. EBD is a new design methodology derived from ATDM. It provides step-by-step procedures to guide a designer in an environment changing process (Zeng, 2004a; Zeng, 2004b; Zeng, 2011). Three activities of environment analysis, conflict identification, and solution generation constitute the environment-based procedures. The proposed requirements modeling approach will enrich the EBD from content to practice. Meanwhile, one of the key methods for environment analysis in EBD is linguistic analysis. Echoing the recursive design logic, ROM is designed to support the processing of semantics in design.

3.1 **Axiomatic Theory of Design Modeling (ATDM)**

Axiomatic theory of design modeling is a logical tool for representing and reasoning about structures of design, especially the conceptual design (Zeng, 2002). It provides a formal approach that allows for the development of design theories following logical steps based on mathematical concepts and axioms. The primitive concepts of universe, object, and relation are used in the axiomatic theory of design modeling, based on which two axioms are defined in the axiomatic theory of design modeling.

[Axiom1] Everything in the universe is an object.

[Axiom 2] There are relations between objects.

Two corollaries of the axiomatic theory od design modeling are introduced to represent various relations in the universe.

[Corollary1] Every object in the universe includes other objects. Symbolically,

$$\mathrm{A} \supseteq \mathrm{B}, \forall \mathrm{A}\, \exists \mathrm{B}, \tag{3-1}$$

[Corollary2] Every object in the universe interacts with other objects. Symbolically,

$$\mathrm{C} = \mathrm{A} \otimes \mathrm{B}, \forall \mathrm{A}, \mathrm{B}\ \exists C, \tag{3-2}$$

where C is called the interaction of A and B.

Based on the Corollary 1 and 2, a key concept in ATDM, the structure operation is developed to model the structure of complex objects. The structure operation, denoted by $\oplus$, is defined as the union ($\cup$) of an object O and the interaction ($\otimes$) of the object with itself. The structure operation is developed.

$$\oplus \mathrm{O} = \mathrm{O} \cup (\mathrm{O} \otimes \mathrm{O}), \tag{3-3}$$

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

In addition, an object is primitive if and only if

$$\oplus O = O. \tag{3-4}$$

A primitive object includes only one object. The designation of a primitive object depends on the context of design and the designer's expertise.

3.2 **Environment-Based Design (EBD)**

The traditional view of the design process is that design evolution goes through the following stages: specification of design requirements, design synthesis, and design evaluation. These three stages iterate until a satisfying design solution is found.

Zeng and Cheng indicated that design is a recursive process in which a satisfying design solution must pass an evaluation defined by the design knowledge that is recursively dependent on the design solution to be evaluated (Zeng and Cheng, 1991). Since the design knowledge, which implies the design criteria, is part of the design problem, the generation of design solutions indeed changes the original design problem. This observation leads to the proposal of the recursive logic as the logic of design (Zeng and Cheng, 1991). Based on this logic, the design process is described as a series of design states defined by both product descriptions and product requirements, as is shown in Figure 3-1 (Zeng and Gu, 1999), where design requirements and design solutions co-evolve throughout the design process. Therefore, it is fundamentally impossible to distinguish design problem and design solutions.

When the word design problem is used, the design problem is given with a partial solution attached. When the word design solution is used, the design solution is bundled with further design problems. In this thesis, design states, design problem, and design solutions will be used interchangeably to mean the state of design. The recursive structure of design can be formally represented by the evolution of both design requirements and product descriptions (Zeng and Gu, 1999).

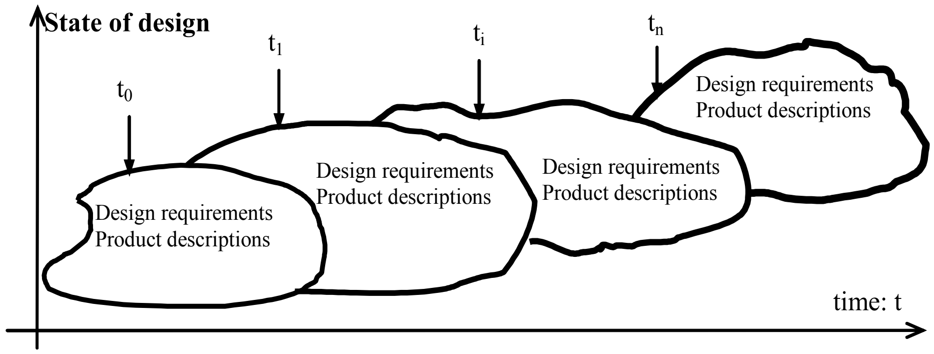


Figure 3-1 Evolution of the design process (Zeng, 2004b)

Different from traditional design methodologies, which are largely based on the understanding that a generic design process comprises analysis, synthesis, and evaluation, the Environment-Based Design Theory (Zeng, 2004a; Zeng, 2004b; Zeng, 2011) was logically derived from the axiomatic theory of design modeling (Zeng, 2002), which was founded on the recursive logic of design (Zeng and Cheng, 1991). EBD is a prescriptive model of design that guides designers from the elicitation of customer requirements throughout the generation and evaluation of design concepts. Also, EBD is a descriptive model of the design process that illustrates how designers accomplish a design task.

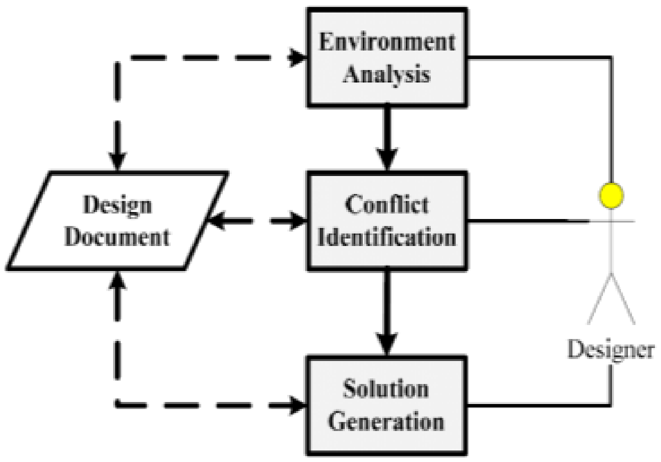


Figure 3-2 EBD: process model (Zeng, 2011)

As is illustrated in Figure 3-2, EBD includes the following three main activities: environment analysis, conflict identification, and concept generation. These three activities work together to update environment and its internal relationships to generate progressively and simultaneously and refine the design specifications and design solutions.

The objective of environment analysis is to identify the key environment components, in which the product works, and the relationships between the environment components and as well as between product and environments. From the environment implied in the design problem described by the customer(s), the designer will introduce extra environment components that are relevant to the design problem at hands. The results from this analysis constitute an environment system. One of the key methods for environment analysis is linguistic analysis (Chen *et al.*, 2007). Following the environment analysis, conflicts should be identified among the relations between environment components. At the third stage of EBD, a set of key environment conflicts will be chosen to be resolved by generating some design concepts. This process continues until no more unacceptable environment conflicts exist.

It is shown that both design requirements and product descriptions, as illustrated in Figure 3-1, are implied in product system (Zeng et al., 2004), which is called Product-Environment System (PES) in this thesis. A PES is defined as the structure of an object (Ω) including both a product (S) and its environment (E).

$$\oplus\Omega = \oplus(E \cup S) = (\oplus E) \cup (\oplus S) \cup (E \otimes S) \cup (S \otimes E), \forall E, S[E \cap S = \Phi], \tag{3-5}$$

where Φ is the object that is included in any object. ⊕E and ⊕S are structures of the environment and product, respectively; E⊗S and S⊗E are the interactions between environment and product. A PES can be illustrated in Figure 3-3.

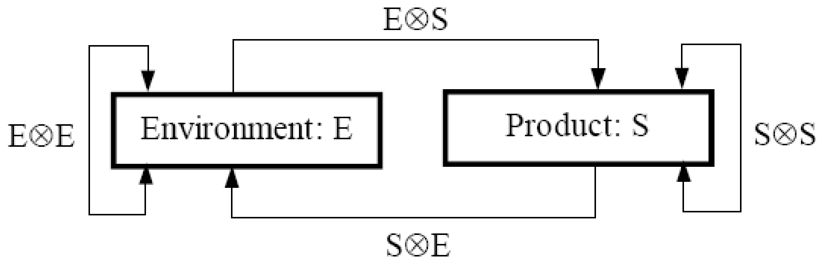


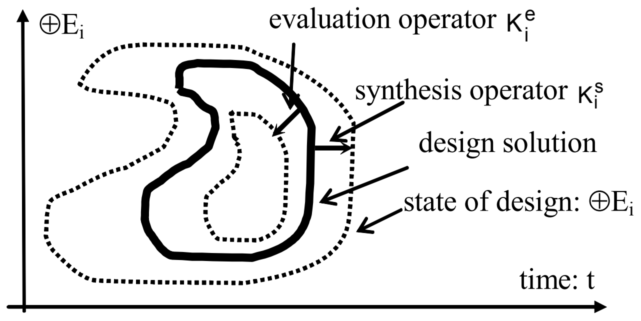
Figure 3-3 Product-Environment System (Zeng et al., 2004)

Since environment as well as product may have components, structures ⊕E and ⊕S can be further decomposed into the structures of these components as well as their mutual interactions according to the definition of structure operation. Eq. (3-5) indeed presents a recursive structure of a product system. Therefore, the structure operation provides a mechanism that can flexibly represent the structure of any complex object.

EBD theory indicates that the source of design requirements is product environment E (Zeng, 2004b). During the environment based design process, the evolution from the design state $\oplus E_i$ to the design state $\oplus E_{i+1}$ is governed by the following design governing equation, where K_i^s and K_i^e are evaluation and synthesis operators, respectively.

$$\oplus E_{i+1} = K_i^s(K_i^e(\oplus E_i)). \tag{3-6}$$

The synthesis operator stretches the state space of design whereas the evaluation operator folds and reduces the state space. The final design solution is the balance of those two forces. This governing equation is indeed another form of the recursive logic of design (Zeng and Cheng, 1991). (3-6) illustrates this governing equation.



$\oplus E_i$
evaluation operator K_i^e
synthesis operator K_i^s
design solution
state of design: $\oplus E_i$
time: t

Figure 3-4 State space of design under synthesis and evaluation operators (Zeng, 2004b)

The EBD with its theorem of design logic, design evolution, design formulation and design process provides theoretical foundation for this thesis in requirement elicitation and transformation.

3.3 **Recursive Object Model (ROM)**

Recursive object model (ROM) is a graphic representation of linguistic structure, derived from axiomatic theory of design modeling (Zeng, 2008). ROM uses five symbols to represent primitive object, compound object, constraint relation, predicate relation and connection relation, as shown in Table 3-1. These objects and relations can be mapped in the design problem described by natural language. ROM can be used to collect, organize, interpret, and analyze the characteristics by inferring from multiple object relationships implied in the natural language.

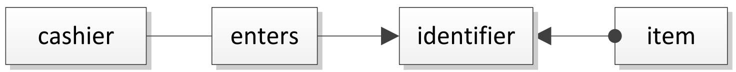
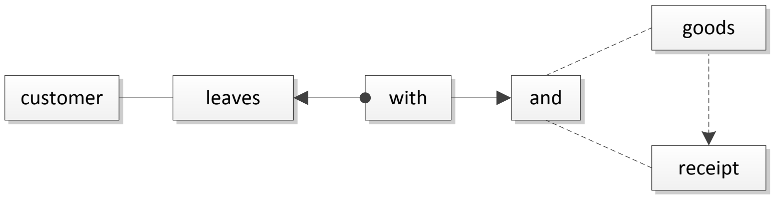
Table 3-1 Elements of recursive object model (ROM) (Zeng, 2008)

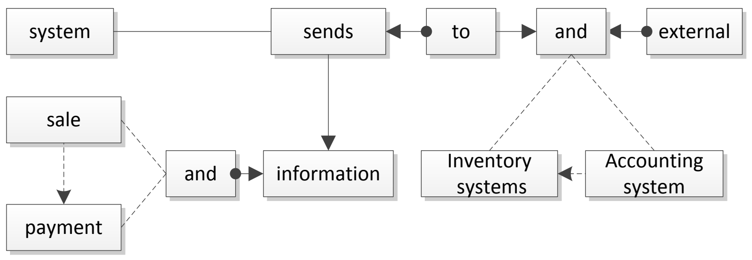
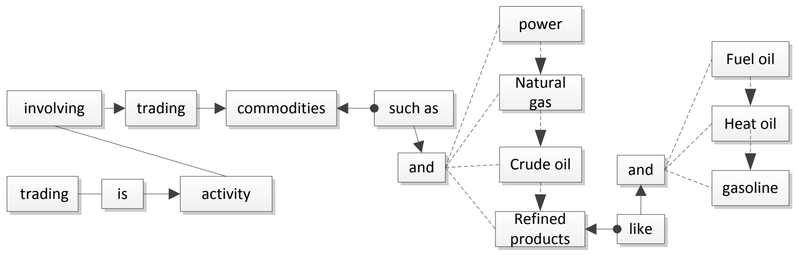
Type		**Symbol**	**Description**
Object	Primitive Object	$\boxed{O}$	Everything in the universe
	Compound Object	$\boxed{\boxed{O}}$	An object that includes at least 2 other objects
Relations	Constraint Relation	$\bullet\!-\boxed{\xi}\!\rightarrow$	A descriptive or limiting relation
	Connection Relation	$--\boxed{\iota}\dashrightarrow$	To connect two objects that do not constrain each other
	Predicate Relation	$-\boxed{\rho}\!\rightarrow$	An object's action on the other or an object's states.

ROM has been applied to software engineering (Seresht and Ormandjieva, 2008), language translation (Wen et al., 2013; Wen et al., 2011), requirements elicitation (Wang and Zeng, 2009), and cognitive design research (Zhu et al., 2007). Such as Seresht and Ormandjieva used this model and Expert Comparable Contextual (ECC) models to elicit use cases from requirements text (Seresht and Ormandjieva, 2008). Chen and Zeng proposed an approach to automatically transform a requirement text into two UML diagrams – use case and class diagram based on ROM (Zeng, 2008). It has been proved that ROM is effective for the collection of the right information, identification of conflicts, and solution generation (Zeng, 2011)

Table 3-2 shows several examples of ROM diagrams to illustrate how to represent natural language using ROM.

Table 3-2 Examples of ROM diagrams

Natural language	**ROM diagram**
Cashier enters item identifier.	
Customer leaves with receipt and goods.	

System sends sale and payment information to the external Accounting system and Inventory system.	 system sends to and external sale and information Inventory systems Accounting system payment
Energy trading is the activity involving trading energy related commodities, such as power, natural gas, crude oil, and refined products like fuel oil, heat oil, gasoline.	 power Fuel oil Natural gas involving trading commodities such as Heat oil and Crude oil and trading is activity gasoline Refined products like

Chapter 4
Environment-based Requirements Roadmap and Dynamic Requirements Elicitation based on Requirements Roadmap

This chapter presents an environment-based roadmap for requirements in terms of the lifecycle of a product and the environment components that the product resides in. This roadmap builds up the criteria for completeness and necessity of the requirements. The first criterion classifies the product requirements in terms of the product life cycle whereas the second classifies them by different levels from natural, built and human environments.

In this chapter, the environment-based roadmap is applied in the dynamic inquiry approach for eliciting requirements from design text represented by ROM diagram. A case study of energy trading system is used to show the feasibility of this approach.

The framework of this chapter is illustrated in Figure 4-1, in which, clarifying and explicating the initial requirements towards refined requirements is necessary preparation for effective requirements elicitation. Based on our research, a question-asking based communication is effective in collecting and refining requirements based on a well-defined requirement roadmap. The research foundation of requirements roadmap and Question-Asking strategy is the Environment Based Design (EBD) methodology while Recursive Object Model (ROM) has been found to be a valuable tool for representing the semantics of design requirements in both question asking and semantic analysis.

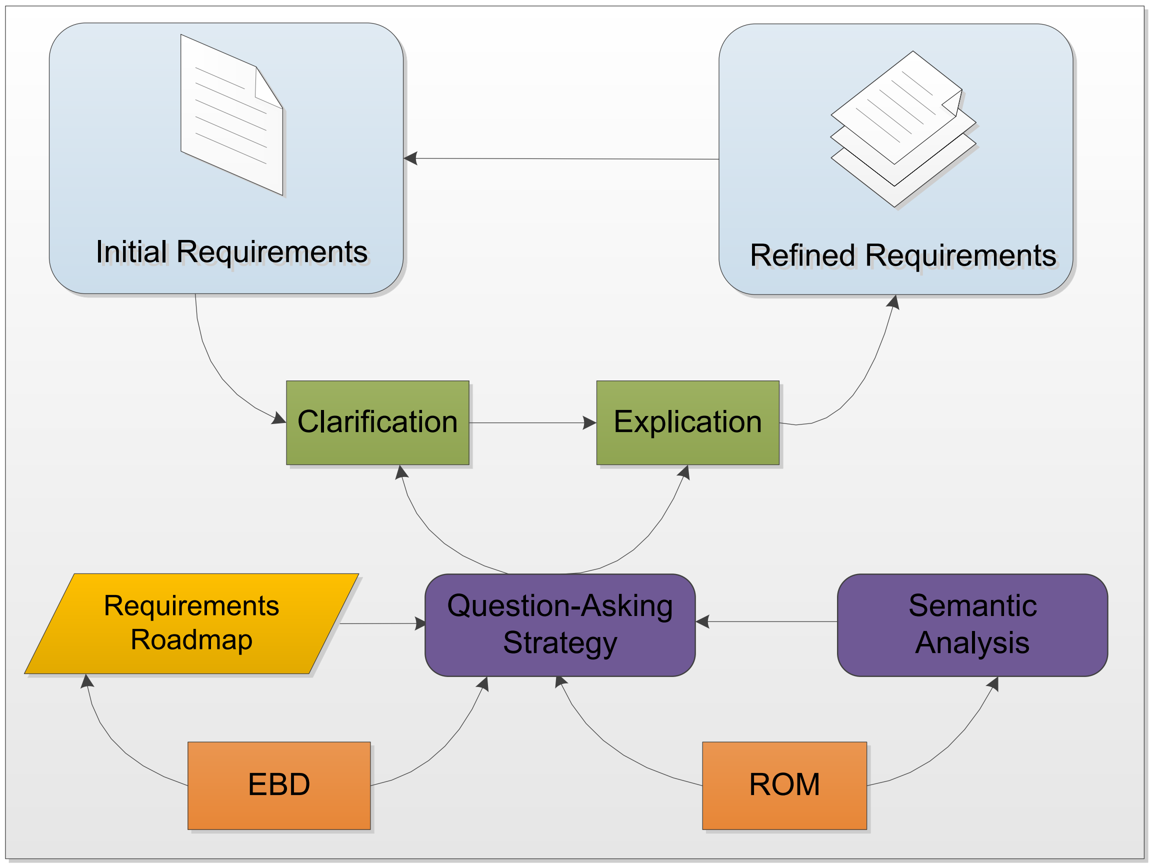


Figure 4-1 Requirement clarification and explication

4.1 Environment-based Requirements Roadmap

Requirements elicitation is the first and indispensable stage in the product life cycle. Therefore, the adequate list of requirements to be elicited at certain stage of product life cycle is important in requirements gathering process. If these requirements are incomplete, it may cause a huge waste of resource to make up the missing requirements in the later stages; however, if else too much or necessary requirements are considered at earlier stages, they may limit the product in some degree, therefore the best solution may be missed.

For the success of a product design, high quality requirements are demand: unambiguous, verifiable, precise, independent, necessary, consistent, understandable, clear, complete, nonredundant etc. (Hull et al., 2005; Leffingwell and Widrig., 2003; Young, 2001; Zielczynski, 2007). Among these criteria, some are related to representation of natural language and are easy to be implemented, such as unambiguous, understandable and clear. Whereas, the criteria of necessary and complete are closely related to others and fatefully impact the whole design process. Even there are no clear criteria about the necessity and completeness to implement. Therefore, this thesis is focus on proposing a roadmap for the two requirement criteria: necessity and completeness.

Necessity means to collect right requirements at right time, whereas completeness means to elicit all the requirements based on the environments throughout the whole life cycle. It is difficult to clearly define the criteria of completeness and necessity of requirements. However, it is feasible to approach the goal directed by an effective roadmap. This section proposes an environment-based requirements roadmap for collecting necessary and complete requirements.

4.1.1 Necessity of requirements

The necessity of requirements is addressed by the logic of design. The traditional view of the design process is that design evolution goes through the following stages: specification of design requirements, design synthesis, and design evaluation. These three stages iterate until a satisfying design solution is found. While in recursive design logic of view (Zeng and Cheng, 1991), the design process is described as a series of design states defined by both product descriptions and product requirements, as is shown in Figure 3-1 (Zeng and Gu, 1999), where design requirements and design solutions co-evolve throughout the design process or life cycle. When the word design problem is used, the design problem is given with a partial solution attached. When the word design solution is used, the design solution is bundled with further design problems. Therefore, it is fundamentally impossible to distinguish design problem and design solutions.

The requirements depend on the changes of solutions, which cause the uncertainty of design requirements. Therefore, to decrease the times of iteration in life cycle, we have to find out the only necessary requirements at certain stage.

In the design process illustrated in Figure 4-2, the design state could evolve to a new design state with a more abstract or more detailed design solution. The more abstract design state often implies a design problem that reflects better the customer's real intent. While the more detailed design state implies more complete requirements and product descriptions. In a design stage, specific design requirements should be identified for the design solution to the stage. If more requirements out of the stage are determined at a specific time, the design solution may be limited by requirements. While, if less requirements are given at the specific time, the design solution could be beyond the requirements. Therefore, specific requirements should be collected at specific time for accurate design solutions.

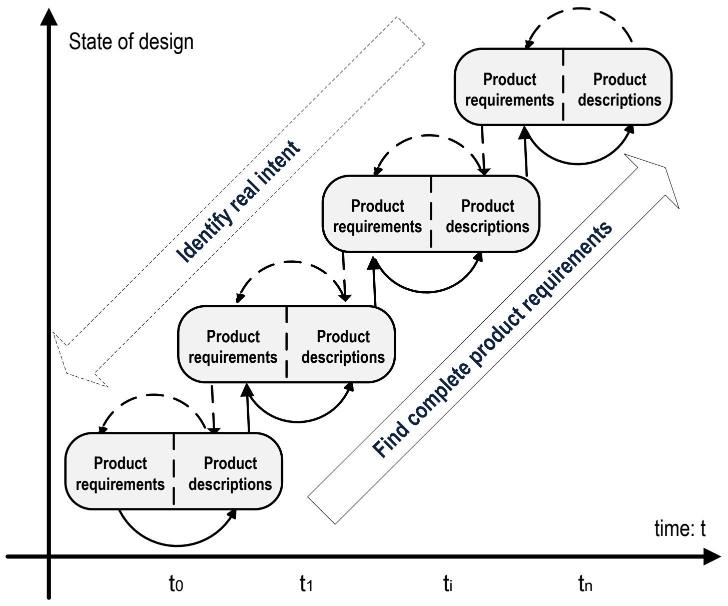


Figure 4-2 Evolution of the design process (Wang and Zeng, 2009)

4.1.2 Completeness of requirements

Based on the theorem of ATDM, It is shown that all the product requirements in a design problem are imposed by the product environment (E) in which the product is expected to work (Zeng, 2004b). In EBD theory, the source of design requirements is product environments. Product environments are the driving forces of a design process and provide a foundation for the classification and management of the product requirements (Zeng, 2004b).

Illustrated in Figure 3-3, both design requirements and product descriptions are implied in Product-Environment System (PES). A PES is defined as the structure of an object (Ω) including both a product (S) and its environment (E), which is represented in Equation (4-1).

$$\oplus\Omega = \oplus(E \cup S) = (\oplus E) \cup (\oplus S) \cup (E \otimes S) \cup (S \otimes E), \forall E, S [E \cap S = \Phi], \qquad (4\text{-}1)$$

Product requirements are part of interactions between product and environment. Design constraints belong to the relations from environment E to the product S (E⊗S) whereas product functions belong to the relations from product S to environment E (S⊗E).

It is relatively easy to identify the environment in which the product is expected to work. In general, the product environment can be partitioned into a finite number of sub-environments. It can be observed from the statement that any product will work in three environments: natural, built, and human. To work in natural environment, a product should obey all natural laws, otherwise the product will not be able to exist. This involves requirements such as safety and reliability. The built environment includes all artifacts built or created by human beings. To work in the built environment, a product must satisfy the requirements such as manufacturability and transportability. The human environment includes all human users and operators in the life cycle of a product. To survive in the human environment, a product must satisfy the requirements such as salability, operability, and maintainability.

Obviously, different ways to organize the components in product environment will lead to different formulations of product requirements. Such as Chen and Zeng formulate design problems in terms of different classification schemes of environment as natural laws, social law and regulations, technical limitation, cost, time and human resource, basic functions, extended functions, exception control level, and human-machine interface (Chen and Zeng, 2004). Corresponding to the subjective and objective realms adopted by Erden et al (Erden et al., 2008), environments can be divided into subjective and objective environments (Wen et al., 2013). The subjective environments include the users of the product whereas the objective environments include all other environment components that have impact on the behaviour of the product.

The environment components and the relationships between these environment components compose the environment system. Theoretically, the completeness of requirements depends on the environments of the product: the more environment components and their relations are considered, the more complete requirements are collected.

4.1.3 Environment-based requirements roadmap

From demand and supply points of view, design is a recursive process of generating requirements by the demand side and satisfying it by the supply side, which is usually the designer. And product environment could be defined by all the players included in the demand side. These players are human environments which perform different functions in the product life cycle. For effective eliciting complete and necessay product requirements at different stages in the process of product design, it is useful to classify and order these requirements in terms of product life cycle.

The Product Life Cycle (PLC) is used to map the lifespan of a product. There are specific stages in the life of a product for different disciplines such as system engineering, product design, manufacturing, software engineering, marketing etc. From an engineering perspective, the stages of product life cycle include design, manufacture, sales, transportation, use, maintenance, and recycle (Chen and Zeng, 2006). A typical life cycle of software includes requirements phase, specification phase, design phase, implementation phase, integration phase, maintenance phase and retirement (Schach, 2002). These are four stages of introduction stage, growth stage, maturity stage and decline stage in marketing (Esmaeilsabzali et al., 2010).

An environment-based requirements roadmap is proposed in this thesis, which is illustrated in Figure 4-3. From definition the roadmap is a plan or guide to show how something is arranged or can be accomplished. This roadmap categorizes product environments in terms of two criteria. One criterion partitions product environments based on the product life cycle. The other criterion classifies the product environment into natural, built, and human environments. Considering both of product life cycle and environment components will help for eliciting necessary requirements at specific stages and for complete requirements for the whole life cycle.

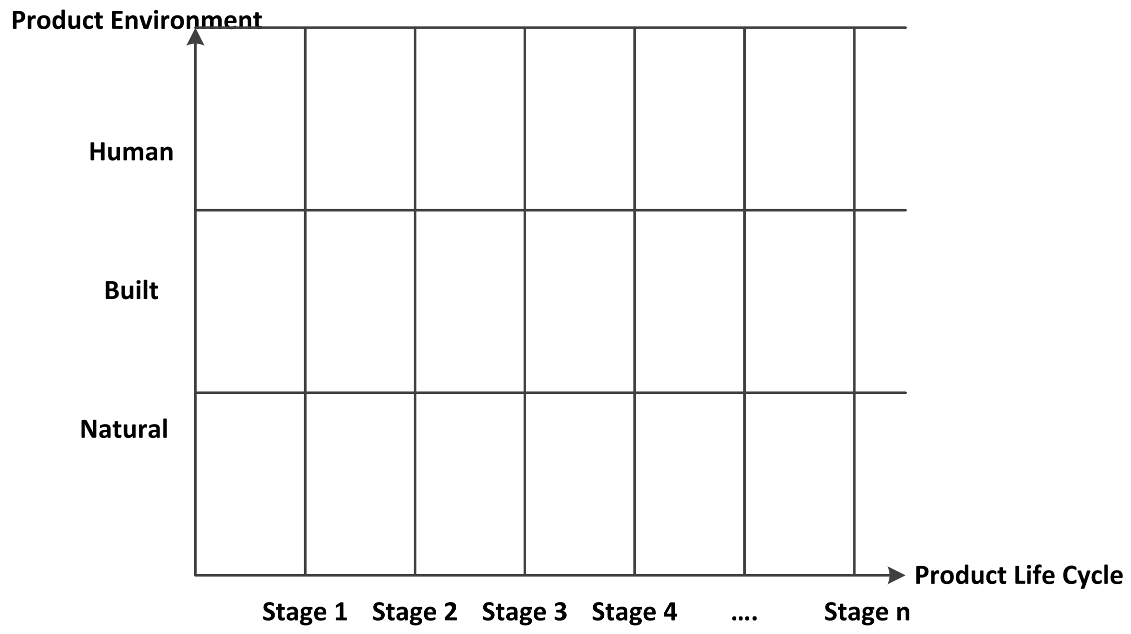


Figure 4-3 Requirements roadmap

The proposed roadmap describes the three basic environment categories for the whole lifecycle of the product to be designed. Any requirement stems from an environment at specific stage of life cycle. While the detailed environment components and the stages of product life cycle are specific with that product.

In software engineering, it is recognized that requirements are categorized into project drivers, project constraints, design constraint, functional requirements, non-functional requirements, and project issues. For example Volere requirement template list the detailed category for requirement document, which is shown in Table 4-1 (Volere, 2010).

Table 4-1 The requirement category in Volere

Category	Order	Requirement item	Details
Project drivers	R1	Purpose of the project	a. The user business or background of the project effort b. Goals of the project
	R2	Stakeholders	a. The client b. The customer c. Other stakeholders d. The Hands-On Users of the product e. Personas f. Priorities assigned to users g. User participation h. Maintenance users and service technicians
Project constraints	R3	Mandated constraints	a. Solution constraints b. Implementation environment of the current system c. Partner or collaborative applications d. Off-the-Shelf software e. Anticipated workplace environment f. Schedule constraints g. Budget constraints
	R4	Naming conventions and terminology	a. Definitions of all terms, including acronyms, used in the project
	R5	Relevant facts and assumptions	a. Relevant facts b. Business rules c. Assumptions
Functional requirements	R6	The scope of the work	a. The current situation b. The context of the work c. Working partitioning d. Specifying a business use case (BUC)
	R7	Business data model and data dictionary	a. Data model b. Data dictionary
	R8	The scope of the product	a. Product boundary b. Product use case table
	R9	Functional and data requirements	a. Functional requirements

Non-functional requirements	R10	Look and feel requirements	a. Appearance requirements b. Style requirements
	R11	Usability and humanity requirements	a. Ease of user requirements b. Personalization and internationalization requirements c. Learning requirements d. Understandability and politeness requirements e. Accessibility requirements
	R12	Performance requirements	a. Speed and latency requirements b. Safety-critical requirements c. Precision or accuracy requirements d. Reliability and availability requirements e. Robustness or fault-tolerance requirements f. Capacity requirements g. Scalability or extensibility requirements h. Longevity requirements
	R13	Operational and environmental requirements	a. Expected physical environment b. Requirements for interfacing with adjacent systems c. Productization requirements d. Release requirements
	R14	Maintainability and support requirements	a. Maintenance requirements b. Supportability requirements c. Adaptability requirements
	R15	Security requirements	a. Access requirements b. Integrity requirements c. Privacy requirements d. Audit requirements e. Immunity requirements
	R16	Cultural and political requirements	a. Cultural requirements b. Political requirements
	R17	Legal requirements	a. Compliance requirements b. Standards requirements
Project issues	R18	Open issues	
	R19	Off-the-shelf solutions	a. Ready-made products b. Reusable components c. Products that can be copied
	R20	New problems	a. Effects on the current environment

			b. Effects on the installed systems c. Potential user problems d. Limitations in the anticipated implementation environment that may inhibit the new product e. Follow-up problems
	R21	Tasks	a. Project planning b. Lanning of the development phases
	R22	Migration to the new product	a. Requirements for migration to the new product b. Data that has to be modified or translated for the new system
	R23	Risks	
	R24	Costs	
	R25	User documentation and training	a. User documentation requirements b. Training requirements
	R26	Waiting room	
	R27	Ideas for solutions	

However, on account to the complexity and workload of modern software, it is not facilitative to apply the template in a computer-aided environment. In proposed Environment-based requirement roadmap, the three types of environments can be further classified into seven categories as Volere, whereas the stages correspond to the seven stages of product life cycle. This Environment-based requirement roadmap for software product is illustrated in Table 4-2, in which at each stage of software life cycle, different category environment should be considered for collecting requirements.

Table 4-2 Environment-based requirement roadmap category

Environments		**Stages in product life cycle**						
		Requirements	**Specification**	**Design**	**Implementation**	**Integration**	**Manufacture**	**Retirement**
Human Environments	Purpose	R1	R1	R1	R1	R1	R1	R1
	Stakeholders	R2	R2	R2	R2	R2	R2	R2
	Functional requirements	R6 R7 R8 R9	R6 R7 R8 R9	R6 R7 R8 R9	R6 R7 R8 R9	R6 R7 R8 R9	R6 R7 R8 R9	R6 R7 R8 R9
Built Environments	Constrains	R3 R4 R5	R3 R4 R5	R3 R4 R5	R3 R4 R5	R3 R4 R5	R3 R4 R5	R3 R4 R5
	Non-functional requirements	R10 R11 R12 R13 R14 R15 R16 R17	R10 R11 R12 R13 R14 R15 R16 R17	R10 R11 R12 R13 R14 R15 R16 R17	R10 R11 R12 R13 R14 R15 R16 R17	R10 R11 R12 R13 R14 R15 R16 R17	R10 R11 R12 R13 R14 R15 R16 R17	R10 R11 R12 R13 R14 R15 R16 R17
	Project issues	R18 R19 R20 R21 R22 R23 R24 R25 R26 R27	R18 R19 R20 R21 R22 R23 R24 R25 R26 R27	R18 R19 R20 R21 R22 R23 R24 R25 R26 R27	R18 R19 R20 R21 R22 R23 R24 R25 R26 R27	R18 R19 R20 R21 R22 R23 R24 R25 R26 R27	R18 R19 R20 R21 R22 R23 R24 R25 R26 R27	R18 R19 R20 R21 R22 R23 R24 R25 R26 R27
Natural Environments	Constrains	R3 R4 R5	R3 R4 R5	R3 R4 R5	R3 R4 R5	R3 R4 R5	R3 R4 R5	R3 R4 R5

4.2 Dynamic Requirements Elicitation Based on Environment-based Roadmap

This section introduces a dynamic inquiry approach to eliciting requirements based on proposed Environment-based roadmap. An algorithm of question asking is presented to assist requirements elicitation.

4.2.1 Dynamic elicitation through question asking

In recursive design process illustrated in Figure 3-1, design stage could evolve to a new stage. On the one hand, designers may have to remove some information from the current design state in order to identify the real intent. On the other hand, designers may supplement information to find complete product requirements. In this process, questions set up goals which lead the design stage to move in either of the two directions as shown in Figure 4-2.

A generic inquiry approach was proposed in an attempt to ensure that the real needs of the customers were being met and to support requirements extension process for dynamically eliciting more complete requirements (Wang and Zeng, 2009; Wang et al., 2013). The iterative inquiry process, illustrated in Figure 4-4, starts from design problem description described in natural language provided by customers. The design problem description will be represented as ROM diagram by a computer tool ROMA. Based on the ROM diagram the process finally gets the results of design requirements through two types of inquiries: generic questions and domain questions. The first type of questions are generated for the clarification and extension of the meaning of design problem whereas the second type of questions of domain specific questions are generated for implicit design information related to the current problem. An intermediate component is product-environment system, which is generated after design problem analysis and prepared for domain problem extension.

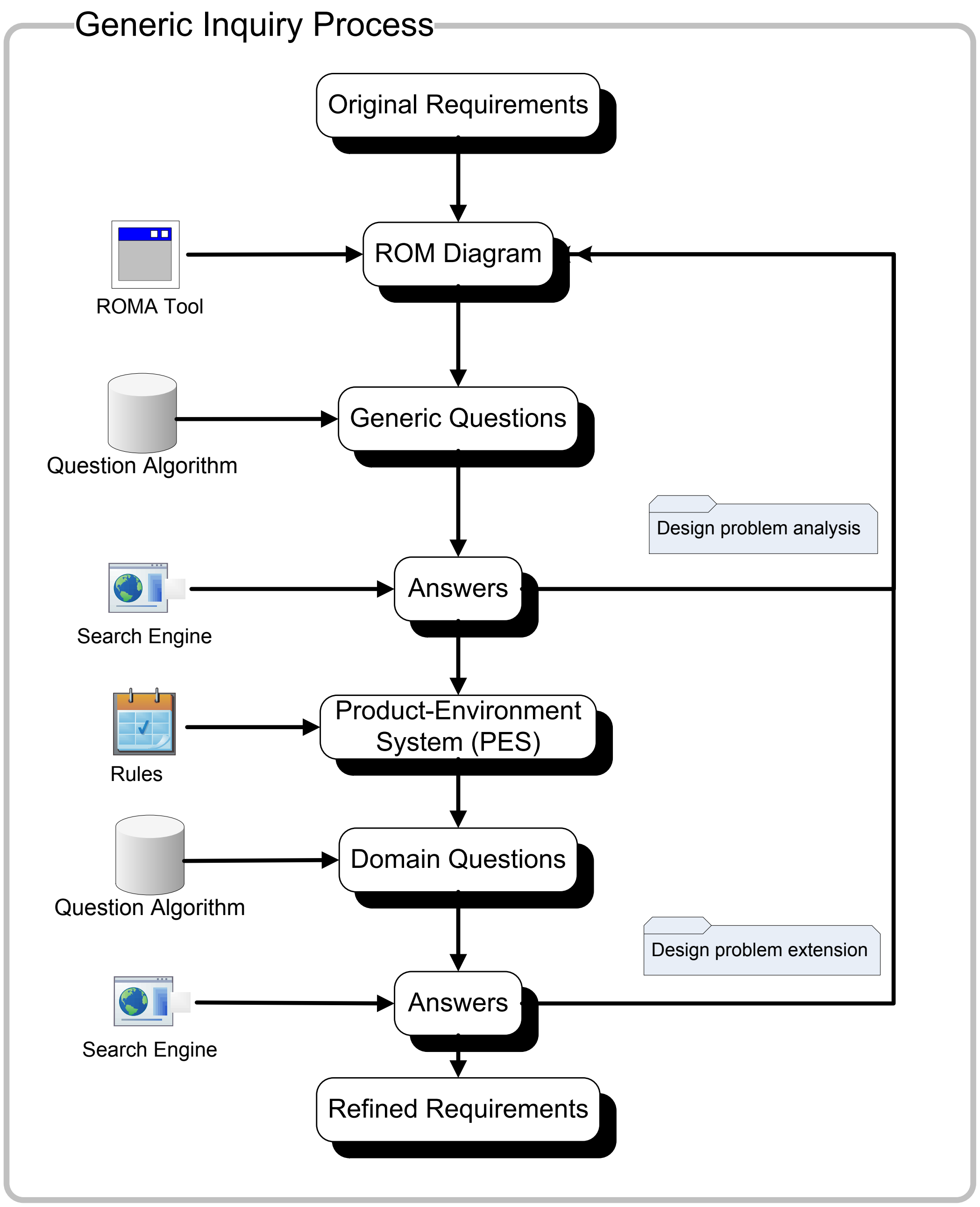


Figure 4-4 Generic inquiry process for requirements elicitation

Both of the inquiry processes generate questions by applying question generation algorithm which consists of question rules, templates and procedures.

4.2.2 Algorithm of question asking

The algorithm of question asking for requirements elicitation is shown in Table 4-3. The input is design text described by natural language, whereas the output is design requirements. The main steps in the algorithm are ROM diagram generation, generic question generation, PES identification and domain question generation.

Table 4-3 Question asking algorithm for requirements elicitation

```
Algorithm RequirementsElicitation(Text: design text)
1:    Transform design text into ROM diagram ROM←Transform(design text)
2:    Analyze ROM diagram through generic questions
3:    Update ROM diagram
4:    Identify Product-Environment System
5:    Extend design problem through domain questions
6:    Update Product-Environment System
7:    Output design requirements
```

Since the generation of ROM diagram from design text has been dealt with elsewhere (Zeng, 2008), this chapter focus on the other three steps.

4.2.2.1 Algorithm of generic question generation

The generic questions illustrated in Figure 4-5 aim to clarify the meanings of provided information to identify the customer’s real intent. The input of generic question generation is a ROM diagram corresponding to design problem description. The ROM diagram will go through an object analysis process to rank and record all objects and their adjacent objects into a list. Based on the object list, questions are generated by the algorithm illustrated in Table 4-3 through the rules given in Table 4-4 and templates in Table 4-5.

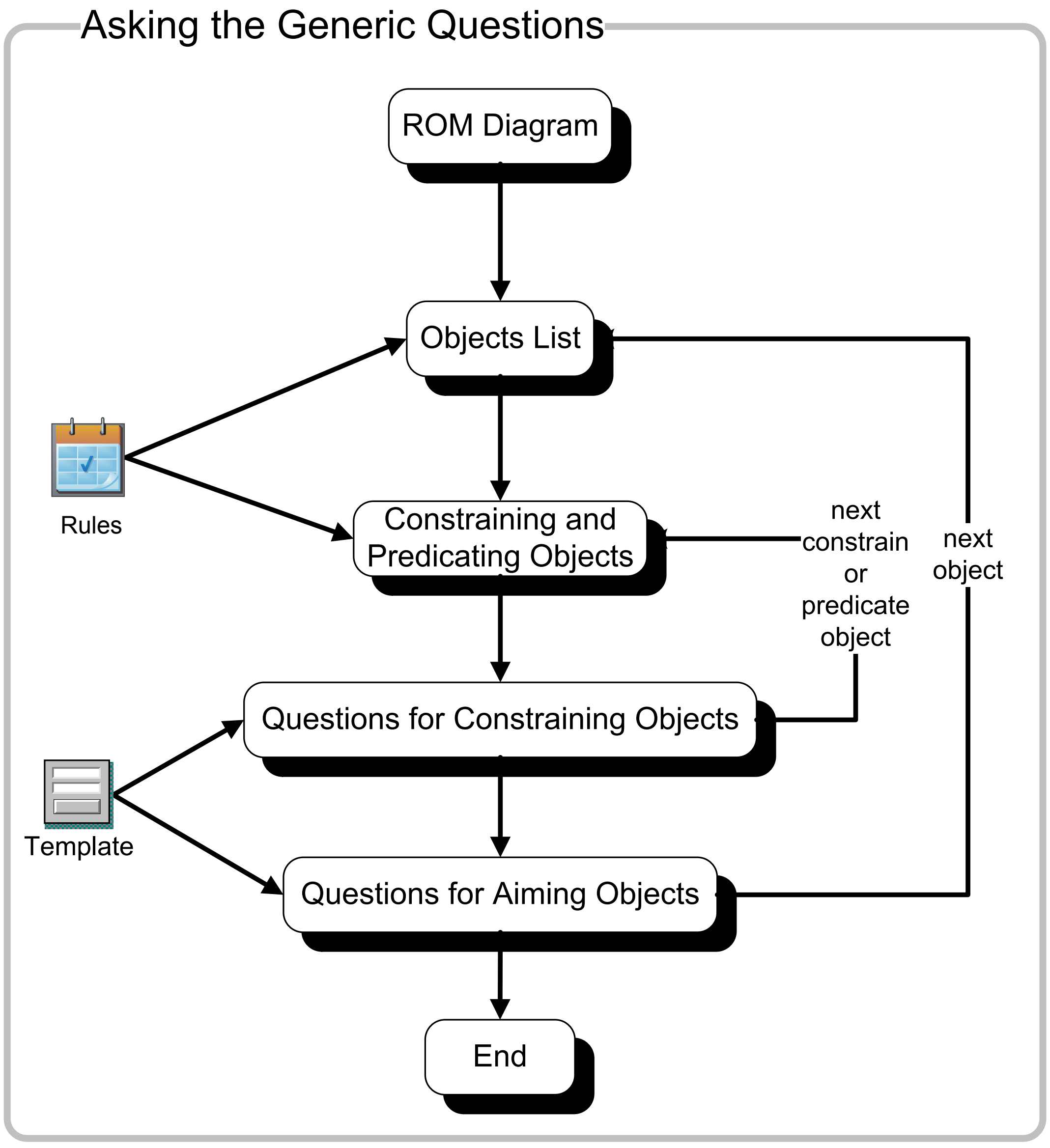

Asking the Generic Questions
ROM Diagram
Objects List
Rules
Constraining and Predicating Objects
next constrain or predicate object
next object
Questions for Constraining Objects
Template
Questions for Aiming Objects
End

Figure 4-5 Asking the generic questions

Table 4-4 Rules for objects analysis

• **Rule1**	An object with the most constrains (constraining objects and predicating objects) should be considered first.
• **Rule2**	Before an object can be further defined, the objects constraining them or having predicate relations with them should be further refined. Exceptions should be applied in sub rules. **Rule 2.1** Exceptions for the constraining objects: the objects of determiner, quantifier and number do not need to be refined; the objects of preposition and conjunction do not need to be refined, however, the objects connecting them should be refined further. **Rule 2.2** Exceptions for the predicate relations: some verbs including linking verbs (is/are/was/be), auxiliary verbs (have/has/had), and modal verbs (need/may) do not need to be refined, however, the objects connecting with them should be refined. **Rule 2.3** Exceptions for connecting objects: conjunctions (and/or/ but) do not need to be refined; however, their connected objects should be refined.
• **Rule3**	All the objects should be indicated "asked" if they have been refined, and they will not be asked repeatedly.

Table 4-5 Questions template for object analysis (Wang and Zeng, 2009)

T1	For a concrete, proper, or abstract noun *N*	Question: What is *N*?
T2	For a noun naming a quantity *Q* of an object *N*, such as height, width, length, capacity, and level, such as height, width, length, capacity,	Question: How many / much / long / big /… is the *Q* of *N*?

	level	
T3	For a verb *V*	Question: Why *V*? Who *V*? How to *V*? When *V*? or Where *V*?
T4	For a modifier *M* of a verb *V*	Question: Why *V M*?
T5	For an adjective or an adverb *A*	Question: What do you mean by *A*?
T6	For a relation *R* that misses related objects	Question: What (who) *R* (the given object)? Or (the given object) *R* what (whom)?

Table 4-6 Algorithm for generic question generation

```
Algorithm GenericQuestin(ROMDiagram: ROM)
1:    Determine the order of objects from ROM
2:    Repeat
3:        Ask a generic question
4:        Collect answer for the generic question
5:        Generate ROM diagram for the answer
6:        Update ROM diagram by merging the answer
7:        Repeat Step 1-6
8:    Until all the necessary objects in ROM are defined
9:    Output ROM diagram
```

4.2.2.2 Algorithm of PES identification

A PES is an important intermediate model, which is consist of products, environments, and relations among products and environments (Zeng, 2002). Products and environments can be further decomposed into components and attributes for different purposes. The algorithm of identification of PES from ROM is shown in Table 4-8, which applies the identification rules listed in Table 4-7.

Table 4-7 Identification rules from ROM diagram to PES

Identify product
Rule 1: If the POS feature of an object *O* in a ROM diagram is noun (*N*), and its ROM feature satisfies one of the following conditions, then this object is a product (P_r): 1) The object has a predicate relation directed from a human (E_h) object through a meta verb (V_m). 2) The object is a center object which has at least one predicate relation directing towards another object (O_x) through a function verb (V_f), but no predicate relation directed towards it from object (O_y) through a function verb.
Identify product components
Rule 2: If the POS feature of an object *O* in a ROM diagram is noun (*N*) and its ROM feature satisfies one of the following conditions, then this object is a product component (C_p): 1) The object is constrained by a product (P_r) or any other product component (C_p) object and is neither an attribute nor environment. 2) The object has a predicate relation directed from a product or product component through a structure verb (V_s). 3) The object is constrained by a preposition object (*P*) which connects a product or product component.
Identify product attributes
Rule 3: If the POS of an object *O* in a ROM diagram is noun (*N*) or adjective (A_j), and its ROM features satisfy one of the following conditions, then this object is an attribute (A_p) of a product (P_r) or component (C_p): 1) It constraining a product or a product component and is neither a product nor product component. 2) It has a predicate relation directed from a product or product component through a linking verb (V_l).
Identify product attribute value

Rule 4: If the POS feature of an object O in a ROM diagram is noun (N), adjective (A_j), or adverb (A_v), and the ROM feature satisfies one of the following conditions, then this object is an attribute value (V_a):

1) It constrains an attribute of a product or a product component.
2) It has a predicate relation directed from an attribute object through a linking verb (V_l).

Identify environments

Rule 5: If the POS feature of an object O in a ROM diagram is noun (N) and its ROM feature satisfies one of the following conditions, then this object is an environment (E_p) of a product (P_r) or product component (C_p):

1) It has a predicate relation directed from a product or product component through a function verb (V_f), but it is not a product, product component, attribute, or attribute value.
2) It has a predicate relation directed from an environment object through a function verb (V_f).
3) It constrains a function object (V_f) of a product or product component through a preposition object (P).

Environments can be classified further as the Rule 5.1 to Rule 5.3

Rule 5.1: If an object in a ROM diagram is an environment object and is also a human user or operator in the life cycle of the product, then this object is a human environment.

Rule 5.2: If an object in a ROM diagram is an environment object and is also an artefact built or created by human beings, then this object is a built environment.

Rule 5.3: If an object in a ROM diagram is an environment object but is neither a human nor a built environment, then this object is a natural environment.

Identify environment components

Rule 6: If the POS feature of an object O in a ROM diagram is noun (N) and its ROM feature satisfies one of the following conditions, then this object is an environment component (C_e):

1) The object is constrained by an environment (E_p) or environment component (C_e), but it is not a product or a product component.
2) The object has a predicate relation directed towards it from an environment object through a structure verb (V_s).
3) The object is constrained by a preposition object (P) which connects an environment object.

Identify environment attributes

Rule 7: If the POS of an object O in a ROM diagram is noun (N) or adjective (A_j), and the ROM features satisfy one of the following conditions, then this object is an attribute (A_e) of an environment (E_p):

1) It is constraining an environment (E_p) or an environment component (C_e).
2) It has a predicate relation directed from an environment or an environment component through a linking verb (V_l).

Identify environment attribute value

Rule 8: If the POS feature of an object *O* in a ROM diagram is noun (*N*), adjective (A_j), or adverb (A_v) and its ROM feature satisfies one of the following conditions, then this object is an environment attribute value (V_{ae}):
1) It constrains an environment attribute object (A_e).
2) It has a predicate relation directed from an environment attribute object and the predicate verb is a linking verb (V_l).

Identify relations

Rule 9: Relations exist among product, product components, and environments. Those relations reflect constraint relation or predicate relations in the ROM diagram.

Table 4-8 Algorithm for identification of PES from ROM

Algorithm of identifying PES *PES*(*ROMDiagram: ROM*)

```
1:    Determine the product object from ROM
2:    Repeat
3:        Identify environments
4:        Identify product components and attributes
5:        Identify environments from environments
6:    Until all the noun objects in ROM are defined
7:    Output the PES model
```

Algorithm of determining product *Product*(*ROMDiagram: ROM*)

```
1:    Sort the noun objects into a list $O_n$ by the number of predicate and
      constraint relations
2:    for all $O \in O_n$
3:        if $O$ satisfies Rule 1 then $Product \leftarrow O$
4:        else delete $O$ from $O_n$
5:        end if
6:    end for
```

Algorithm of identifying environments *Environment*(*Product or Component: S*)

```
1:    for all verb $V$ directed from $S$ to noun object $N$
2:        if $V$ satisfies Rule 11, $Environment \leftarrow N$, $Function \leftarrow V+N$
3:    end for
```

Algorithm of identifying product components and attributes *Component_Attribute*(*Product or Component: S*)

```
1:  for all $N$ which has a predicated relation directed from $S$ by $O_v$
2:      if $O_v$ is a structure verb that satisfies Rule 2.2) then $Component \leftarrow N$
3:  end for
4:  for all adjective $A_j$ which has a predicate relation directed from $S$ by $O_v$
5:      if $O_v$ is a linking verb that satisfies Rule 3.2) then $attribute \leftarrow A_j$
6:  end for
7:  for all $O_c \in O$ which is constraining $S$ through a preposition object
8:      if $O_c$ satisfies Rule 2.3) then $Component \leftarrow O_c$
9:  end for
10: for all $O_c \in O$ which is constrained by $S$
11:     if $O_c$ satisfy Rule 2.1) then $Component \leftarrow O_c$
12: end for
13: for all $O_a \in O$ which is constraining $S$
14:     if $O_a$ satisfy Rule 3.1) then $attribute \leftarrow O_a$
15: end for
```

Algorithm of identifying environments from an environment

Environment_Environment(*Environment: E*)

```
1:  for all $O_e$ which has a predicated relation directed from $E$ by $O_v$
2:      if $O_v$ is a function verb that satisfies Rule 5.2) then $Environment \leftarrow O_e$
3:      if $O_v$ is a structure verb that satisfies Rule 6.2) then $EnvironmentComponent \leftarrow O_e$
4:      if $O_v$ is a linking verb that satisfies Rule 7.2) then $EnvironmentAttribute \leftarrow O_e$
5:  end for
6:  for all noun object $O_e$ which is constrained by $E$
7:      if $O_e$ satisfies Rule 6.1) then $EnvironmentComponent \leftarrow O_e$
8:  end for
9:  for all noun object $O_e$ which is constrained by $E$ through a preposition object
10:     if $O_c$ satisfies Rule 6.3) then $EnvironmentComponent \leftarrow O_c$
```

```
11:     end for
```

4.2.2.3 Algorithm of domain question generation

Asking domain specific questions illustrated in Figure 4-6 aims to identify complete environment components and their relations for collecting complete requirements based on the EBD-based roadmap. The procedure for asking domain specific questions is given in Table 4-9. The input of this procedure is PES, which identifies the product, product components, product attributes, product environments, and relations among them. At the beginning of domain analysis, the PES may not complete and need to be supplemented systematically. The algorithm of domain question generation is shown in Table 4-10.

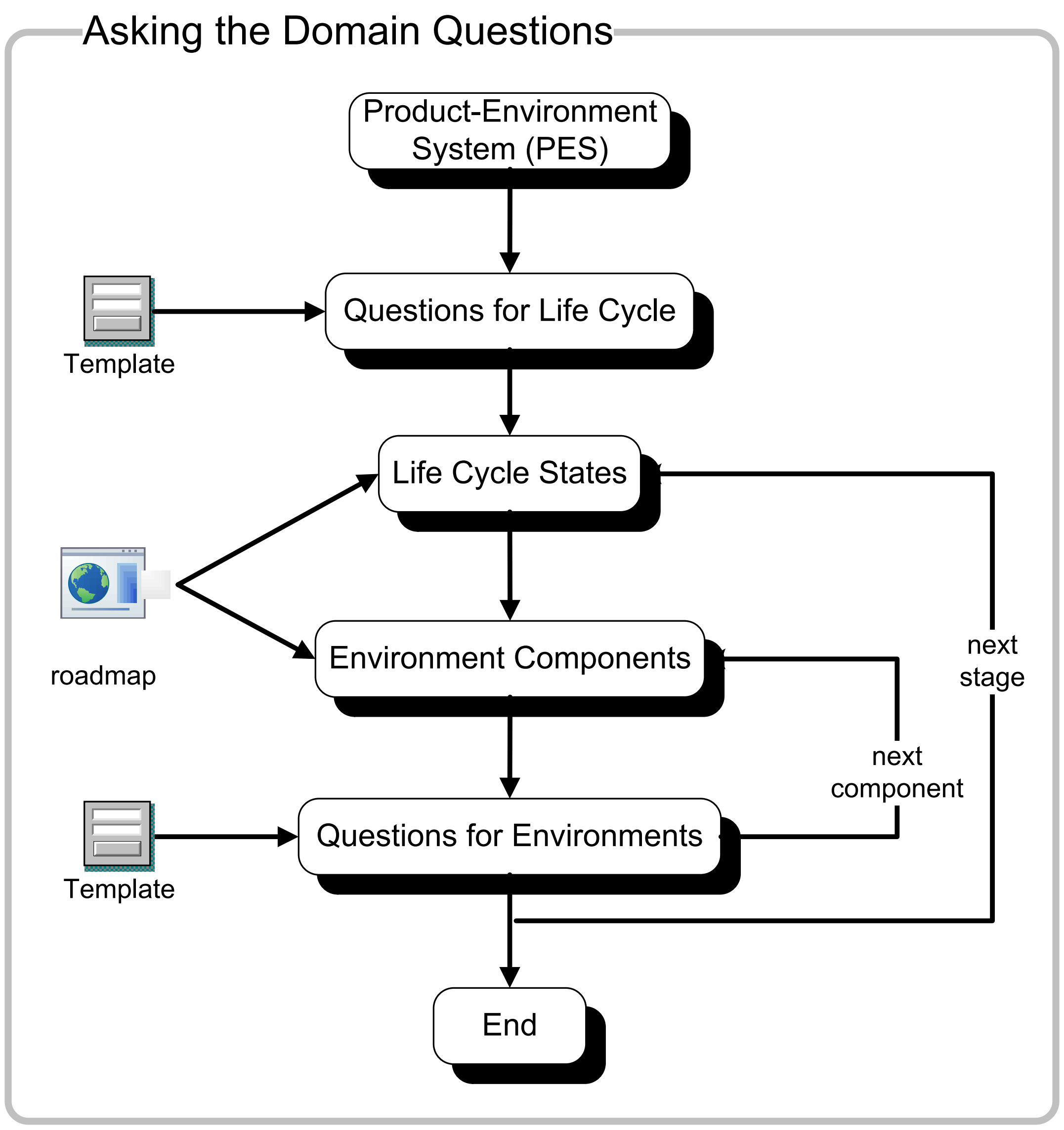


Figure 4-6 Asking domain specific questions

Table 4-9 Questions generation rules for domain specific questions

Rule 1	The life cycle of the product should be identified by asking the question: what is the life cycle of the product to be designed?
Rule 2	The environments should be identified at each stage of the lifecycle according to the EBD based roadmap by asking the

	question: what environments are related to the product in the stage X?
Rule 3	The relations between product and environments and between environments should be identified by asking the questions: what relations between A and B?
Rule 4	The sequence of questions is determined by the levels of requirements in roadmap so that those requirements at the lower levels have higher priority and can be asked earlier.
Rule 5	The answers should be gone through generic analysis for accuracy.

Table 4-10 Algorithm for domain question generation

```
Algorithm DomainQuestion(PES)
1:    Get product and environment objects from PES
2:    Ask a question about lifecycle of the product by Rule 1
3:    Collect answer about lifecycle
4:    Repeat
5:        Ask a domain question on environment by Rule 2
6:        Collect answer for the domain question
7:        Analyze answer by generic questions
8:    Until all the environments are defined
9:    Update PES
10:   Repeat
11:       Ask a domain question on relation by Rule 3
12:   Until all the relations are defined
13:   Update and output PES
```

4.3 Evaluation of the Question Asking Approach based on Environment-based Requirements Roadmap: a Case Study

An example of requirements elicitation process for an energy trading software system is performed to illustrate the question asking approach. A software prototype called Question Asking has been developed based on the question generation rules and algorithms presented in the previous section. Another prototype called ROM2PES is used for transforming ROM diagram to PES, which connect two types of question generation. The simulation process and the results of case study are used to evaluate the proposed work.

4.3.1 Energy trading software system background

The following paragraph, provided by an energy trading company, describes their business activities. The aim of this project is to help the company to identify the requirements for the new system starting.

Energy trading is the activity involving trading energy related commodities, such as power, natural gas, crude oil, and refined products like fuel oil, heat oil, gasoline etc. Energy is not only a consumer product, but also an investment product. As a consumer product, energy producers need to know existing demand, potential demand, and existing supply and potential supply; as an investment product, investment institutions need to know the return and risk of the investment. Given the huge demand of energy and big energy price volatility, an automation system is the only choice to manage the energy trading.

From the original description, a more detailed PES system can be completed through a series of question asking process. The PES includes all the possible environments and relations among product and environments, which assists the requirements identification.

4.3.2 Question asking process for energy trading system

First of all, a ROM diagram for the original requirements description is generated, which is illustrated in Figure 4-7. The ROM diagram is the input for the question generation.

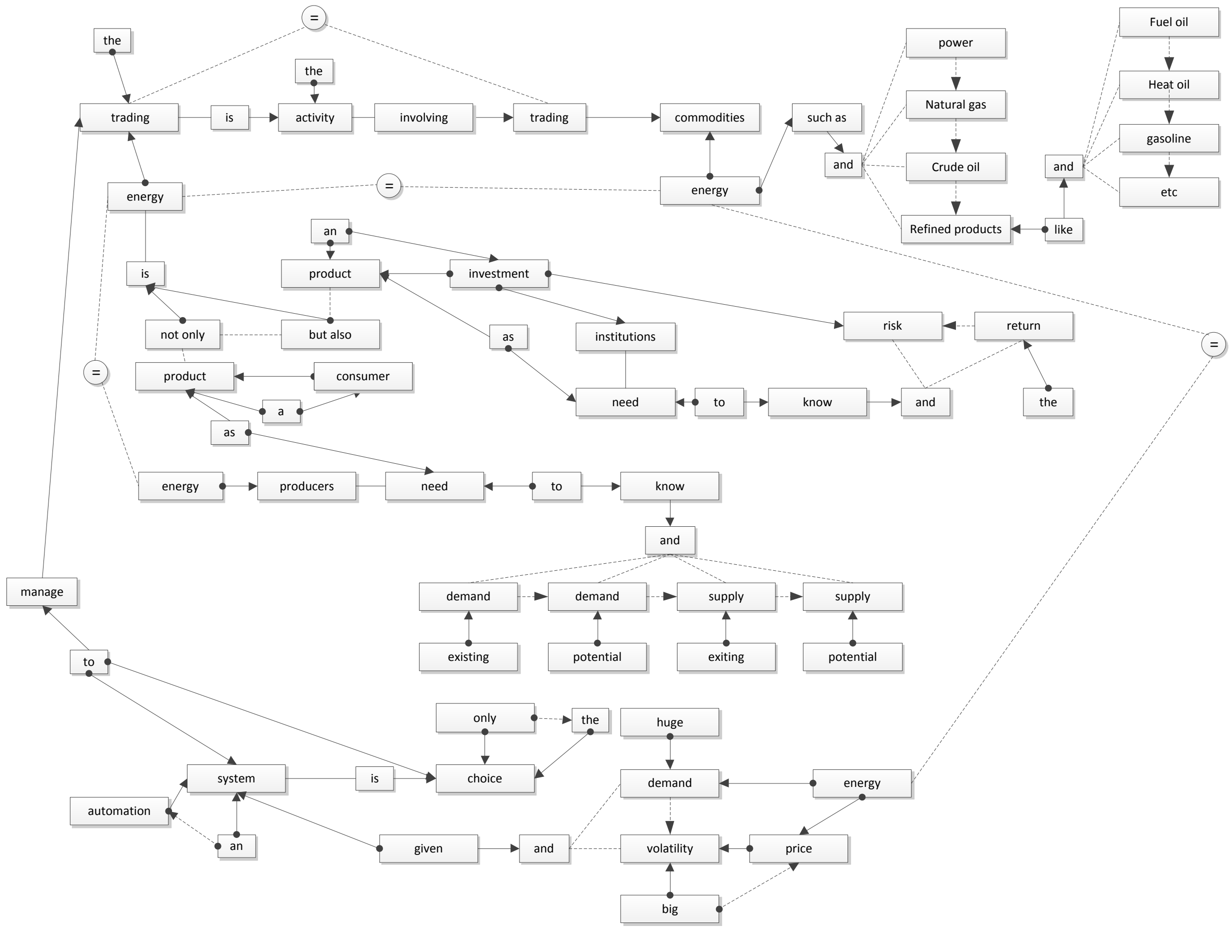


Figure 4-7 The ROM diagram for original requirements

The original PES system based on the given description is identified as Figure 4-8.

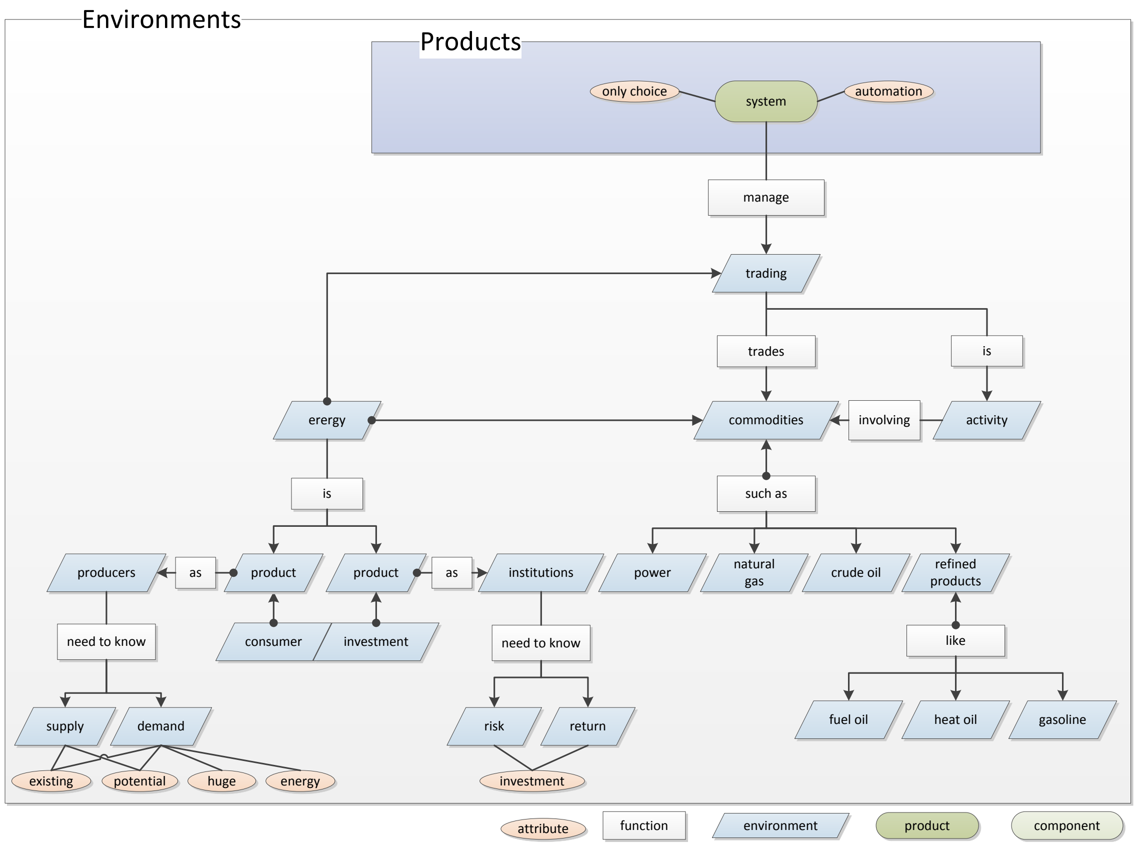


Figure 4-8 The PES of automation system

In this case, all the objects are identified and the numbers of relations on each object in the ROM diagram are calculated by the software prototype. The meaningful noun objects and their relation numbers are listed in Table 4-11. Also the constraining objects of each noun object are listed in the table.

Table 4-11 Object analysis

Object order	Object	Number of constrain relations	Constraining object(s)
1	trading	5	energy, is activity, involving, manage, commodities
47	system	3	manage trading, automation, is choice, is given demand and volatility
3	activity	2	is, involving
6	commodities	2	energy, is traded
7	energy	2	is product, such as
21	product	2	for, consumer
23	product	2	investment, for
30	institutions	2	investment, need
50	choice	2	only, is
55	demand	2	huge, energy
56	volatility	2	big, price
38	demand	1	existing
40	demand	1	potential
42	supply	1	existing
44	supply	1	potential
57	price	1	energy

36	risk	1	investment

Below are the examples that illustrate how the algorithm is implemented according to the ranking of noun objects shown in Table 4-11.

According to Rule1 “center object is the (noun or verb form noun) object with the most relations of predicate and constrain”, the center object should be the entry object for analyzing a text.

In this case, object 1 “trading” is the center object with five predicate and constrain relations. Object 7 “energy” constrains the center object; objects that have predicate relation with it are object 2 “is”, object 48 “manage”, object 4 “involving”, and object 6 “commodities”.

Based on Rule2 “before an object can be further defined, the objects constraining and/or predicating them should be refined”, for defining center object “trading”, other five objects (“energy”, “is”, “involving”, “trading”, “commodities”) and/or their constraining and predicating objects should be pre-defined.

Therefore, for object “energy”, its constraints include “such as” and “like”. No questions to ask according to Rule 2.1 (preposition) and Rule 2.3 (and/or). But the objects related to them should be defined, so the questions will be asked from Question 1 to 8 as in the Table 4-12. Then predicate of “energy” is “is” and its consequence. Questions will be asked from Question 9 to 37 to refine “energy”. During the question asking process, questions for a verb object follow the question template in Table 4-5. For example the Question 17 to 20 are for verb “need to”.

After “energy” is refined, questions go back to “commodities” as Question 38. Then other questions according to the sequence of object list will be generated until all the objects in the list are refined.

Table 4-12 List of the first round questions

Question Number	Related Object(s)	Question
1	energy	What is power?
2	energy	What is natural gas?
3	energy	What is crude oil?
4	energy	What is refined product?
5	energy	What is gasoline?
6	energy	What is heat oil?
7	energy	What is fuel oil?
8	energy	What is energy?
9	Energy is	Who is the consumer?
10	Energy is	What is a consumer product?
11	Energy is	What are energy producers?
12	Energy is	How is energy a consumer product?
13	consumer	What is the existing demand?
14	consumer	What is the potential demand?
15	consumer	What is the existing supply?
16	consumer	What is the potential supply?
17	need to	Why do energy producers need to know existing and potential demand and supply?
18	need to	When do energy producers need to know existing and potential demand and supply?
19	know	How do energy producers know existing and potential demand and supply?

20	know	Where do energy producers know existing and potential demand and supply from?
21	product	What is investment?
22	product	What are investment institutions?
23	product	What is the investment return for an energy product?
24	product	What is investment risk for an energy product?
25	is	How is energy an investment product?
26	need to	Why do investment institutions need to know investment return and risk for an energy product?
27	need to	How do investment institutions need to know investment return and risk for an energy product?
28	need to	Where do investment institutions need to know investment return and risk for an energy product from?
29	given	What is huge energy demand?
30	huge	How huge is energy demand?
31	given	What is energy price volatility?
32	big	How big is volatility of the energy price?
33	given	Who (or what) give the huge energy demand and energy price volatility?
34	given	Why do they give the huge energy demand and energy price volatility?
35	given	When do they give the huge energy demand and energy price volatility?
36	given	How to give the huge energy demand and energy price volatility?
37	given	Where to give the huge energy demand and energy price volatility?
38	commodities	What are energy-related commodities?
39	trading	Who trades energy-related commodities?
40	trading	Why do they trade energy-related commodities?
41	trading	How do they trade energy-related commodities?
42	trading	When do they trade energy-related commodities?
43	trading	Where do they trade energy-related commodities?

44	system	What is trading?
45	manage	What is energy trading?
46	activity	What activity is involving trading energy related commodities?
47	involving	How is energy trading an activity involving trading energy related commodities?
48	system	What is automation? What is an automation system?
49	system	What is an automation system?
50	is	What is a choice? (What are choices to manage energy trading?)
51	is	How is an automation system a choice to manage energy trading?
52	manage	Why does an automation system manage energy trading?
53	manage	When does an automation system manage energy trading?
54	manage	Where does an automation system manage energy trading?

The answers of these questions are collected from all resources and analyzed as the similar procedures as original requirements, which are addressed in the project report (Wang, 2012). Afterwards, the system requirements are more clear and detailed.

According to the updated requirements based on the first round of answers mainly given by the company, the product of this case is the “automation system”; its main function is “managing energy trading”. To collect domain related requirements, the second round of domain questions about lifecycle in energy trading and environments are generated and the answers are collected from some resources mainly from customer.

The questions and answers are listed in Table 4-13. The first question is about life cycle: What is the life cycle of energy trading? After the stages of life cycle are identified, questions about environments of each stage will be asked. For example, Questions 2 to 8 in Table 4-13 are for transaction capture process.

Table 4-13 Questions and answers about life cycle

#	Question	Answer
1	What is the life cycle of energy trading?	The life cycle of managing energy trading includes Transaction Capture, Pricing Feeds, Contract Management, Risk Management, Operations and Nominations, Invoicing/Accounts Payable/Accounts Receivable, PnL Analysis/Reporting, and Management Reporting/Decision.
2	Who (stakeholders) will involve in transaction capture process? (Such as who are the client/ customer/users/other stakeholders of a current system?)	Traders, marketers, and operation managers.
3	What are the roles of different stakeholders in transaction capture process?	Traders will bid the market and make decisions to execute the transaction.
4	What are the activities for different stakeholders in transaction capture process?	Traders enter transactions.
5	What relationships (or collaboration) are there between the stakeholders in transaction capture process with the stakeholders in other events such as pricing feeds?	Each stakeholder is playing individual important role in the whole trading business. They will collaborate to make sure all information is complete and accurate.
6	What are the existing business processes for transaction capture, including the manual and automated processes?	Transactions will be manually booked in the beginning in different systems or spreadsheets, and then be consolidated into one automation system.
7	Could you give a scenario(s)	For example, Trader Jack is trading WTI crude

	that may happen in transaction capture process if possible?	oil in CME at Chicago. Jack will go to CME online system to enter the transaction, then through CME gateway, the transaction will be imported into the in-house system used by Jack. From then on, Jack would track and analyze the transaction in the in-house system.
8	What do you want to change or improve for transaction capture in new system?	User friendly;
9	Who (stakeholders) will involve in pricing feeds process? (Who are the client/ customer/users/other stakeholders of a current system?)	Product Control managers and Settlement managers.
10	What are the roles of different stakeholders in pricing feeds process?	Product Control managers to make sure correct products are used, and make sure the market data is used properly for each product.
11	What are the activities for different stakeholders in pricing feeds process?	Product Control managers will load and verify market data for each energy product.
12	What relationships (or collaboration) are there between the stakeholders in pricing feeds process with the stakeholders in other events such as contract management?	Each stakeholder is playing individual important role in the whole trading business. They will collaborate to make sure all information is complete and accurate.
13	What are the existing business processes for pricing feeds, including the manual and automated processes?	Stakeholders will look up the market data and settlement prices from the market. Then, those data will either be manually entered into the energy trading system, or through other pricing feeding system.
14	Could you give a scenario(s) that may happen in pricing feeds process if possible?	For example, one transaction is trading WTI for September 2012 contract month. As of August 20, 2012, the price for September 2012 WTI is known. On August 20, 2012, Settlement managers would go to CME website or other market data source to look up the price. Then that price will be manually entered into the in-house energy trading system.
15	What do you want to change	User friendly;

	or improve for pricing feeds in new system?	
16	Who (stakeholders) will involve in contract management process? (Who are the client/ customer/users/other stakeholders of a current system?)	Contract managers will involve in contract management process.
17	What are the roles of different stakeholders in contract management process?	Contract mangers are responsible for managing the contract.
18	What are the activities for different stakeholders in contract management process?	Contract managers will manage company, contract and confirmation.
19	What relationships (or collaboration) are there between the stakeholders in contract management process with the stakeholders in other events such as pricing feeds?	Each stakeholder is playing individual important role in the whole trading business. They will collaborate to make sure all information is complete and accurate.
20	What are the existing business processes for contract management, including the manual and automated processes?	Company and contract information will be entered into system manually, the confirm process will be automated by system.
21	Could you give a scenario(s) that may happen in contract management process if possible?	Contract manager will set up the company profile, including company name, address, contact and credit related information, then the contract with the company will be set up, and all transactions tied to the contract would be verified and confirmed.
22	What do you want to change or improve for contract management in new system?	Automate most of the processes.
23	Who (stakeholders) will involve in risk management process? (Who are the client/ customer/users/other stakeholders of a current	Risk managers will involve in risk management process.

	system?)	
24	What are the roles of different stakeholders in risk management process?	Risk managers will be responsible for managing energy trading related risk limits.
25	What are the activities for different stakeholders in risk management process?	Risk managers will use different scenarios to manage market risk, credit risk and operational risk.
26	What relationships (or collaboration) are there between the stakeholders in risk management process with the stakeholders in other events such as contract management?	Each stakeholder is playing individual important role in the whole trading business. They will collaborate to make sure all information is complete and accurate.
27	What are the existing business processes for risk management, including the manual and automated processes?	Most risk related processes have been automated. Scenarios will be set up, then each scenario will be processed by system, the output for the system will be used to help manage all risk limits.
28	Could you give a scenario(s) that may happen in risk management process if possible?	For example, given the volatility of the crude market and the regulation, extra capita is set aside to make sure the worst loss would be covered. So the worst loss is the market risk limit and would be calculated by the system. Commonly, VaR (value-at-risk) would be used to monitor the market risk. Risk managers would decide the scenario settings for the VaR, then the system would calculate the VaR covering all energy trading business. The result will be monitored closely to make sure the VaR is within the business limit. If there is limit breach, investigation is required to understand the mechanics.
29	What do you want to change or improve for risk management in new system?	User friendly;
30	Who (stakeholders) will involve in operations and nominations process? Who are the client/ customer/users/other	Operation mangers will involve in operations and nominations process.

	stakeholders of a current system?)	
31	What are the roles of different stakeholders in operations and nominations process?	Operation managers are responsible for operations and nominations.
32	What are the activities for different stakeholders in operations and nominations process?	Operation managers will verify trading product being setup correctly, and will be responsible for scheduling and nomination for physical delivery.
33	What relationships (or collaboration) are there between the stakeholders in operations and nominations process with the stakeholders in other events such as contract management?	Each stakeholder is playing individual important role in the whole trading business. They will collaborate to make sure all information is complete and accurate.
34	What are the existing business processes for operations and nominations, including the manual and automated processes?	After transaction is booked into system, operation managers will valid the transaction to make sure the information is complete and accurate. For physical delivery, operation managers will nominate the delivery volume with counterparty or pipeline operator.
35	Could you give a scenario(s) that may happen in operations and nominations process if possible?	For example, a purchase contract has been assigned with counterparty to deliver natural gas at City-Gate in Chicago through TransCanada Pipeline Inc. After the transaction is entered into the energy trading system, operation managers will check the details of the transaction to make sure the delivery date, delivery volume and pricing are correct according to the contract. Then when the delivery time comes, operation managers will schedule the delivery volume with TransCanada Pipeline Inc, and notify the counterparty the time and location to receive the natural gas.
36	What do you want to change or improve for operations and nominations in new system?	User friendly;
37	Who (stakeholders) will involve in Invoicing/Accounts	Accountants will involve in this process.

	Payable/Accounts Receivable process? (Who are the client/ customer/users/other stakeholders of a current system?)	
38	What are the roles of different stakeholders in Invoicing/Accounts Payable/Accounts Receivable process?	Accountants will be responsible for all aspects of this process.
39	What are the activities for different stakeholders in Invoicing/Accounts Payable/Accounts Receivable process?	Accountants enter settled prices for each traded product, generate invoice and verify invoice with counterparty statements. General ledgers will be maintained and all accounting related process and reporting would be managed properly.
40	What relationships (or collaboration) are there between the stakeholders in Invoicing/Accounts Payable/Accounts Receivable process with the stakeholders in other events such as contract management?	Each stakeholder is playing individual important role in the whole trading business. They will collaborate to make sure all information is complete and accurate.
41	What are the existing business processes for Invoicing/Accounts Payable/Accounts Receivable, including the manual and automated processes?	Once the prices are fed into energy trading system and transactions are verified, accountants will generate invoice statements. After reconciling the statements from counterparties, the invoices will be delivered to counterparties. At the same time, the invoice statements will be used to generate General Ledger, financial statements, financial reporting and financial analysis.
42	Could you give a scenario(s) that may happen in Invoicing/Accounts Payable/Accounts Receivable process if possible?	For example, transaction was entered for a WTI crude oil fix-float swap to pay fixed price of $95 USD per barrel to receive Last Day June 2012 WTI future contract. The invoice payment date for this transaction was June 5, 2012. The Last Day price would be known on May 20, 2012, it was $97 USD per barrel. In this case, the settlement price of $97 USD per

		barrel would be collected from market and entered into the energy trading system on May 20, 2012. An invoice statement would be generated and sent out before June 5, 2012 to counterparty to pay $2 USD per barrel and the invoice due date would be June 5, 2012. The invoice would be posted as General Ledger entry and would be used for financial analysis, and financial reporting.
43	What do you want to change or improve for Invoicing/Accounts Payable/Accounts Receivable in new system?	User friendly;
44	Who (stakeholders) will involve in PnL Analysis/Reporting process? (Who are the client/ customer/users/other stakeholders of a current system?)	Product Control managers will involve in PnL Anlysis/Reporting process.
45	What are the roles of different stakeholders in PnL Analysis/Reporting process?	Product Control managers are responsible for PnL Analysis/Reporting.
46	What are the activities for different stakeholders in PnL Analysis/Reporting process?	Product Control managers would assure correct products and correct models are used to value the trading business. All historical cash and forward value are monitored and analyzed closely to provide accurate PnL (Profit and Loss). At the same time, the PnL would be explained properly.
47	What relationships (or collaboration) are there between the stakeholders in PnL Analysis/Reporting process with the stakeholders in other events such as contract management?	Product Control managers would work very closely to operation group to make sure correct products be traded. Product Control managers will compare the PnL with trading desk's estimate. If there is discrepancy, a detailed explanation should be provided.
48	What are the existing business processes for PnL Analysis/Reporting, including	Most of PnL Analysis/Reporting functions are automated. Given the complexity of this process, lots of spreadsheets will be used to

	the manual and automated processes?	help the analysis. Daily PnL will be generated by the energy trading system, then the PnL will be explained by different categories based on market factors.
49	Could you give a scenario(s) that may happen in PnL Analysis/Reporting process if possible?	For example, PnL is $1,000,000 between 2 business days. PnL analysis is required to find out which market factors contributed the PnL, and how much contribution for each factor. Commodity price, interest rate, currency exchange rate, volatility and time are the most important market factors to impact PnL analysis.
50	What do you want to change or improve for PnL Analysis/Reporting in new system?	Better models for PnL analysis;
51	Who (stakeholders) will involve in Management Reporting/Decision process? (Who are the client/ customer/users/other stakeholders of a current system?)	Senior Executives will involve in Manager Report/Decision process.
52	What are the roles of different stakeholders in Management Reporting/Decision process?	Senior Executives will be responsible for Management Reporting/Decision process.
53	What are the activities for different stakeholders in Management Reporting/Decision process?	Senior Executives will monitor the PnL, all risk limits, they will make decisions on business model and risk tolerance.
54	What relationships (or collaboration) are there between the stakeholders in Management Reporting/Decision process with the stakeholders in other events such as contract management?	Senior Executives will monitor all other stakeholders to make sure each segment of the business has been implemented properly.
55	What are the existing business processes for Management Reporting/Decision, including	PnL and risk limits will be reported to Senior Executives on daily basis. Senior Executives will work very closely to marketing group to

	the manual and automated processes?	validate business models against current and future market conditions. Based on the information, decisions would be made to carry on current business strategy and business model, or change them.
56	Could you give a scenario(s) that may happen in Management Reporting/Decision process if possible?	For example, natural gas prices have been subdued in last few years due to technology breakthrough in shale gas . In the meantime, the crude oil price stays high due to tighter supply to meet demand. After analyze the whole business book, a decision would be made to trade less natural gas in short term. The team would spend more effort in crude oil marketing and price discovery.
57	What do you want to change or improve for Management Reporting/Decision in new system?	User friendly;

The case study in this chapter only illustrates the question asking process for collecting more complete and necessary requirements. The outcomes of the project are a few conceptual models of Use Case diagram, state diagram, class diagram, and architecture diagram, which are addressed in the project report (Wang, 2012). The questions generation process is automatically accomplished, which is the contribution, comparing with traditional experienced-based brainstorming process.

Chapter 5 Structure of Conceptual Models

This chapter analyzes the general structure of conceptual models and takes PES model and FBS model for particular instances to find out possibility of transformation from one to another.

5.1 **Conceptual Models**

A conceptual model is high-level abstraction that describes what people can do with the application or service and what concepts they need to understand in order to use it. Specifically, conceptual models can be used in the following aspects: (1) describe structure models in terms of entities, relationships, and constraints; (2) describe behavior or functional models in terms of states, transitions among states, and actions performed in states and transitions; and (3) describe interactions and user interfaces in terms of messages sent and received and information exchanged. Such as in software development, a conceptual model enable clients and analysts to understand one another, enable analysts to communicate successfully with application programmers, and in some cases automatically generate parts of the software application.

Several conceptual models are widely used in different engineering fields, such as Use Case Model, Domain Model, Entity-Relationship (ER) Model, Function-Behaviour-State (FBS) model, and Product-Environment System (PES). Those conceptual models specify and describe concepts and relationships between these concepts. For example, in a class diagram, classes represent concepts, associations represent relationships between concepts and role types of an association represent role types taken by instances of the modelled concepts in various situations. In ER notation, entities represent concepts, cardinality and optionality represent relationships between concepts. In PES, product, product components and their attributes describe the system to be developed, environments and their attributes describe the outside of the system, whereas the relationships represent the interactions between system and outside. Regardless of the different notations, those conceptual models have same composition with the concepts and relationships between these concepts. As well the concepts can be decomposed into several primitive ones. Any conceptual can be formulated as Eq. (5-1), where S denotes a conceptual model; E_i and E_j are primitive concepts; $E_i \otimes E_j$ is the relationship between E_i and E_j.

$$\oplus S = \oplus(\bigcup_{i=1}^{n} E_i) = \bigcup_{i=1}^{n}(\oplus E_i) \cup \bigcup_{i=1}^{n}\bigcup_{\substack{j=1 \\ i \neq j}}^{n}(E_i \otimes E_j) \tag{5-1}$$

5.2 **Product-Environment System (PES)**

During the design process, design description keep evolving from informal and unstructured to more formal and structured representations. However, as was indicated in (Chen and Zeng, 2006; Zeng and Gu, 1999), each design state embodies both design problem and design solutions. At any stage of design, all the design information is included in the structure of the A Product-Environment System (PES). The PES reflects the product, environments and relations between environments and product for a design problem.

A PES is defined as the structure of an object (Ω) including both a product (S) and its environment (E).

$$\oplus\Omega = \oplus(E \cup S) = (\oplus E) \cup (\oplus S) \cup (E \otimes S) \cup (S \otimes E), \forall E, S[E \cap S = \Phi], \tag{5-2}$$

where ⊕E and ⊕S are structures of the environment and product, respectively; E⊗S and S⊗E are the interactions between environment and product. A PES is illustrated in Figure 3-3.

Corresponding to the subjective and objective realms adopted by Erden et al (Erden et al., 2008), we can divide the environment E into subjective and objective environments. The subjective environment, denoted by E_s, includes the users of the product whereas the objective environment, denoted by E_o, includes all of the other environment components that have an impact on the behaviour of the product. Therefore,

$$\begin{aligned} \oplus\,\Omega &= \oplus(E_s \cup E_o \cup S) \\ &\subset \oplus(E_s \cup \oplus(E_o \cup S)) \\ &= (\oplus E_s) \cup (\oplus(E_o \cup S)) \cup (E_s \otimes (\oplus(E_o \otimes S)) \cup (\oplus(E_o \cup S) \otimes E_s) \\ &= E_s \cup (\oplus(E_o \cup S)) \cup (E_s \otimes (\oplus(E_o \otimes S)) \cup (\oplus(E_o \cup S) \otimes E_s). \end{aligned} \tag{5-3}$$

Eq. (5-3) is illustrated in Figure 5-1.

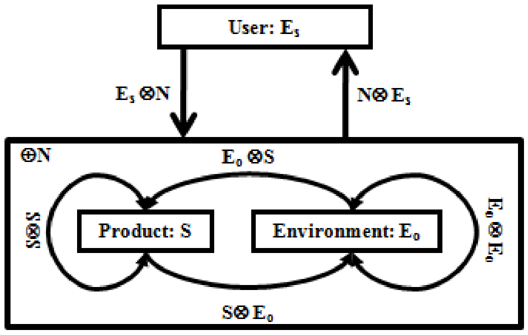


Figure 5-1 Product-environment system from subjective and objective perspective (Wang et al., 2013)

5.3 **FBS Model**

The FBS model is a hierarchical knowledge representation scheme that defines a function as an association between human intention and behaviour (Umeda and Tomiyama, 1995). The FBS model includes functions, behaviours, states, and physical phenomena. In this research, we study the FBS model based on Umeda and Tomiyama's work. FBS modeling includes three parts: representation of function, FBS diagram, causal decomposition and task decomposition (Umeda and Tomiyama, 1995). In order to represent function, the concepts of F-B relationship, state, behaviour, physical phenomena and aspect are introduced. A FBS diagram is used to distinguish between the subjective part and the objective part of a design object, to represent a function as an association of subjective concepts and objective concepts rather than just either of them, and to represent a design object hierarchically in order to support a modeling process that details functional and behavioural descriptions concurrently. Based on the FBS diagram, two approaches were proposed for functional decomposition: causal and task decompositions (Umeda and Tomiyama, 1995).

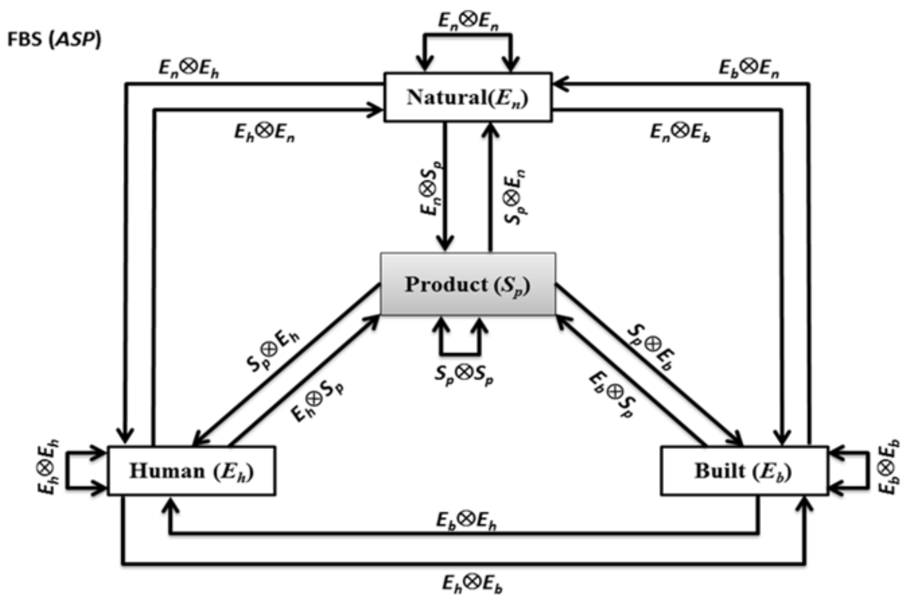


Figure 5-2 ATDM perspective of FBS model (Wang et al., 2013)

In this subsection, we will reformulate FBS using the ATDM theory. As will be shown later in this thesis, this reformulation will be the foundation for the development of an algorithm to transform a design text into a FBS model.

In FBS modeling, Umeda and Tomiyama define a function as “a description of behaviour recognized by a human through abstraction in order to utilize it” (Umeda and Tomiyama, 1995). The ROM diagram of this definition is shown in Figure 5-3, which reveals the relation between function, behavior and human. This relation is formally represented in Eq. (5-4)

$$F \subseteq E_h \otimes B, \tag{5-4}$$

where F denotes function, Eh is human environment, and B is behavior. Eq. (5-4) implies that the function (F) can be represented as a human perception or abstraction of behaviour.

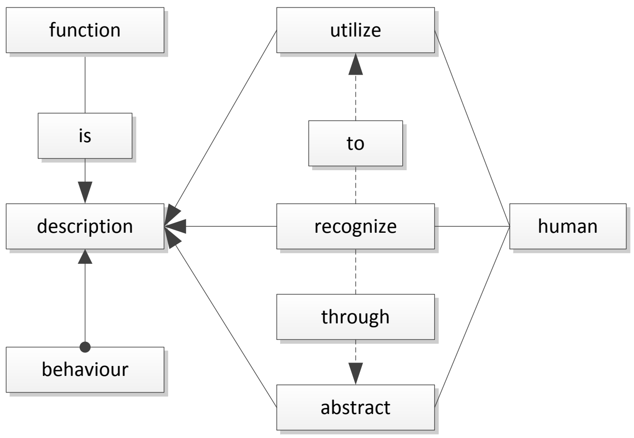


Figure 5-3 The ROM diagram for the definition of function (Wang et al., 2013)

In order to define behavior, the concept of state is introduced. "A state is represented as S(E, A, R), where E denotes identifiers of entities included in this state; A denotes attributes of entities; R denotes relations in the state that includes relations among entities, between entities and attributes, and among attributes" (Umeda and Tomiyama, 1995). This statement can be represented using Axiomatic Theory of Design Modeling (Zeng, 2002). The state S(E, A, R) is represented by Eq. (5-5) and the relation R is denoted in Eq. (5-6).

$$S = E \cup A \cup R, \tag{5-5}$$

$$R = (E \otimes E) \cup (E \otimes A) \cup (A \otimes E) \cup (A \otimes A). \tag{5-6}$$

Substituting R in Eq. (5-6) into Eq. (5-5), we get

$$S = E \cup A \cup (E \otimes E) \cup (E \otimes A) \cup (A \otimes E) \cup (A \otimes A) = \oplus(E \cup A). \tag{5-7}$$

Since A denotes attributes of entities (E), E can be seen as a part of A, i.e. $E \subset A$,

$$S = \oplus A. \tag{5-8}$$

"Behaviour is defined by sequential one and more changes of states over time. Behaviour b is represented as (s0, t0), (s1, t1), … , (sn, tn) ($n \geq 0$; si $\in$ S, ti $\in$ T), where S and T denote a set of states and an ordered set of time respectively" (Umeda and Tomiyama, 1995). Therefore, behaviour is a kind of relation from one state to another.

$$S_t = S \otimes T, \tag{5-9}$$

$$b \subset B. \tag{5-10}$$

"A physical phenomenon PP causes a state transition from (si, ti) to (sj, tj) ($i \leq j$), where s, represents the required condition for activating this phenomenon" (Umeda and Tomiyama, 1995). We use Et to denote the environment and Sp to denote the product in a product system. Then the physical phenomenon PP is a kind of relation from environment to product as shown in Eq. (5-11).

$$pp \subseteq E_t \otimes S_p. \tag{5-11}$$

Thus behaviour b can be described by its initial state (s_0, t_0) and a set of physical phenomena PP.

"An aspect ASP is defined as ASP=(E, A, R, PP, T), where E, A, R, PP ,and T denotes sets of all entities, attributes, relations, physical phenomena and time of the current interest respectively" (Umeda and Tomiyama, 1995). Aspect ASP can be represented by Eq. (5-12), which is the structure of the product system. Therefore, aspect is a kind of description of product system which consists of product, environment and relations.

$$ASP = (S_p \cup E_t \cup R) \subseteq \oplus(S_p \cup E_t). \tag{5-12}$$

By decomposing the product structure, behaviour B can be divided into a series of primitive behaviours, which can be represented as Eq. (5-13).

$$B \supset B_1 \cup B_2 \cup \ldots \cup B_n. \tag{5-13}$$

Therefore, Eq. (5-4) can be expanded as Eq. (5-14).

$$\begin{aligned} F &\subseteq E_h \otimes B \\ &\subset E_h \otimes ((\oplus B_1) \cup (\oplus B_2) \cup \ldots \cup (\oplus B_n) \cup (B_1 \otimes B_2) \cup \ldots \cup (B_m \otimes B_n)) \\ &= F_1 \cup F_2 \cup \ldots \cup F_n \cup (F_1 \otimes F_2) \cup \ldots \cup (F_m \otimes F_n). \end{aligned} \tag{5-14}$$

From the representations of behaviour and state in Eq. (5-9) and Eq. (5-10), a state is the relation of the structure of attributes within the state to time, and behaviour is the change of states over time. Hence, a function could also be decomposed by time. If the decomposition is by time, it is the causal decomposition; otherwise, it is task decomposition.

Table 5-1 summarizes the representations of FBS corresponding to the product-environment system. This correspondence provides the mathematical foundation for the transformation from ROM to FBS, since the ROM diagram for a design text implies a product-environment system.

Table 5-1 Representation of FBS system

FBS	**Product-Environment System**	**Mathematical Representation**
Product: S_p	Structure of entities and attributes	$\oplus (E \cup A)$
State: S_t	Structure of entity attributes at time t	$(\oplus A) \otimes T$
Behavior: B	Relation of one state to another	$S_t \otimes S_t$
Function: F	Relation of human to behavior	$E_h \otimes B$
Physical phenomena: PP	Relation of environment to product	$E_n \otimes S_p$
Aspect: ASP	Product-Environment system	$\oplus (S_p \cup E_n)$

Chapter 6
Formalization of Transformation from Requirements Text to Conceptual Models

After collecting complete requirements, a transformation of design requirements into conceptual models is needed for design in a product development process. The third objective of this PhD research is to develop a general approach to transforming unrestricted natural language based requirement text into structured conceptual models, such as FBS model and domain model by using a formal design method – Environment Based Design (EBD).

6.1 Transformation from Requirements Text to a Conceptual Model

Figure 6-1 shows the transformation process from a requirements text to a conceptual model. This process can be divided into two sub-processes: first, the requirements text described in natural language will go through a linguistic analysis process using the computer tool ROMA, which generates a ROM diagram for the requirements text; then, another transformation process transforms the ROM diagram into a conceptual model. Since the first process has been dealt with elsewhere (Zeng, 2008), this research focuses on the second process. Therefore, the input of this transformation is the ROM diagram corresponding to a requirements text whereas the output is a conceptual model.

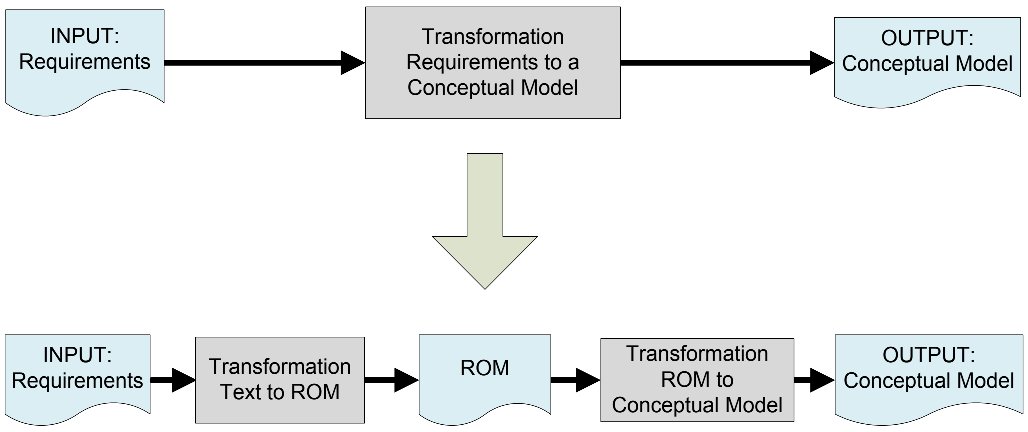

INPUT: Requirements
Transformation Requirements to a Conceptual Model
OUTPUT: Conceptual Model
INPUT: Requirements
Transformation Text to ROM
ROM
Transformation ROM to Conceptual Model
OUTPUT: Conceptual Model

Figure 6-1 Transformation from requirements text to FBS model (Wang et al., 2013)

The input of the transformation is ROM, which is introduced in theoretical foundations. In the following sections, the output FBS and the relations between the input and output are analyzed, and accordingly the transition rules are derived. At last, the algorithms are described. The foundation of this discussion is Axiomatic Theory of Design Modeling (ATDM) (Zeng, 2002).

Requirements text includes paragraphs, phrases, and words. Its structure can be modeled by a ROM diagram (Zeng, 2008), which uses five symbols to represent primitive object, compound object, constraint relation, predicate relation and connection relation, as shown in Table 3-1. ROM is effective for representing natural language, whereas it is not convenient for human designers to draw and to manipulate when the diagram becomes big. In our research ROM diagrams are generated by software ROMA which is developed by Design Lab.

The transformation from ROM to a conceptual model is illustrated in Figure 6-2, in which a Product-Environment System is used as an intermediate between ROM and conceptual model.

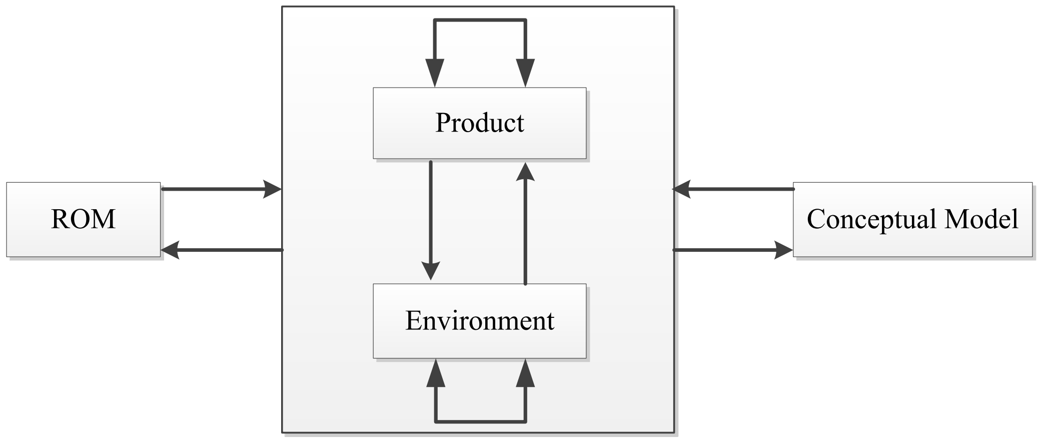


Figure 6-2 Transformation from ROM to a conceptual model

6.2 **Representation of States in Transformation from ROM to Conceptual Model**

In order to define any state in the transformation from ROM to a conceptual model, it is critical to list all of the necessary features for each state. In this research four types features are identified, which are: POS (Part-of-Speech) feature, ROM (Recursive Object Model) feature, PES (Product-Environment System) feature, and CM (Conceptual Model) feature.

Firstly, since each object in a ROM diagram is a word in the design text that needs to be processed, every object in a ROM diagram must have a part of speech (POS). In addition, some of the objects can be further classified according to their linguistic functions. For example, some noun objects describe humans, some other noun objects have their verb counterparts, and some verbs are linking verbs. The POS feature for a transformation state thus includes noun (n), verb (v), adjective (a), adverb (ad), determiner (d), preposition (p), and conjunction (c), together with predefined attributes associated with some semantic functions that the object may carry and are related to the transformation. Secondly, the ROM features for a transformation state are objects, predicate relations, constraint relations and connection relations. Thirdly, the PES features for a transformation state are primitive components included in a product-environment system, which are products, product components, product attributes, product attribute values, environments, environment components, environment attributes, environment attribute values, and relations among them. Finally, FBS features for a transformation state are function, behavior, state, physical phenomena and aspect.

In transforming a ROM diagram to a FBS model, any state may include a combination of the four afore mentioned features: POS, ROM, PES and FBS. Each object in the starting state has both POS and ROM features defined and the other two unknown whereas the ending state is constituted by the objects with all four features defined, Therefore, during the process of transformation from a ROM diagram into a FBS model, an object can be represented as quadruplet of features, which is denoted by f(O) as:

$$f(O) = < ROM(O), POS(O), PES(O), FBS(O) > . \tag{6-1}$$

The aim of the research presented in this chapter is to identify FBS features from ROM and POS features through PES features. Table 6-1 summarizes these six types of object features.

Table 6-1 Object features

Object Features	**Domain**
ROM feature	{object, predicate relation, constraint relation, connection relation, undefined}
POS feature	{noun, verb, adjective, adverb ,determiner, preposition, conjunction, undefined}
PES feature	{product, product component, product attribute, product attribute value, environment, environment component, environment attribute, environment attribute value, relation, undefined}
FBS feature	{function, behavior, state, physical phenomenon, aspect, undefined}
UCD feature	{system, actor, use case, undefined}
DM feature	{class, attribute, association, undefined}

In fact, the category of object and the number of relations associated with each object reflects the role and importance of this object in the ROM diagram. ROM features of an object (*ROM*(*O*)) are a list of relations (*R*) that relate a set of objects (*O*). Each type of objects has relations with other objects in the ROM diagram. For example, a noun object may be constrained by other objects, may constrain other objects, and may have a predicate relation to or from other objects. Any object may connect with other objects of the same POS by conjunctions.

All of the possible relations to a noun object are illustrated in Figure 6-3. The constraint relation to a noun object N from an adjective or noun object B is denoted by $C_s(B,N)$; the constraint relation to a noun object N_4 is denoted by $C_s(N, N_4)$; the constraint relation to noun object N_1 through a preposition object P is denoted by $C_s(N_2,P,N)$; the predicate relation directing from noun object N_3 to N through verb V_1 is denoted by $V_1(N_3, N)$; the predicate relation directing from object N to a noun or adjective object A through verb V_2 is denoted by $V_2(N, A)$; the connection relation between N and noun object N_2 is denoted by $C_n(N, N_2)$. Thereby, the ROM feature of a noun object N can be denoted by

$$\begin{aligned} &\mathrm{ROM}(N) = C_s(B,N) \cup C_s(N,N_4) \cup C_s(N_2,P,N) \cup \\ &V_1(N_3,N) \cup V_2(N,A) \cup C_n(N,N_2). \end{aligned} \tag{6-2}$$

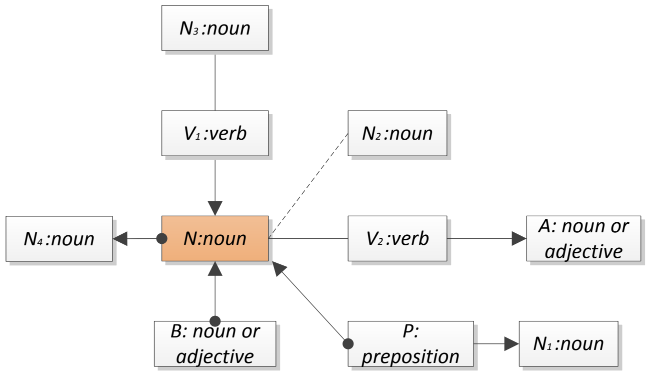


Figure 6-3 ROM feature of a noun object N

As is shown in Figure 6-4, a verb object V may be constrained by an adverb object A, connected with another verb V_1 by conjunction, or has predicate relation directing from a noun object *N* to a noun or adjective object *B*. The ROM feature of a verb object *V* can be denoted by

$$\mathrm{ROM}(V) = C_s(A,V) \cup V(N,B) \cup C_n(V,V_1). \tag{6-3}$$

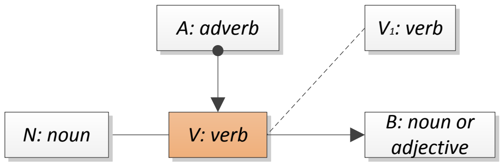


Figure 6-4 ROM feature of a verb object *V*

Meanwhile, the type of verbs in a predicate relation may determine the role of the verbs in the design. The verbs can be categorized into meta verbs (V_m), function verbs (V_f), linking verbs (V_l), and structure verbs (V_s) as shown in Table 6-2. A meta verb relates designers to a product or its working environment; a function verb relates a product to its working environment; a linking verb introduces an attribute of the product; and a structure verb defines the components of a product.

Table 6-2 Verb category

Verb category	**Description**	**Examples**
Meta verbs (V_m)	relates designers to a product system	design, develop
Function verbs (V_f)	relates a product to its environment	support, maintain, raise
Linking verbs (V_l)	introduces a product's properties	be, is, are
Structure verbs (V_s)	defines a product's components	have, include, consist of

An adjective, A_j in Figure 6-5, may constrain a noun object N_2, may be constrained by an adverb A_v, may have a predicate relation directed by a linking verb V_l, or connect with another adjective A_{j1}. The basic ROM diagram for an adjective object is shown in Figure 6-5. The ROM feature of an adjective object A_j can be denoted by

$$\text{ROM}(A_j) = C_s(A_j, N_2) \cup C_s(A_v, A_j) \cup V_l(N_1, A_j) \cup C_n(A_j, A_{j1}). \tag{6-4}$$

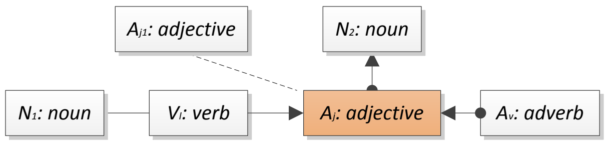


Figure 6-5 ROM feature of an adjective object A_j

Similarly, an adverb A_v may constrain a verb object V or an adjective object A_j, and may be connected with another adverb A_{v1}, as is shown in Figure 6-6. The ROM feature of an adverb object A_v can be denoted by

$$\text{ROM}(A_v) = C_s(A_v, V) \cup C_s(A_v, A_j) \cup C_n(A_v, A_{v1}). \tag{6-5}$$

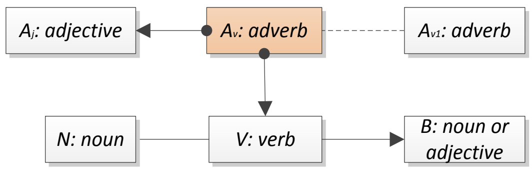


Figure 6-6 ROM feature of an adverb object A_v

A determiner D can only constrain a noun object N as is shown in Figure 6-7. The ROM feature of a determiner object D can be denoted by

$$\mathrm{ROM}(D) = C_s(D, N). \qquad (6\text{-}6)$$

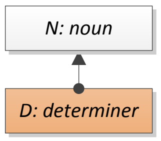


Figure 6-7 ROM feature of a determiner object D

A preposition P may constrain a verb object V or a noun object N_3 from a noun object N_2, as is shown in Figure 6-8. The ROM feature of a preposition object P can be denoted by

$$\mathrm{ROM}(P) = C_s(N_2, P, V) \cup C_s(N_2, P, N_3). \qquad (6\text{-}7)$$

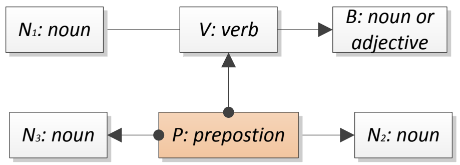


Figure 6-8 ROM feature of a preposition object P

A conjunction C_j can only connect two same types of objects B_1 and B_2 as is shown in Figure 6-9. The ROM feature of a conjunction object C_j can be denoted by

$$\mathrm{ROM}(J) = C_j(B_1, B_2). \quad (6\text{-}8)$$

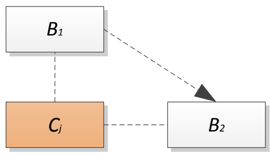


Figure 6-9 ROM feature of a conjunction object C_j

According to the analysis of word features in a ROM diagram, different types of words may play different roles in a product-environment system. For example, a noun object can be a product, a product component, an environment, or an attribute. An adjective or adverb object can be an attribute. A verb object can be an interaction between two other objects. Preposition and conjunction objects connect other PES features into a system. The mappings between POS features, ROM features, PES features, and FBS features are described in Table 6-3.

Table 6-3 Object mappings between POS, ROM, PES and FBS features

POS Feature	**ROM Features**	**PES Features**	**FBS Features**
Noun (N)	$\mathrm{ROM}(N) = C_s(B,N) \cup C_s(N,N_4) \cup C_s(N_2,P,N) \cup V_1(N_3,N) \cup V_2(N,A) \cup C_n(N,N_2).$	Product (P_r) Product Component (C_p) Attribute (A_p) Attribute Value (V_a) Environment (E_p) Environment component (C_e)	State (S)

		Environment Attribute (A_e) Environment Attribute Value (V_{ae})	
Verb (V)	$\mathrm{ROM}(V) = C_s(A,V) \cup V(N,B) \cup C_n(V,V_1).$	Relation (R)	Function (F) Behavior (B)
Adjective (A_j)	$\mathrm{ROM}(A_j) = C_s(A_j,N_2) \cup C_s(A_v,A_j) \cup V_l(N_1,A_j) \cup C_n(A_j,A_{j1}).$	Attribute (A_p/ A_e)	State (S)
Adverb (A_v)	$\mathrm{ROM}(A_v) = C_s(A_v,V) \cup C_s(A_v,A_j) \cup C_n(A_v,A_{v1}).$	Relation (R)	State (S)
Determiner (D)	$\mathrm{ROM}(D) = C_s(D,N).$	Relation (R)	n/a
Preposition (P)	$\mathrm{ROM}(P) = C_s(N_2,P,V) \cup C_s(N_2,P,N_3).$	Relation (R)	n/a
Conjunction (C_j)	$\mathrm{ROM}(J) = C_j(B_1,B_2).$	Relation (R)	n/a

Chapter 7
Algorithm of Transformation from Requirements to FBS

This chapter introduces the algorithm of transformation from requirements text to FBS. According to the traditional understanding, an algorithm is a finite, unambiguous description of an effective procedure for the solution of a class of problems. The procedure in an algorithm is often called a transformation. A transformation is defined by a set of transitions which deal with all the possible cases included in the class of problems for which the algorithm was designed (Davis et al., 1994).

Based on the structure operation, the transformation system (Σ) from a ROM diagram (ROM) to a FBS model (FBS) can be formally represented in Eq. (7-1).

$$\Sigma = \oplus(\mathrm{ROM} \cup \mathrm{FBS}) = (\oplus\mathrm{ROM}) \cup (\oplus\mathrm{FBS}) \cup (\mathrm{ROM} \otimes \mathrm{FBS}) \cup (\mathrm{FBS} \otimes \mathrm{ROM}), \qquad (7\text{-}1)$$

which is illustrated in Figure 7-1.

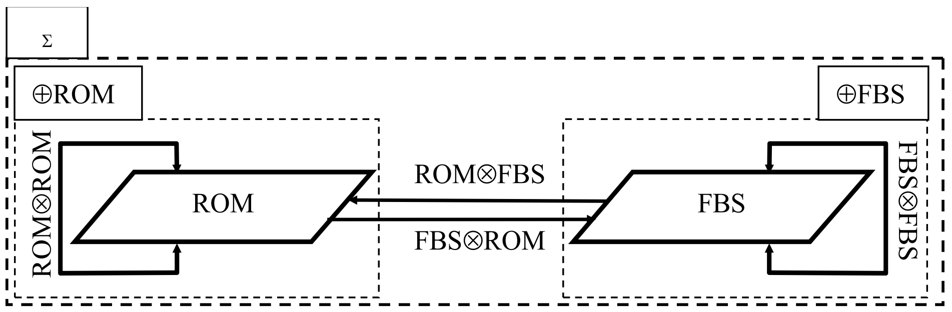


Figure 7-1 Structure of ROM - FBS system (Wang et al., 2013)

The transformation algorithm is part of ROM⊗FBS. In order to develop this algorithm, the structures of the ROM diagram (⊕ROM) and of the FBS (⊕FBS) must first be formalized. ROM is introduced in Chapter 3, whereas FBS has been formulated in Section 5.3. This chapter focuses on the transformation rules and procedures.

7.1 **Transformation Rules from ROM to FBS**

7.1.1 Transformation rules from ROM to PES

The transformation from a ROM diagram to a product-environment system (PES) means to identify the PES features for each object according to its POS and ROM features.

Based on basic ROM and POS features of each type of object summarized in Section 6.2, a ROM diagram with PES features is illustrated in Figure 7-2, which shows all of the possible roles and relations that may be included in a product-environment system (PES). The transition rules from POS and ROM features of a given design text to PES can be derived from this ROM diagram.

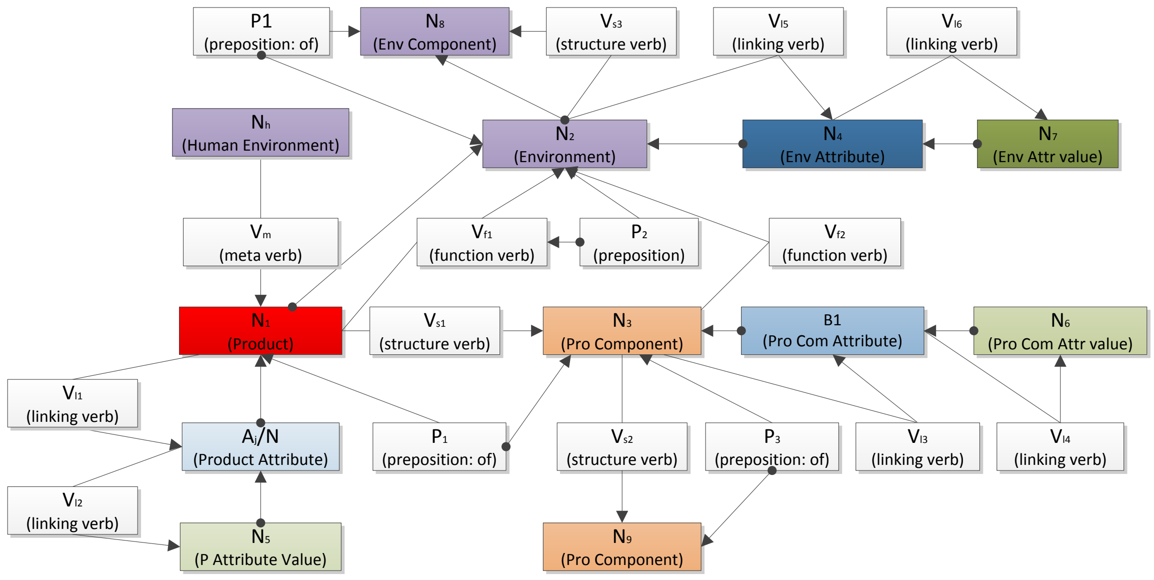


Figure 7-2 ROM and PES features: a complete map

In a ROM diagram, the number of relations associated with each object can be calculated from the diagram. A center object is defined as the object that has the most number of predicate and constrained relations. One ROM diagram may have one or more center objects. In most cases, a center object is a noun object. A center object is important in ROM and is often the starting point for analyzing the ROM diagram. For example, in Figure 6-3, which shows the ROM feature of a noun object N, object N has two predicate relations and two constrained relations, which could affect the semantics of the noun object; therefore, N is the center object. Noun objects play roles in PES such as products, components, attributes, and environments. The center of a PES is product; therefore, determining the product object is a precondition for identifying PES features from a ROM diagram.

Rule1 given in Table 7-1 is used to identify the product object from a ROM diagram. There are two possibilities: 1) the noun object has a predicate relation directed by a meta verb such as "design" and "develop" with human object being its subject; and 2) the noun object is a center object that is related to at least one function verb directed towards other objects, but no function verb directed towards it. After the product object is identified, the PES feature of the product object is updated. Then the components, component values, attributes, and environments can be identified recursively according to related rules listed in Table 7-1. Relations exist among product, product components, component value, and environments through predicate relations and constrain relations.

Table 7-1 Transition rules from ROM to PES

Identify product

Rule 1: If the POS feature of an object *O* in a ROM diagram is noun (*N*), and its ROM feature satisfies one of the following conditions, then this object is a product (P_r):

3) The object has a predicate relation directed from a human (E_h) object through a meta verb (V_m).
4) The object is a center object which has at least one predicate relation directing towards another object (O_x) through a function verb (V_f), but no predicate relation directed towards it from object (O_y) through a function verb.

$$(POS(O)=N)\wedge(ROM(O)=(V_m(E_h,O)\vee(V_f(O,O_x)\wedge\neg V_f(O_y,O)))\rightarrow(PES(O)=P_r),\exists O.$$

Identify product components

Rule 2: If the POS feature of an object *O* in a ROM diagram is noun (*N*) and its ROM feature satisfies one of the following conditions, then this object is a product component (C_p):

4) The object is constrained by a product (P_r) or any other product component (C_p) object and is neither an attribute nor environment.
5) The object has a predicate relation directed from a product or product component through a structure verb (V_s).
6) The object is constrained by a preposition object (*P*) which connects a product or product component.

$$(POS(O)=N)\wedge(ROM(O)=C_s((P_r\cup C_p),O)\vee V_s((P_r\cup C_p),O)\vee C_s((P_r\cup C_p),P,O))\rightarrow PES(O)=C_p,\exists O.$$

Identify product attributes

Rule 3: If the POS of an object *O* in a ROM diagram is noun (*N*) or adjective (A_j), and its ROM features satisfy one of the following conditions, then this object is an attribute (A_p) of a product (P_r) or component (C_p):

3) It constraining a product or a product component and is neither a product nor product component.
4) It has a predicate relation directed from a product or product component through a linking verb (V_l).

$$(POS(O)=N\cup A_j)\wedge(ROM(O)=C_s(O,(P_r\cup C_p))\vee V_l((P_r\cup C_p),O))\rightarrow PES(O)=A_p,\exists O.$$

Identify product attribute value

Rule 4: If the POS feature of an object *O* in a ROM diagram is noun (*N*), adjective (A_j), or adverb (A_v), and the ROM feature satisfies one of the following conditions, then this object is an attribute value (V_a):

3) It constrains an attribute of a product or a product component.
4) It has a predicate relation directed from an attribute object through a linking verb

(V_l).

$$(POS(O) = N \cup A_j \cup A_v) \wedge (ROM(O) = C_s(O, A_p) \vee V_l(A_p, O)) \rightarrow PES(O) = V_a, \exists O.$$

Identify environments

Rule 5: If the POS feature of an object *O* in a ROM diagram is noun (*N*) and its ROM feature satisfies one of the following conditions, then this object is an environment (E_p) of a product (P_r) or product component (C_p):

4) It has a predicate relation directed from a product or product component through a function verb (V_f), but it is not a product, product component, attribute, or attribute value.

5) It has a predicate relation directed from an environment object through a function verb (V_f).

6) It constrains a function object (V_f) of a product or product component through a preposition object (*P*).

$$(POS(O) = N) \wedge (ROM(O) = (V_f((P_r \cup C_p), O) \vee V_f(E_p, O) \vee C_s(O, P, V_f)) \rightarrow PES(O) = E_p, \exists O.$$

Environments can be classified further as the Rule 5.1 to Rule 5.3

Rule 5.1: If an object in a ROM diagram is an environment object and is also a human user or operator in the life cycle of the product, then this object is a human environment.

Rule 5.2: If an object in a ROM diagram is an environment object and is also an artefact built or created by human beings, then this object is a built environment.

Rule 5.3: If an object in a ROM diagram is an environment object but is neither a human nor a built environment, then this object is a natural environment.

Identify environment components

Rule 6: If the POS feature of an object *O* in a ROM diagram is noun (*N*) and its ROM feature satisfies one of the following conditions, then this object is an environment component (C_e):

4) The object is constrained by an environment (E_p) or environment component (C_e), but it is not a product or a product component.

5) The object has a predicate relation directed towards it from an environment object through a structure verb (V_s).

6) The object is constrained by a preposition object (*P*) which connects an environment object.

$$(POS(O) = N) \wedge (ROM(O) = C_s((E_p \cup C_e), O) \vee V_s((E_p \cup C_e), O) \vee C_s((E_p \cup C_e), P, O)) \rightarrow PES(O) = C_e, \exists O.$$

Identify environment attributes

Rule 7: If the POS of an object *O* in a ROM diagram is noun (*N*) or adjective (A_j), and the ROM features satisfy one of the following conditions, then this object is an attribute (A_e) of an environment (E_p):

1) It is constrained by an environment (E_p) or an environment component (C_e).

2) It has a predicate relation directed from an environment or an environment component through a linking verb (V_l).

$$(POS(O) = N \cup A_j) \wedge (ROM(O) = C_s(O,(E_p \cup C_e)) \vee V_l((E_p \cup C_e),O)) \rightarrow PES(O) = A_e, \exists O.$$

Identify environment attribute value

Rule 8: If the POS feature of an object *O* in a ROM diagram is noun (*N*), adjective (A_j), or adverb (A_v) and its ROM feature satisfies one of the following conditions, then this object is an environment attribute value (V_{ae}):
1) It constrains an environment attribute object (A_e).
2) It has a predicate relation directed from an environment attribute object and the predicate verb is a linking verb (V_l).

$$(POS(O) = N \cup A_j \cup A_d) \wedge (ROM(O) = C_s(O,A_e) \vee V_l(A_e,O)) \rightarrow PES(O) = V_{ae}, \exists O.$$

Identify relations

Rule 9: Relations exist among product, product components, and environments. Those relations reflect constraint relation or predicate relations in the ROM diagram.

Some examples given in Table 7-2 demonstrate how the rules in Table 7-1 are applied according to the ROM diagram shown in Figure 7-2.

Table 7-2 Application examples of rules in Table 7-1.

Condition	Result	Rule
"N_h" is human and "V_m" is a meta verb.	"N_1" is a "product".	Rule 1-1)
"N_2" is a center object with eight constraining and predicate relations, but it has a predicate relation directed from "N_1" through "V_{f1}"and another predicate relation from "N_3" through "V_{f2}", so "N_2" is not a product. Similarly the second center object "N_3" is not a product. Then the third center object "N_1" has six constraining and predicate relations.	"N_1" is a "product".	Rule 1-2)
"N_1" is a product, and "N_3" has a relation "$V(N_1,N_3)$".	"N_3" is a component of the product "N_1"	Rule 2-2)
"N_1" is a product, and "N_3" has a relation	"N_3" is a component of the	Rule 2-3)

"$C_s(N_1,P_{of},N_3)$".	product "N_1"	
"$N3$" is product component, and "B_1" has a relation "$C_s(B_1,N_3)$"	"B_1" is an attribute of "N_3"	Rule 3-1)
"$N3$" is product component, and "B_1" has a relation "$V_{13}(B_1,N_3)$"	"B_1" is an attribute of "N_3"	Rule 3-2)
"N_1" is a product, and "N_2" has a relation "$V_{f1}(N_1,N_3)$"	"N_2" is an environment of "N_1"	Rule 5-1)

7.1.2 Transformation rules from PES to FBS

Transformation from product-environment system (PES) to FBS model is the process of identifying FBS features based on PES, ROM and POS features. The components of FBS features include states, functions, behaviors, physical phenomena and aspect.

Transition rules from product-environment system (PES) to a FBS model are shown in Table 7-3. It must be noted that since function and behaviour are generally distinguished only relatively according to the stage of a design (Zeng and Gu, 1999), they are treated by the same rule.

Table 7-3 Transition rules from PES to FBS

Define states
Rule 10: If two objects in a ROM diagram are related in such a way that:1) the first object is a product or a product component, and 2) the second object is the attribute of the product or product component, then both objects together makes an element of a state.
Define behaviours and functions
Rule 11: If two objects in a ROM diagram are related in such a way that: 1) the first object is a function verb directed from a product or product component, 2) the second object is a noun object directed from the first object, then both objects together makes a behaviour or a function.   **Rule 12**: If the product is the noun form of a verb (e.g. *verb*+"er/or"), then one of the main functions is the combination of the following two objects: the first object is the verb form of the product whereas the second object is the closest noun object

constraining the product.
Define physical phenomena
Rule 13: If a combination of three objects in a ROM diagram satisfies the condition that the first object is an environment, the second object is a verb object directed from an environment, and the third object is a product or product component directed from the verb object, then this combination is a physical phenomenon.
Define aspects
Rule 14: Aspect means a whole product environment-system.

7.2 **The transformation Algorithm**

Figure 7-3 shows the framework for the transformation from a design text to a FBS model. First, the design document described in natural language will go through a linguistic analysis process using the computer tool ROMA, which generates the ROM diagram of the design text. Then, the ROM diagram is transformed into the FBS model through another computer tool called R2FBS based on the transition rules introduced in the previous sections. This section will introduce the algorithms transforming a ROM diagram to a FBS model through the product environment system (PES).

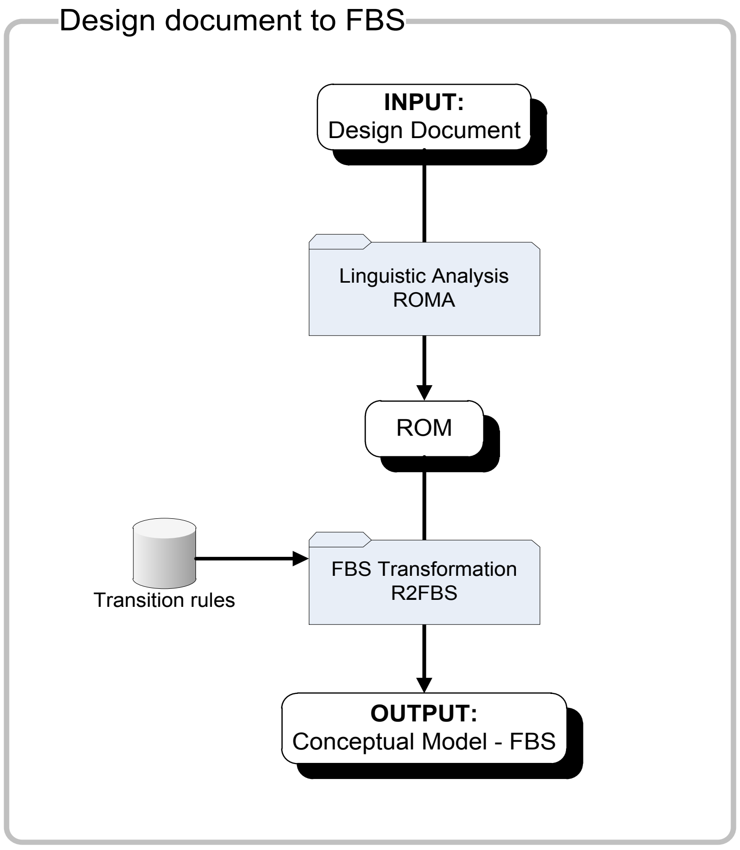


Figure 7-3 Framework for the transformation of a design text into a FBS model

Since all of FBS features come directly from PES features, once PES features are determined, a FBS feature is defined, hence, transformation processes from ROM to PES and from PES to FBS are combined.

The proposed algorithm is shown in Table 7-4. The starting point of this algorithm is to determine the product object in the ROM diagram. Then the product components, product attributes, attribute values, environments, environment components, environment attribute objects and relations among objects can be determined by traversing the ROM diagram while applying the transition rules.

Table 7-4 Transformation algorithm from ROM to FBS

```
Algorithm R2FBS(ROMDiagram: ROM)
1:  Determine the product object from ROM
2:  Repeat
3:      Identify functions and environments
4:      Identify product components and attributes
5:      Identify environments from environments
6:  Until all the noun objects in ROM are defined
7:  Output the FBS model
```

In the algorithm of determining product object shown in Table 7-5, all noun objects in ROM are sorted into a list according to the number of predicate and constraint relations. If the object with most such relations satisfies Rule 1 then the product is determined. Otherwise, the object is deleted from the list and the next object will be decided recursively, until the product object is determined or no product can be found.

Table 7-5 Algorithm of determining product object

```
Algorithm Product(ROMDiagram: ROM)
1:  Sort the noun objects into a list $O_n$ by the number of predicate and
    constraint relations
2:  for all $O \in O_n$
3:      if $O$ satisfies Rule 1 then $Product \leftarrow O$
4:      else delete $O$ from $O_n$
5:      end if
6:  end for
```

In the algorithm of identifying functions and environments shown in Table 7-6, if product S is a noun form of a verb, then its function is the verb form with its closest constraint of S according to Rule 12. If S has a predicate relation directing to a noun object by a function verb, then a function and environment can be identified according to Rule 11. For each identified environment, its related environments can be further identified by calling algorithm of identifying environments from an environment.

Table 7-6 Algorithm of identifying functions and environments

```
Algorithm Function_Environment(Product or Component: S)
1:  if $S$ satisfies Rule 12 then
2:      $Function \leftarrow$ verb form of $S$ + closest constrain of $S$
3:  end if
4:  for all verb $V$ directed from $S$ to noun object $N$
5:      if $V$ satisfies Rule 11, $Environment \leftarrow N$, $Function \leftarrow V + N$
6:  end for
```

In the algorithm of identifying product components and attributes shown in Table 7-7, if product S has a predicate relation directing to a noun by a structure verb, then a product component can be identified according to Rule 2.2). If S has a predicate relation directed to a noun or adjective by a linking verb, then an attribute of the product can be identified according to Rule 3.2). If S has a constraint relation by a preposition, then a product component identified according to Rule 2.3). If S has a constraint relation with a noun object, then a product attribute can be identified according to Rule 3.1). If S constrains another noun object, then a product component can be identified according to Rule 2.1).

For each identified component, its sub functions, environments and attributes then can be determined by related algorithms.

Table 7-7 Algorithm of identifying product components and attributes

```
Algorithm Component_Attribute(Product or Component: S)
1:   for all $N$ which has a predicated relation directed from $S$ by $O_v$
2:       if $O_v$ is a structure verb that satisfies Rule 2.2) then $Component \leftarrow N$
3:   end for
4:   for all adjective $A_j$ which has a predicate relation directed from $S$ by $O_v$
5:       if $O_v$ is a linking verb that satisfies Rule 3.2) then $attribute \leftarrow A_j$
6:   end for
7:   for all $O_c \in O$ which is constraining $S$ through a preposition object
8:       if $O_c$ satisfies Rule 2.3) then $Component \leftarrow O_c$
9:   end for
10:  for all $O_c \in O$ which is constrained by $S$
11:      if $O_c$ satisfy Rule 2.1) then $Component \leftarrow O_c$
12:  end for
13:  for all $O_a \in O$ which is constraining $S$
14:      if $O_a$ satisfy Rule 3.1) then $attribute \leftarrow O_a$
15:  end for
```

In the algorithm of identifying environments from an environment shown in Table 7-8, if environment E has a predicate relation directing to others through a function verb, then a new environment can be identified according to Rule 5.2). If E has a predicate relation directing to others through a structure verb, then an environment component can be identified according to Rule 6.2). If E has a predicate relation directing to others through a linking verb, then an environment attribute can be identified according to Rule 7.2). If *E* has a constraint relation by a noun or preposition, then a new environment can be identified according to Rule 6.1) and 6.3). Whenever an environment is identified, new environments related to which can be then identified recursively by the algorithm.

Table 7-8 Algorithm of identifying environment from environment

```
Algorithm  Environment_Environment(Environment: E)
1:   for all $O_e$ which has a predicated relation directed from $E$ by $O_v$
2:      if $O_v$ is a function verb that satisfies Rule 5.2) then $Environment \leftarrow O_e$
3:      if $O_v$ is a structure verb that satisfies Rule 6.2) then $EnvironmentComponent \leftarrow O_e$
4:      if $O_v$ is a linking verb that satisfies Rule 7.2) then $EnvironmentAttribute \leftarrow O_e$
5:   end for
6:   for all noun object $O_e$ which is constrained by $E$
7:      if $O_e$ satisfies Rule 6.1) then $EnvironmentComponent \leftarrow O_e$
8:   end for
9:   for all noun object $O_e$ which is constrained by $E$ through a preposition object
10:     if $O_c$ satisfies Rule 6.3) then $EnvironmentComponent \leftarrow O_c$
11:  end for
```

Chapter 8
Algorithm of Transformation from Requirements Text to UML

Current engineering practice is to generate UML diagrams from original customer requirements manually through iterative communicating with the customer. This is often a recursive process: gathering and formulating customer requirements, generating preliminary solutions, and refining customer requirements (Zeng and Cheng, 1991). The final requirement specification, often in the form of UML diagrams, comes from such a brainstorming process. However, as the business becomes more and more complex, multiple customers with different backgrounds are usually involved in the requirement modeling process. Misunderstanding of the customer's real needs is a major issue that may lead to incorrect UML models. There exists a contradiction between ambiguous natural language based product requirements description and the precise UML diagrams that model the product requirements.

Furthermore, for complex engineering projects, requirement document includes a great amount of information, which is extremely tedious for human processing. Efforts have been made to develop automatic or semi-automatic processes to bridge those two extremes: unrestricted natural language text and structured formal representation (Mala and Uma, 2006; Nuseibeh and Easterbrook, 2000). Still, due to the difficulties in processing unrestricted natural language, the success from those efforts is limited (Fantechi et al., 1994; Gnesi et al., 2005; Osborne and MacNish, 1996).

To bridge the gap between unrestricted natural language and formal UML diagrams, an intermediate representation will be useful. The approach proposed in this chapter is based on such an intermediate representation: Recursive Object Model (ROM) (Zeng, 2008). ROM, which is derived from a mathematical theory (Zeng, 2002), can represent all the linguistic elements in natural language. The semantics of a text can be derived from the ROM diagram. The proposed approach firstly generates the ROM diagram of a text describing the product requirements, from which use case and class diagrams are extracted.

Automatic generation of UML models relies on the full understanding of 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 UML standard. Our research aims to simulate the human activities in requirement analysis process and automatically generate UML diagrams through a software system.

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 the semantic structure. This subsection describes the procedures and rules for the automatic generation of UML models from the ROM diagram representing a text.

The transformation from ROM to UML consists of two parts: transformation from ROM to PES and from PES to UML. The first part is the same as transformation from ROM to FBS which is described in 7.1. In this section, the transformation from PES to UML will be dressed from transformation rules and algorithm. Since several categories of UML are used popularly with different features and presentations, which results in different transformation rules and algorithms. Our current research is mainly focused on Use Case Diagram and Class Diagram. Use Case diagram has two types of objects – actor and action whereas Class diagram has class name, method and property. Each component of the two types of UML will be analyzed through ATDM. In this thesis, we apply Use Case diagram and domain model as examples to show how the transformation works founded by the same theory.

8.1 **Transformation Algorithm from ROM to Use Case Model**

8.1.1 Use Case Model analysis

The UML provides use case model notation to illustrate the names of actors, use cases, and the relationships between them. A use case diagram in Figure 8-1 illustrates use cases in a web-based file system. Usually use cases deal primarily in the functional or behavioral requirements that indicate what the system will do (Larman, 2004). A use case diagram does not show the detail of the use cases: it only summarizes some of the relationships between use cases, actors, and systems, for example who uses the system, and what they can do with it.

The components of a Use Case diagram include three parts: a) system or application, b) actors such as people, organizations, or external systems, c) actions.

There are three kinds of actors: primary actors having user goals, supporting actors providing service, and offstage actors having an interest in the behavior of the use case. Take an example of POS system, casher is a primary actor, automated payment authorization service is a supporting actor, and government tax agency is an offstage actor.

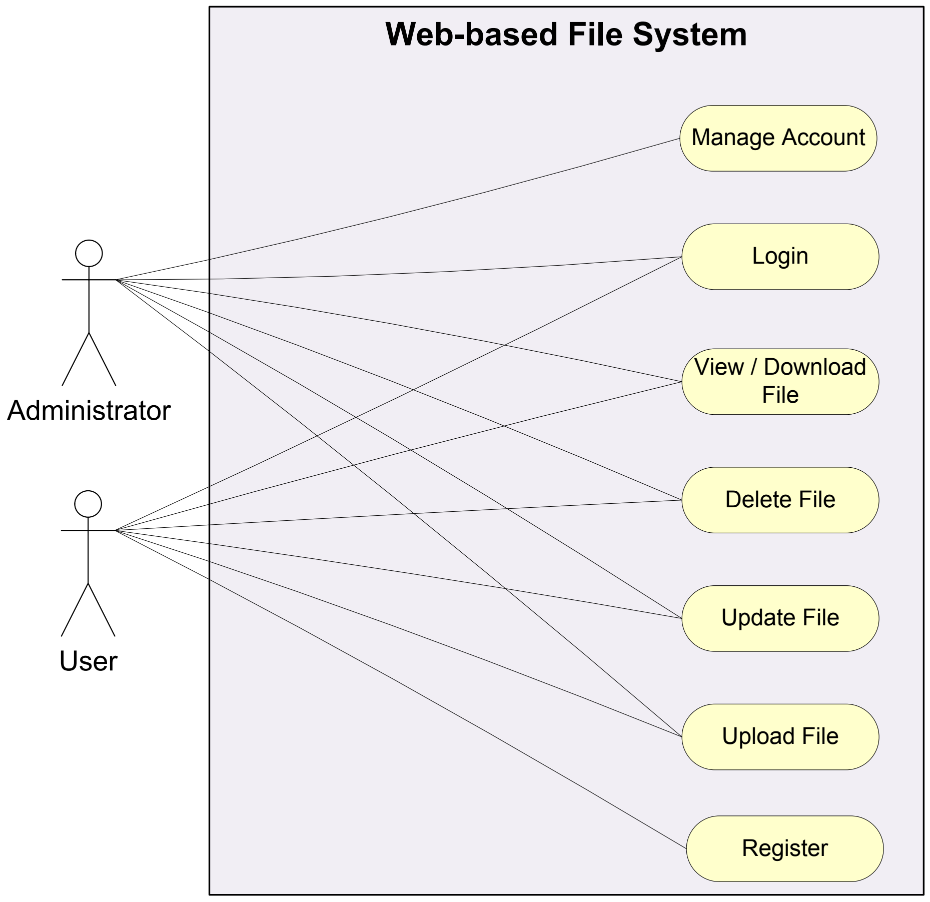


Figure 8-1 An example of Use case diagram

In software engineering, the basic procedure of finding primary use cases are described below: (Larman, 2004)

1. Choose the system boundary: identify if it is a software application, the hardware and application as a unit, that plus a person using it, or an entire organization.
2. Identify the primary actors that have goals fulfilled through using services of the system.
3. Identify the goals for each primary actor.
4. Define use cases that satisfy user goals; name them according to their goal. Usually, user-goal level use cases will be one-to-one with user goals, but there is at least one exception, as will be examined.

This procedure directs professionals to identify use cases from requirements manually. While, the objective of this research is to provide an automatic platform for an unprofessional to transform requirement text into use case model.

8.1.2 Transformation algorithms

The transformation from ROM to UCM shown in Figure 8-2Figure 8-4 can be decomposed into two parts: from a ROM diagram to a product-environment system (PES) and from a PES to a use case model (UCM). The transformation from ROM to PES has been addressed in Chapter 7. This section focuses on the second process: from PES to UCM.

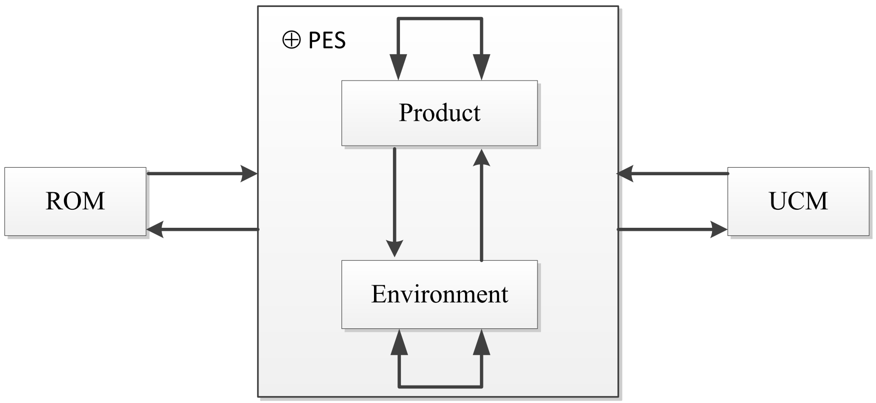


Figure 8-2 Transformation from ROM to UCM

A PES consists of the structure of product, product components, product attributes, environments, environment components, environment attributes, and relations among them. After transformed from ROM to PES, the objects in a ROM diagram have been updated with PES feature.

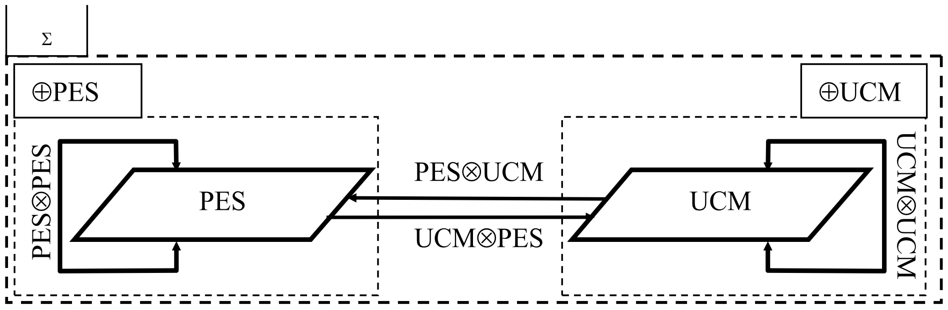


Figure 8-3 Structure of PES - UCM system

Transformation from product-environment system (PES) to use case model (UCM) is the process of identifying use case features based on PES, ROM and POS features, which is shown in Figure 8-3. The structure of PES can be illustrated as Figure 5-1 corresponding to the subjective and objective realms. Environment *E* can be divided into subjective and objective environments. The subjective environment, denoted by *Es*, includes the users of the product whereas the objective environment, denoted by E_o, includes all of the other environment components that have an impact on the behaviour of the product.

Thus the PES provides the intermediate of ROM diagram and use case diagram. Such as, an actor does not belong to system, but it belongs to environments, most case as a human environment. And the system includes product, product components, and product attributes. Actions are relations of actors to the system and other environments, which are verb phrases.

Transition rules from product-environment system (PES) to a use case model are shown in Table 8-1.

Table 8-1 Transition rules from PES to use case model

Define system
Rule 1: The product object in PES is the system.
Rule 1.1: A component of product belongs to the system.
Rule 1.2: A product attribute belongs to system.
Rule 1.3: A product attribute value belongs to system.
Define actor
Rule 2: An environment with human attribute is an actor.
Rule 3: A special noun phrase about organizations or external systems is an actor.
Define action
Rule 4: If two objects in a ROM diagram are related in such a way that: 1) the first object is a function verb directed from an actor, 2) the second object is a noun object directed from the first object, then both objects together makes an action.

Since all of use case features come directly from PES features, once PES features are determined, the use case model is defined, hence, transformation processes from ROM to PES and from PES to UCD are combined.

The proposed algorithm is shown in Table 8-1. The algorithm first determines the system which is composed of product object, product components, product attributes, and attribute values. Recursively determining actors are the second step. Then actions performed by each actor can be determined by applying the transition rules from the ROM diagram.

Table 8-2 Transformation algorithm from PES to use case model

```
Algorithm PES2UCM(ROMDiagram: PES)
1:    Determine the system from PES
2:    Repeat
3:        Determine the actor from PES
4:    Until all the noun objects in PES are considered
5:    Repeat
6:        Determine the action of an actor from PES
7:    Until all the actor are considered
8:    Output the UCM  model
```

In the algorithm of determining use case system shown in Table 8-3, all noun objects in ROM satisfies Rule 1 then the product is determined.

Table 8-3 Algorithm of determining system of use case model

```
Algorithm System(ROMDiagram: PES)
1:    for all $O \in O_n$
```

```
2:      if $O$ satisfies Rule 1 then $System \leftarrow O$
3:      end if
4:  end for
```

Similarly, the objects satisfying Rule 2 or Rule 3 are identified as actors shown in Table 8-4.

Table 8-4 Algorithm of determining actors of use case model

```
Algorithm Actor(ROMDiagram: PES)
1:  for all $O \in O_n$
2:      if $O$ satisfies Rule 2 or Rule 3 then $Actor \leftarrow O$
3:      end if
4:  end for
```

In the algorithm of identifying actions shown in Table 8-5, if an actor A has a predicate relation directing to a noun object by a function verb, then an action of the actor can be identified according to Rule 4.

Table 8-5 Algorithm of identifying actions

```
Algorithm Action(Actor: A, ROMDiagram: PES)
1:  for all verb $V$ directed from $A$ to noun object $N$
2:      if $V$ satisfies Rule 4, $Action \leftarrow V+N$
3:  end for
```

8.2 **Transformation Algorithm from ROM to Domain Model**

8.2.1 Domain model analysis

A domain model is a visual representation of conceptual classes or real-situation objects which illustrates noteworthy concepts in a domain (Larman, 2004). It can act as a source of inspiration for designing some software objects. Domain models have also been called conceptual models, domain object models, and analysis object models.

Applying UML notation, a domain model provides a conceptual perspective illustrated with a set of class diagrams in which no operations (method signatures) are defined. An example of domain model is shown in Figure 8-4. The components of a domain model include domain objects or conceptual classes, associations between conceptual classes, and attributes of conceptual classes.

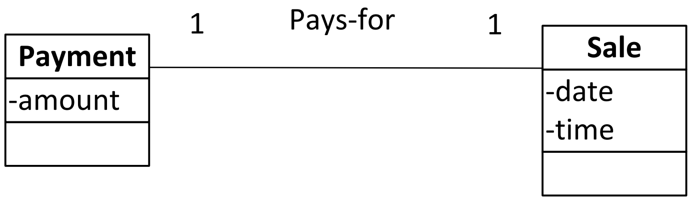


Figure 8-4 An example of domain model

The procedure of creating a domain model is ffinding the conceptual classes and then adding association and attributes.

8.2.1.1 Conceptual classes

A conceptual class is a real-world concept or thing in a conceptual or essential perspective (Larman, 2004). The UP Domain Model contains conceptual classes.

In software engineering, there are three traditional strategies to find conceptual classes:

A) Reusing existing models which are published, well-crafted domain models or data models for many common domains, such as inventory, finance, health, and so forth, however, this method is outside our scope.

B) Using a category list shown in Table 8-6.

Table 8-6 Conceptual Class Category List (Larman, 2004).

Conceptual Class Category	**Examples**
Business transactions	Sale Payment Reservation
Transaction line items	SalesLineItem
Product or service related to a transaction	Item Flight Seat Meal
Rules and policies	RefundPolicy CancellationPolicy
Roles of people or organizations; actors	Cashier Customer Monopoly Player Passenger Airline
Place of transaction or service	Store Airport Plane

	Seat
Noteworthy events	Sale Payment Flight Landing
Physical objects	Item Register Board Airplane
Descriptions of things	ProductDescription FlightDescription
Catalogs	ProductCatalog FlightCatalog
Containers of things (physical or information)	Store Bin Board Airplane
Things in a container	Item Passenger
Other collaborating systems	CreditAuthorizationSystem AirTrafficControl
Records of finance, work, contracts, legal matters	Receipt Ledger EmploymentContract MaintenanceLog
Financial instruments	Cash Check LineOfCredit TicketCredit
Schedules, manuals, documents for references	DailyPriceChangeList RepairManual RepairSchedule
Organizations	SalesDepartment ObjectAirline

C) Identifying noun phrases through linguistic analysis. In textual description of a domain, nouns and noun phrases are considered as candidate conceptual classes or attributes (Moreno, 1997).

Challenges:

- However, in such natural language modeling method, ambiguity of natural language and technical noun-to-class mapping are still challenges.
- The sources of linguistic analysis can be use cases, other documents or the minds of experts. Among those, the fully dressed use cases are one rich source to mine for noun phrase identification of complete domain.
- Distinguishing the candidate conceptual classes and attributes is another challenge.
- Different noun phrases may represent the same conceptual class or attribute, which is another challenge.
- For the imprecision and ambiguities of natural language, linguistic analysis is recommended in combination with the conceptual class category list technique.

8.2.1.2 Association and attributes

An association is a relationship between classes. In the UML, associations are defined as "the semantic relationship between two or more classifiers that involve connections among their instances." The name of an association should comply with the convention of “ClassName-VerbPhrase-ClassName” format where the verb phrase creates a sequence that is readable and meaningful.

An attribute is a logical data value of an object. Informally, most attribute types should be what are often thought of as "primitive" data types, such as numbers and booleans. The type of an attribute should not normally be a complex domain concept, such as a Sale or Airport.

The rule of distinguishing Attributes and Classes: If we do not think of some conceptual class X as a number or text in the real world, X is probably a conceptual class, not an attribute.

As a conceptual model, domain model does not need to list methods, whereas methods can be shown in class diagram for object oriented analysis. However, in this research for illustrating the functions of each class, the methods may be shown if they can be identified from semantics.

8.2.2 Transformation algorithms

The transformation from ROM to DM shown in Figure 8-5 takes the product-environment system (PES) as an intermediate between ROM and domain model (DM). Therefore, the transformation from PES to DM is the focus on the section.

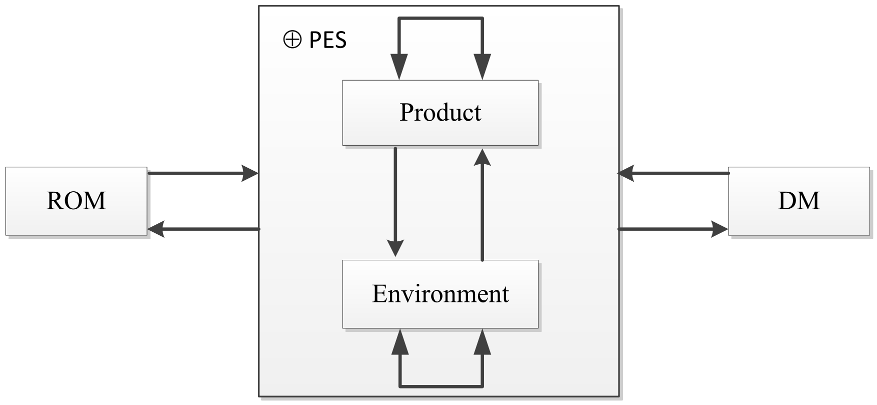


Figure 8-5 Transformation from ROM to DM

The components of a domain model include conceptual classes, associations, attributes, and methods. In a PES, product, product components, product attributes, product attribute value, environments, environment components, environment attributes, environment attribute value, and relations have been identified and updated with PES features in ROM diagram.

The product to be designed is the center of the ROM diagram for a description with most constrained relations. Whereas, a product object is not a conceptual class since it does not exist in the time of design. And product components are not conceptual classes for the same reason. Only environments, environment components, and environment attributes are possibly conceptual classes, since these describe things or services of real-world. Therefore, identifying conceptual classes, associations, attributes, and methods of a domain model based on PES, ROM and POS features is simplified in practice. Such as, conceptual classes are noun objects with

Thus the PES provides the intermediate of ROM diagram and use case diagram. Such as, an actor does not belong to system, but it belongs to environments, in most case as a human environment. And the system includes product, product components, and product attributes. Actions are relations of actors to the system and other environments, which are verb phrases.

Transition rules from product-environment system (PES) to a domain model are shown in Table 8-7.

Table 8-7 Transition rules from PES to domain model

Define conceptual classes
Rule 1: The product or product component object in PES is not a conceptual class.
Rule 2: An environment or environment component may be a conceptual class.
Define attributes
Rule 3: If an environment E_1 is constrained by another environment E_2, E_1 can be the attribute of E_2.
Define associations
Rule 4: The relation of two objects in PES forms the association of the two classes in domain model.
Rule 5: The format of association is *ClassName-VerbPhrase-ClassName*.

The conceptual classes, associations and attributes of PES can be identified from PES features and ROM features easily.

The proposed algorithm is shown in Table 8-8. The algorithm first determines the classes which are identified from environments in PES. Recursively determining attributes is the second step. Then associations between two classes can be determined by applying the transition rules from the ROM diagram.

Table 8-8 Transformation algorithm from PES to domain model

```
Algorithm PES2UCM(ROMDiagram: PES)
1:    Repeat
2:        Determine the class from PES
3:    Until all the environment objects in PES are considered
4:    Repeat
```

```
5:          Determine the attribute of a class from PES
6:      Until all the environment are considered
7:      Repeat
8:          Determine the association of two classes from PES
9:      Until all the relations are considered
10:     Output the DM model
```

In the algorithm of determining a conceptual class shown in Table 8-9, all noun objects in ROM satisfies Rule 1 and Rule 2 then the product is determined.

Table 8-9 Algorithm of determining conceptual class of domain model

```
Algorithm Class(ROMDiagram: PES)
1:  for all $O \in O_n$
2:      if O satisfies Rule 1 and Rule 2 then $Class \leftarrow O$
3:      end if
4:  end for
```

Similarly, the objects satisfying Rule 3 are identified as attributes shown in Table 8-10.

Table 8-10 Algorithm of determining attributes of domain model

```
Algorithm Attribute(ROMDiagram: PES)
1:  for all $O \in O_n$
2:      if O satisfies Rule 3 then $Attribute \leftarrow O$
3:      end if
4:  end for
```

In the algorithm of identifying association shown inTable 8-5, if an actor A has a predicate relation directing to a noun object by a function verb, then an action of the actor can be identified according to Rule 4.

Table 8-11 Algorithm of identifying associations

```
Algorithm  Association(Class: A ,B; ROMDiagram: PES)
1:    for all verb $V$ directed from $A$ to noun object $N$
2:        if $V$ satisfies Rule 4, $Action \leftarrow V + N$
3:    end for
```

Chapter 9
Case Studies for Transformation from Requirements Text to Conceptual Models

Two software prototypes called R2FBS and R2UML have been developed by Min Wang based on the transition rules presented in the previous sections. The prototypes are implemented in the Microsoft Windows environment using C#. The input of the software is a XRD file which stores a ROM diagram corresponding to a design text and the output is a FBS model and UML diagram respectively. R2FBS and R2UML have three critical functional parts. One is the XML parsing combined with graph traversal algorithms and calculation of relations for each object. The second is an algorithm that identifies the PES features from ROM and POS features. The third one is the transformation from PES features to specific conceptual model features. Three examples from different engineering disciplines are used to show how the algorithms work. Besides, the results of these cases are evaluated comparing with the results from experts. The first two examples are used to transform FBS model, while the last one for Use Case and Domain model.

9.1 Design Patent of Low Temperature Clothes Dryer

A United States Patent on "a low temperature clothes dryer" is chosen as an example to show how the rules are applied. The following gives the description of the design 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 therefrom, 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 No.: US 7,377,052 B2; Date of Patent: May 27, 2008*

The text in the design patent of this low temperature clothes dryer is transformed into a ROM diagram as in Figure 9-1, which is the input for proposed algorithm to automatically generating a FBS model from a ROM diagram.

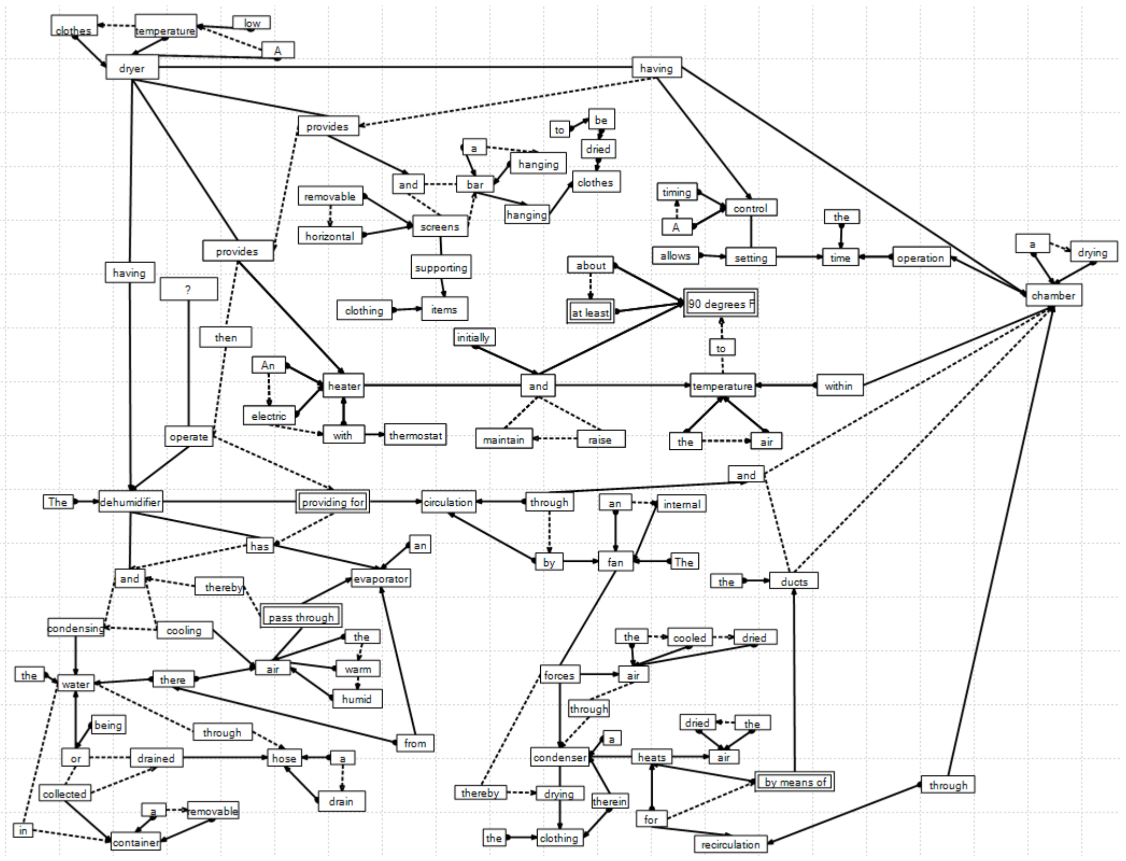


Figure 9-1 ROM diagram for the low temperature clothes dryer

In this example, all the objects are identified and the numbers of relations on each object in the ROM diagram are calculated. The major noun objects and relation numbers are listed in Table 9-1.

Table 9-1 Noun objects in ROM diagram

Object	**Number of predicate**	**Preposition object**	**Predicate +object**	**Role in FBS**
dryer	5		having chamber; provides bar; provides screens; provides heater; having dehumidifier	product
dryer → dry clothes				main function
drying chamber	0	through, within		component
hanging bar	1		hanging clothes	component
horizontal screens	1		supporting items	component
electric heater	2		maintain temperature; raise temperature	component
dehumidifier	4		provide for circulation; has evaporator; condensing water; cooling air	component
timing control	1		setting time	component
evaporator	0	through, from		component

internal fan	0	by	component
thermostat	0	with	component
ducts	0	through	component
removable container	0	in	component
drain hose	0	through	component
condenser	0	through	component
clothes	0		environmen t

By applying the algorithm *R2FBS* introduced in the previous section, the FBS modeling process is shown in the following three steps:

Step 1: Determining product by applying algorithm Product(ROM)

The input is *ROM*, for each noun object in which, the constraint relations and predicate relations are calculated. The noun object "*dryer*" has five predicate relations and three constraint relations, which is the greatest number of predicate and constraint relations in all noun objects. Furthermore, it has no predicate relation directing towards it. Therefore, "*dryer*" can be identified as product object based on Rule 1.

Step 2: Identifying functions and environments by applying algorithm Function_Environment("dryer")

The input is product "*dryer*", which is a noun form of the verb "*dry*" and ts closest constraint is "*clothes*". Therefore, the main function is "*dry clothes*", and environment is "*clothes*" according to Rule 12.

Step 3: Identifying product components and attributes by applying algorithm Component_Attribute("dryer")

> The algorithm searches for noun and adjective objects that constrain the "*dryer*". Those objects are identified as attributes. For example "low temperature" is an attribute of "dryer" according to Rule 3.1).
>
> Then the algorithm searches for noun objects which are directed from "*dryer*" by structure verbs of "*provides*" and "*having*"; therefore, the components of "*screens*", "*bar*", "*chamber*", "*heater*", and "*dehumidifier*" are identified according to Rule 2.2). Based on identified product components, the algorithm calls *Function_Environment*(*component*) and *Componens_Attribute*(*component*) recursively, then sub functions, environments, and components of these components can be identified, such as a function of "*supporting items*" for "*screens*", function of "*hanging clothes*" for "*bar*", attribute of "*drying*" for "*chamber*", attribute of "*removable*" for "*screens*", and components of "ducts", "evaporator", "thermostat", "container", "hose", and "condenser" are identified through related rules. For each new identified environment, *Environment_Environment*(*environment*) is called to identify environments related to it.

At last, the output of the design patent example is shown in Table 9-2, which lists the identified components of PES and FBS by prototype of R2FBS. The PES diagram is generated based on the elicited components shown in Figure 9-2, which illustrates the product, components, functions, environments, attributes and the relations among them for the dryer patent example. A FBS diagram generated by the prototype is shown in Figure 9-3.

Table 9-2 The output of design patent example

PES & FBS	Product	dryer
	Main function	dry clothes
	Product component	horizontal screens, hanging bar, drying chamber, thermostat, timing control, electric heater, dehumidifier, ducts, internal fan, condenser, removable container, drain hose, evaporator
	Product attribute (state)	low temperature for dryer, removable for screens, horizontal for screens, hanging for bar, drying for chamber, electric for heater, timing for control, internal for fan, removable for container, drain for hose
	Sub-function	screens -- supporting items, bar -- hanging clothes, control -- setting time, heater -- maintain temperature, heater -- raise temperature, dehumidifier -- providing for circulation, fan -- forces air, condenser -- heats air, dehumidifier -- condensing water, dehumidifier -- cooling air
	Environment	items, clothes, clothing, temperature, time, circulation, water, air
	Environment attribute	clothing for items, dried for clothes, air for temperature operation for time, cooled for air, dried for air, there for water, warm for air, humid for air, there for air

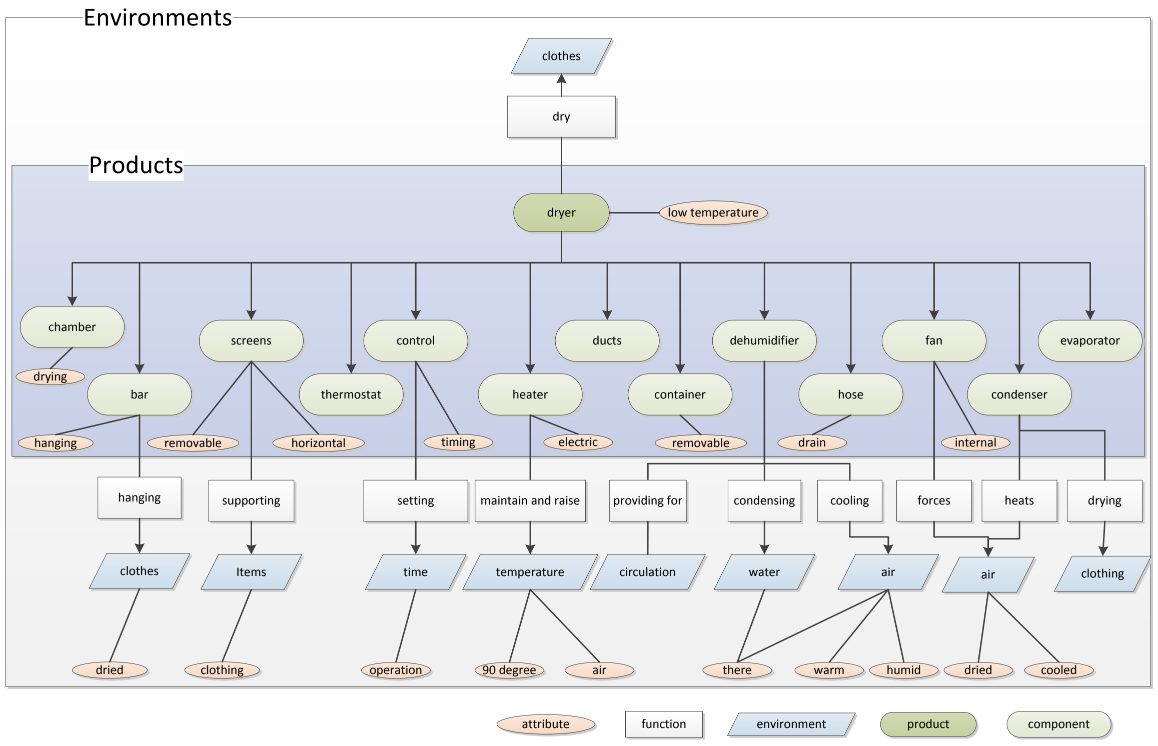
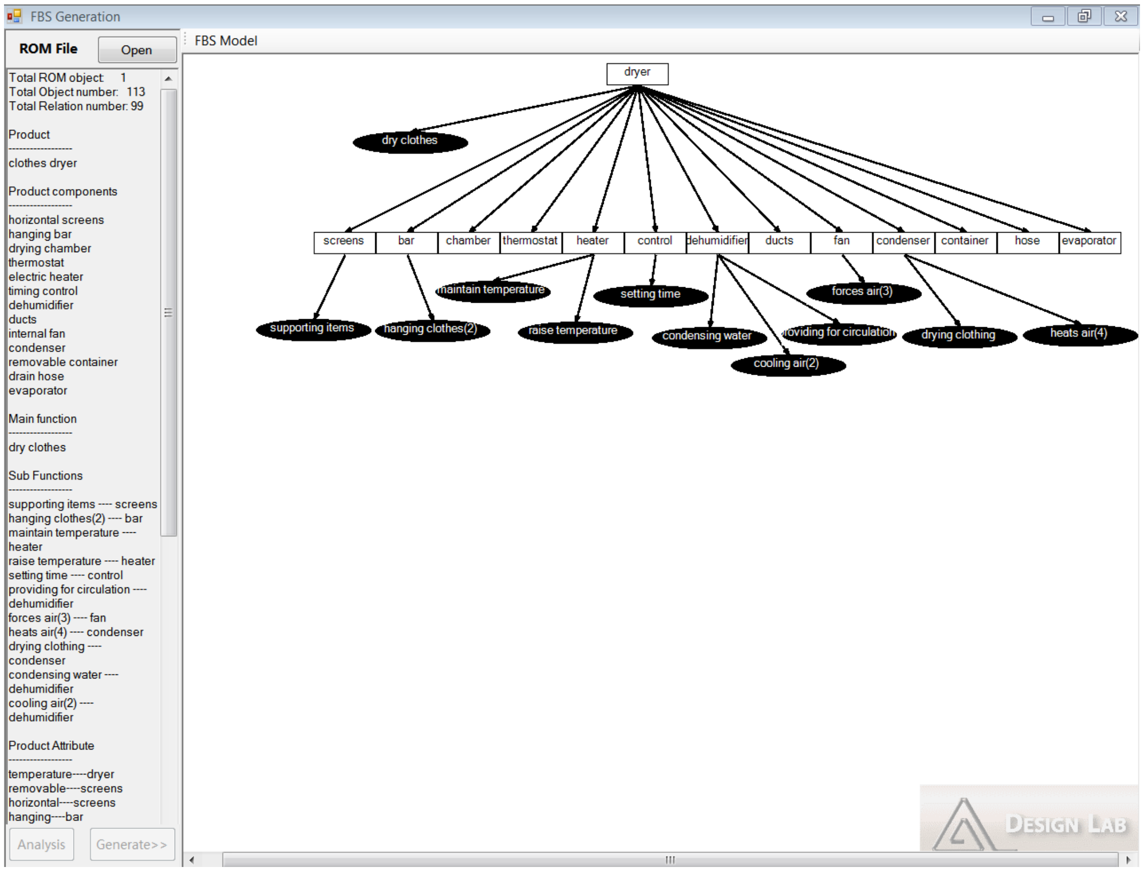


Figure 9-2 PES diagram of automation system example



```
Total ROM object:   1
Total Object number:   113
Total Relation number: 99

Product
----------------
clothes dryer

Product components
----------------
horizontal screens
hanging bar
drying chamber
thermostat
electric heater
timing control
dehumidifier
ducts
internal fan
condenser
removable container
drain hose
evaporator

Main function
----------------
dry clothes

Sub Functions
----------------
supporting items ---- screens
hanging clothes(2) ---- bar
maintain temperature ----
heater
raise temperature ---- heater
setting time ---- control
providing for circulation ----
dehumidifier
forces air(3) ---- fan
heats air(4) ---- condenser
drying clothing ----
condenser
condensing water ----
dehumidifier
cooling air(2) ----
dehumidifier

Product Attribute
----------------
temperature----dryer
removable----screens
horizontal----screens
hanging----bar
```

Figure 9-3 FBS diagram of design paten example

9.2 **Requirements of Energy Trading System**

This second example is extracted from an industrial project. This project aims to identify and develop system requirements starting from a brief description of the energy trading business as shown below.

Energy trading is the activity involving trading energy related commodities, such as power, natural gas, crude oil, and refined products like fuel oil, heat oil, gasoline etc. Energy is not only a consumer product, but also an investment product. As a consumer product, energy producers need to know existing demand, potential demand, and existing supply and potential supply; as an *investment product, investment institutions need to know the return and risk of the investment. Given the huge demand of energy and big energy price volatility, an automation system is the only choice to manage the energy trading.*

In the same way, we generate a ROM diagram for the text, which is illustrated in Figure 9-4.

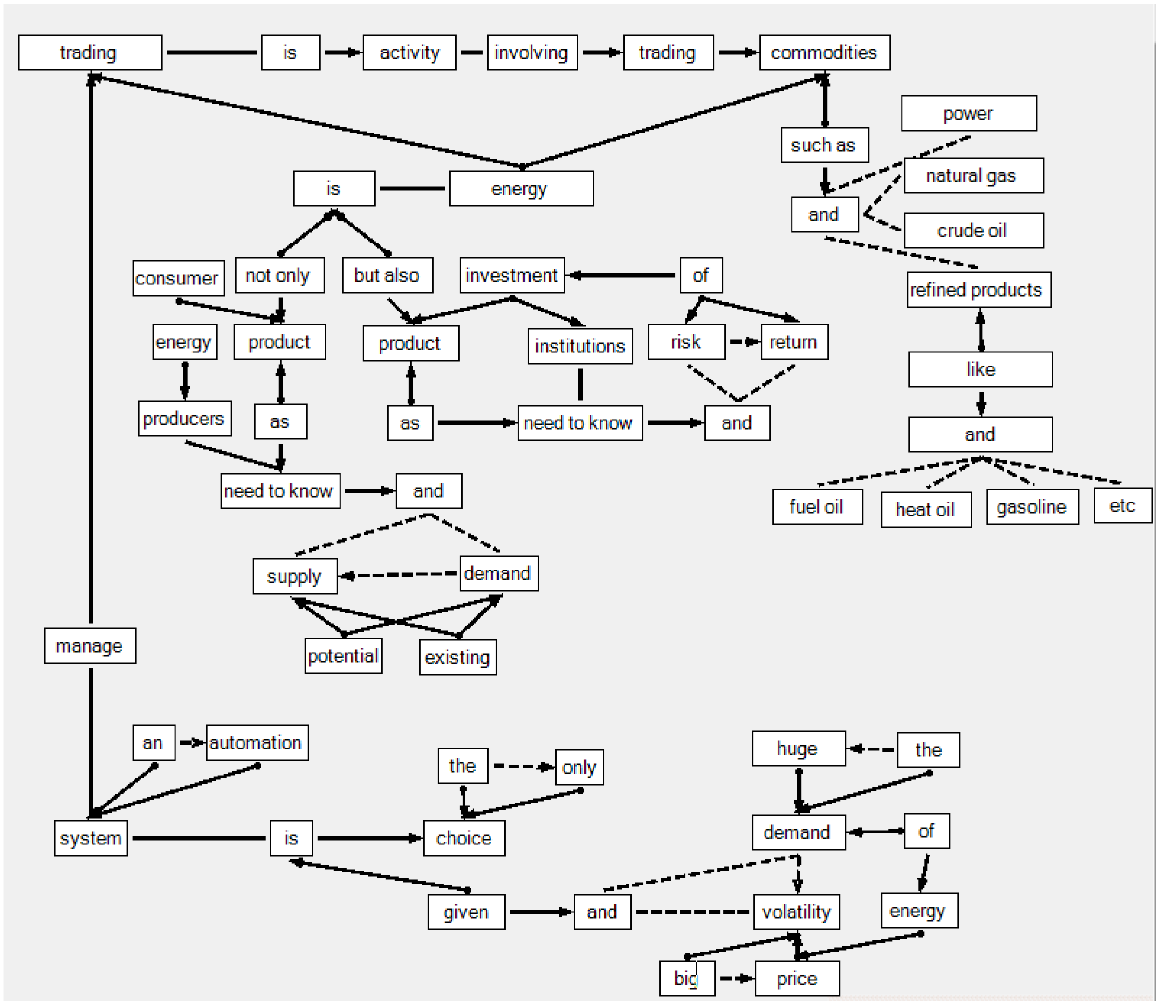


Figure 9-4 ROM diagram of automation system

The output of R2FBS is shown in Table 9-3, and the PES diagram is illustrated in Figure 9-5.

Table 9-3 The output of requirement text example

PES & FBS	Product	automation system
	Product attribute	automation, only choice
	Main function	manage trading
	Product component	n/a
	Sub-function	n/a
	Environment	trading, energy, activity, commodities, product, consumer, demand, producers, supply, investment, return, institutions, risk, power, natural gas, crude oil, refined products, fuel oil, heat oil, gasoline
	Environment attribute	energy for trading, consumer for product, existing for demand, potential for demand, huge for demand, energy for demand, potential for supply, existing for supply, investment for product, investment for risk, energy for commodities, such as for commodities, like for refined products

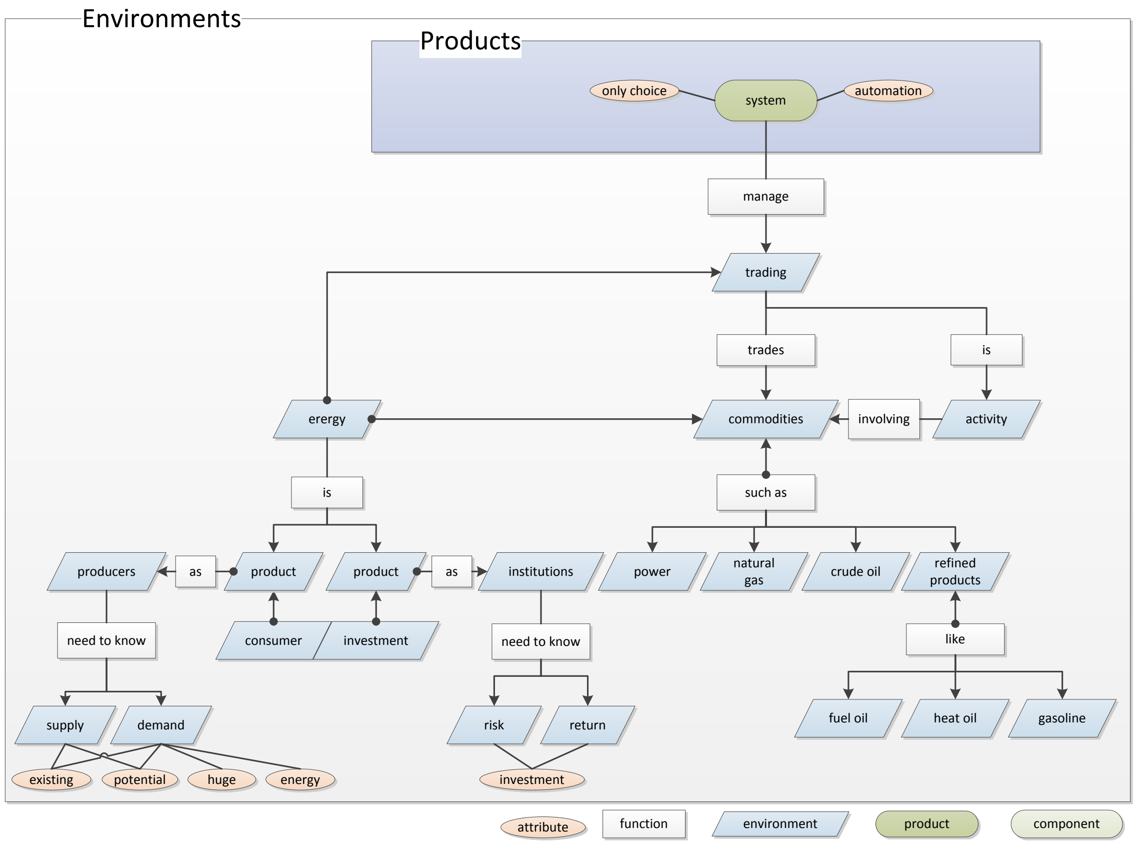


Figure 9-5 PES diagram of automation system example

9.3 **Requirements of POS Management System**

To test the approach of transformation of requirements to UML, a simple example of main scenario text for a POS application will be illustrated in this section. As was discussed above, the input of the case is a natural language based requirement scenario description given as below (Larman, 2004).

1. *Customer arrives at POS checkout with goods and/or services to purchase.*
2. *Cashier starts a new sale.*

3. Cashier enters item identifier.

4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

5. Cashier repeats steps 3-4 until indicates done.

6. System presents total with taxes calculated.

7. Cashier tells customer the total, and asks for payment.

8. Customer pays and system handles payment.

9. System logs completed sale and sends sale and payment information to the external Accounting system and Inventory system.

10. System presents receipt.

11. Customer leaves with receipt and goods.

This requirement scenario text shows a check-out process occurred in most stores. From the system design point of view, the product of this description is a system and there are actors using the system. By analyzing the requirements, the designer can identify the actors, their actions, and some basic functions of this system. The ROM diagram of above description is shown in Figure 9-6.

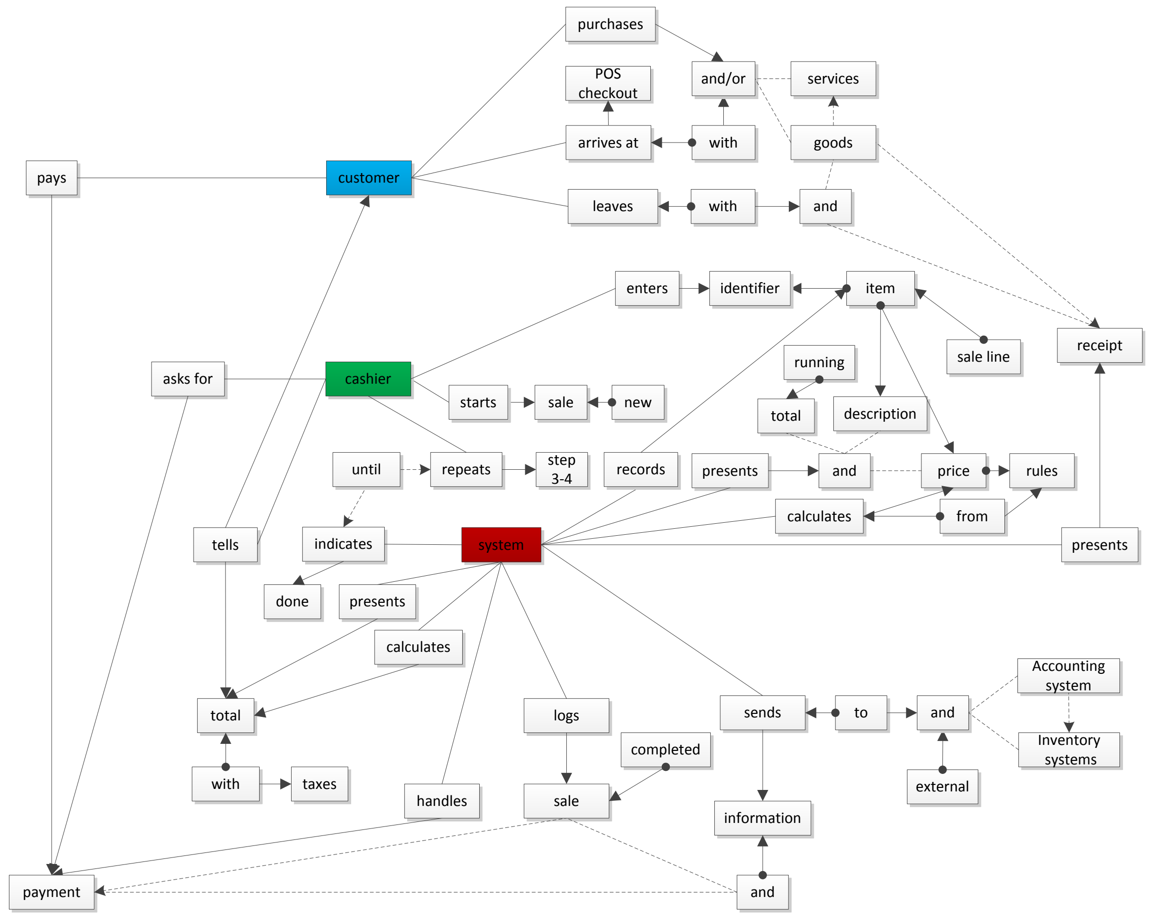


Figure 9-6 ROM diagram of POS system requirement text

The output of PES is shown in Table 9-4, and the PES is illustrated in Figure 9-7.

Table 9-4 The PES of POS system example

PES	Product	System
	Product attribute	Null
	Main function	Records sale line item Presents item description Presents item price

		Presents running total Calculates price Indicates done Calculates total with taxes Presents total with taxes Handles payment Logs completed sale Sends sale and payment information Presents receipt
	Product component	POS checkout
	Sub-function	Null
	Environment	Customer Cashier POS checkout Goods Services Sale line Identifier Sale Item Price Rules Taxes Payment External Accounting System External Inventory System Receipt
	Environment attribute	Completed for sale Running for total

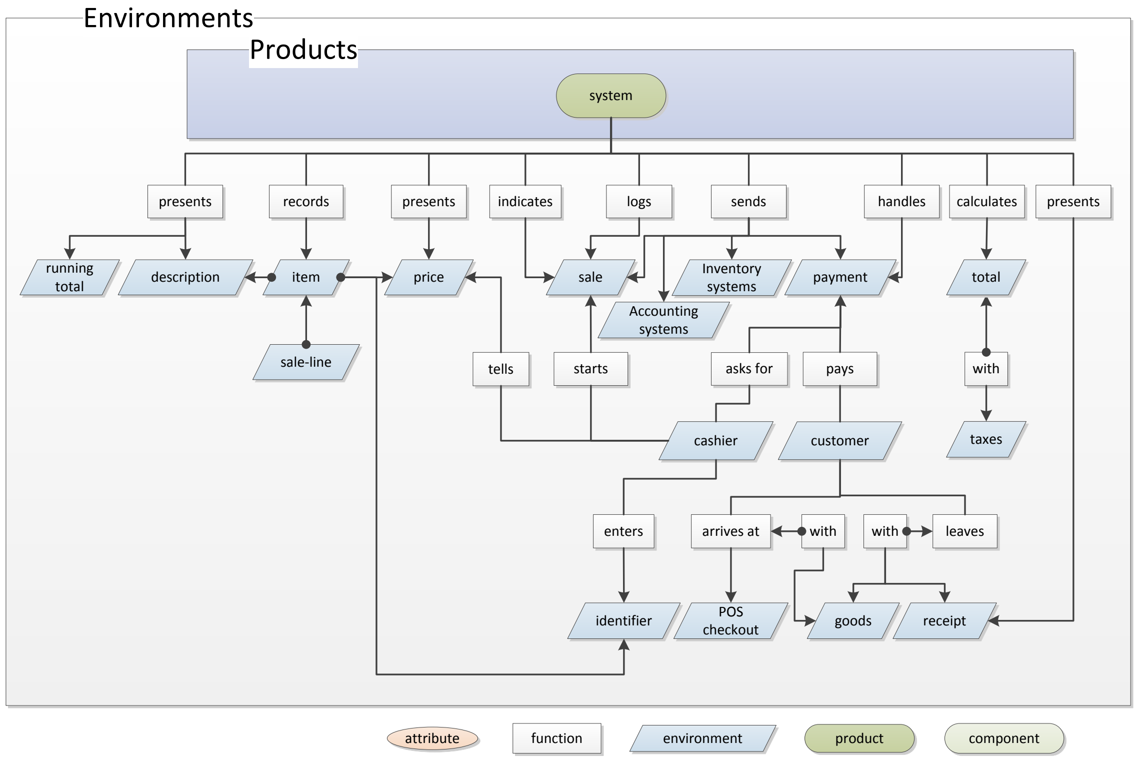
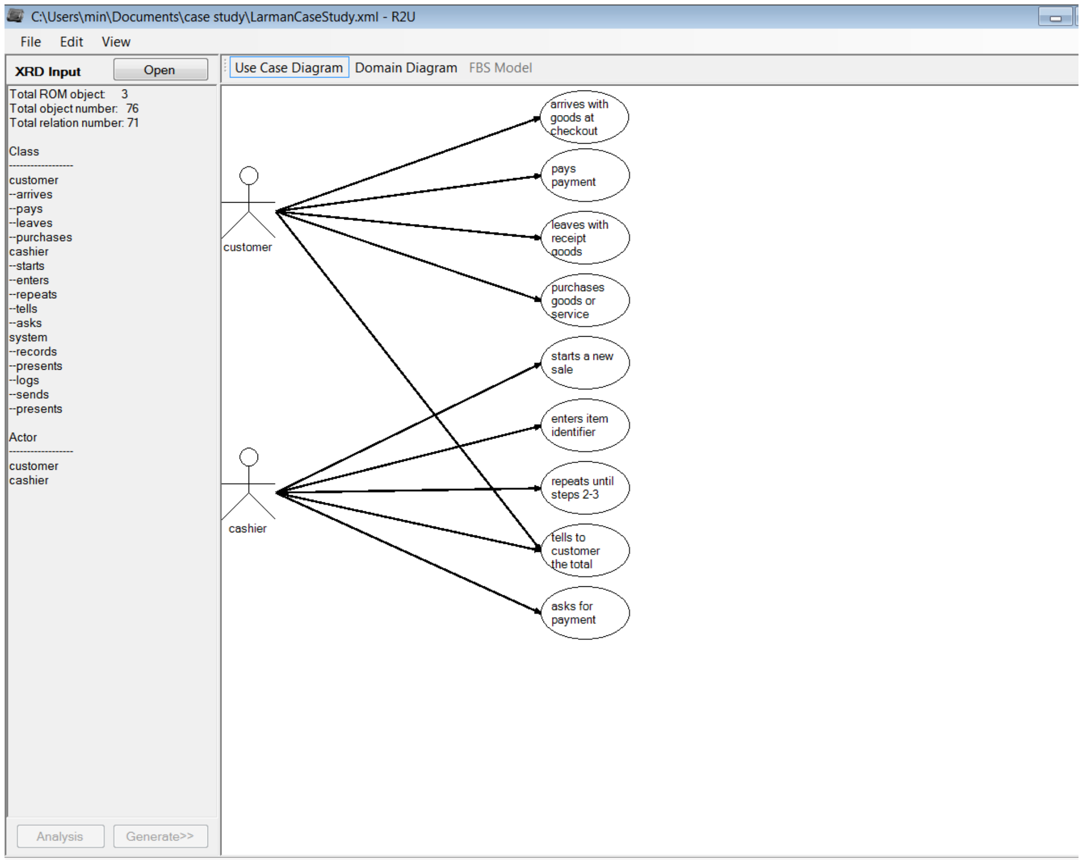


Figure 9-7 PES of POS system example

From XRD file generated by the ROMA system, R2UML software will automatically generate and display the UML diagrams, based on the generation rules introduced in the previous section. Figure 9-8 and Figure 9-9 show the use case and class diagrams of the test case respectively based on the output of PES.



Total ROM object: 3
Total object number: 76
Total relation number: 71

Class
----
customer
--arrives
--pays
--leaves
--purchases
cashier
--starts
--enters
--repeats
--tells
--asks
system
--records
--presents
--logs
--sends
--presents

Actor
----
customer
cashier

Figure 9-8 Use Case diagram output of R2UML

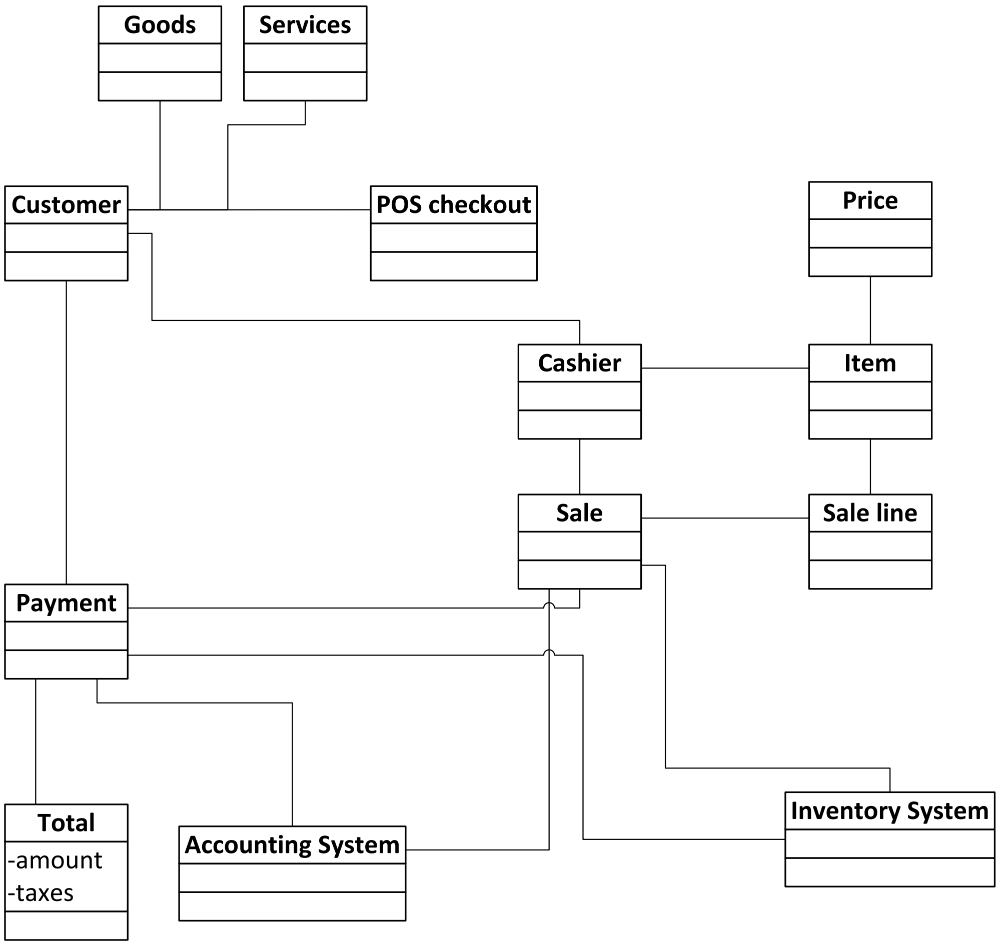


Figure 9-9 Domain Diagram (Class Diagram) output of R2UML

9.4 **Evaluation of the Proposed Methods**

The proposed work is deduced from the recursive logic and Axiomatic Theory and Design Modeling (ATDM) theoretically, which provides theoretical validation. The software prototypes for transformation from ROM to FBS and UML model are development to simulate the algorithms, which provides the simulation validation. Another important evaluation works are experiments through various case studies by applying proposed approaches and prototypes described in the previous subsections. The experimental strategy is developed to show how close the conceptual models developed using our methodology are to those developed by experts in specific domain such as mechanical engineering and requirements engineering. The evaluation method is similar with Seresht's work (Seresht, 2008).

The second case of energy trading system is a real industry project. The aim is helping the company clarify and explicit their requirements through question asking and requirements modeling process. In the project, the transformation from requirements to FBS is only an intermediate for the final deliveries. The clients of the project are very satisfied with our results, which are addressed in the project report (Wang, 2012). Therefore the evaluation of transformation from requirements to conceptual models is focus on the two cases of design patent of low temperature clothes dryer and POS management system.

For the case of design patent of low temperature clothes dryer, the descriptions of patent was distributed to an expert in mechanical engineering, who has in-depth knowledge of modeling the requirements as well as industrial experience. The expert was asked to create a FBS model for the patent description. Then the FBS model developed using our methodology was compared with the expert's model. The comparisons are assigned on the categories of product, product components, functions and attributes. The result "equal" means that the extraction of ROM2FBS is exactly the same as expert's result. "Equivalent" means the result is similar with expert's result in the meaning but with different name. "Incorrect" means the result of ROM2FBS did not exist in the expert's model and it was considered wrong based on our common sense, whiles as “extra” means it was correct or valid but not stated in the expert model. The evaluation results are illustrated in **Table 9-5**. For example our method identified the same product, extra 30.77% but valid (more detailed) product components comparing with expert, and 91.67% equal functions with only 8.33% equivalent functions with different expressions but the same meanings. That was because experts involved domain knowledge in the modeling process, which changed the expressions from text descriptions to technical terminologies. Besides, the percentage of missing components is compared to the expert’s results. For example none of the product, product components and attributes was missing, but 25% functions were missing since the experts put extra functions to the product component which had no such function descriptions in the text however.

We conclude from the results that our approach are better than human analysts at extracting product, functions and attributes exactly according to the descriptions with high accuracy and efficiency.

Table 9-5 Evaluation results of design patent case

Product					
	Equal	Equivalent	Incorrect	Extra	Missing
ROM2FBS	100%	0	0	0	0
Product Components					
	Equal	Equivalent	Incorrect	Extra	Missing
ROM2FBS	69.23%	30.77%	0	0	0
Functions					
	Equal	Equivalent	Incorrect	Extra	Missing

ROM2FBS	91.67%	8.33%	0	0	25%
Attributes					
	Equal	Equivalent	Incorrect	Extra	Missing
ROM2FBS	80%	10%	0	10%	0

For the case of POS management system, the similar evaluation process was performed between experts and the prototype ROM2UML. The evaluation results are illustrated in Table 9-6. We conclude that ROM2UML is closer to the expert's model with high efficiency in terms of completeness of the identified actor, use case, communication and concepts because none of them are missing. Moreover, ROM2UML helped identifying extra information undetected by the analysts.

Table 9-6 Evaluation results of POS management system

Actor					
	Equal	Equivalent	Incorrect	Extra	Missing
ROM2UML	100%	0	0	0	0
Use Case					
	Equal	Equivalent	Incorrect	Extra	Missing
ROM2UML	88.89%	11.11%	0	0	0
Communication					
	Equal	Equivalent	Incorrect	Extra	Missing
ROM2UML	100%	0	0	0	0
Concept					
	Equal	Equivalent	Incorrect	Extra	Missing
ROM2UML	69.23%	7.69%	0	23.08%	0

The results of the experiments proved the validity and feasibility of the proposed methodology. At the same time it proved that the qualities of requirements description in completeness and accuracy are essential, since they directly decide the quality of conceptual models.

9.5 **Summary**

As presented above, the first example about clothes dryer is a patent text, which is associated with the final stage of design; therefore the generated PES-FBS diagram is focused more on product aspects with functions. In contrast, the second example about automation system is a requirement text, which is associated with the early design stage; the PES-FBS diagram is mainly composed by environments of the product. The third example shows the transformation of use case diagram and domain model from requirement text which describes functional scenarios.

Though the given examples used only three short paragraphs respectively, the principles and concepts can be applied to long and large documents. The challenge with large document lies mainly in the complexity of ROM diagrams. The results of examples show that the proposed approach for transformation of design text into FBS model and UML is feasible.

Chapter 10
Conclusions and Future Work

10.1 Conclusions

Requirements elicitation and functional modeling are important at early stages of product design, for which most design information is described by unrestricted natural language. High quality design requirements and function models are extremely useful for the successfulness of the product design and manufacture. Representing and dealing with natural language based requirements are challenging and critical work. This research aims to present criteria for complete and necessary requirements and propose novel approaches to automatically eliciting and formalizing requirements from natural language into structured conceptual models directly.

For requirements elicitation, we propose an Environment-based roadmap for completeness and necessity of requirements; also we develop a question-asking approach to dynamically generate questions for eliciting the necessary and complete requirements based on the roadmap. For requirements modeling, we propose a generic formalization for transforming requirements into conceptual models.

This research applies ROM to represent requirements text. The ROM diagram corresponding to the text carries the main semantic information implied in the text. Both the ROM diagram and the conceptual model are related to a product-environment system through the Axiomatic Theory of Design Modeling. Rules are developed to map the objects and relations in a ROM diagram to the concepts and relations in a conceptual model. Algorithms are developed to support the transformation from a ROM diagram to FBS model, Use Case Model, and Domain Model.

For assisting our research, a few software prototypes have been designed and implemented for question generation and transformation to FBS, Use Case, and Domain Models. Three case studies in different fields are performed to examine and demonstrate how the proposed approach works. For a new research approach derived from EBD theory, this work has been theoretical proven and experimental validated through these case studies.

It must be indicated that the proposed approach does not intend to exclude human users from the loop. On the contrary, this approach may help engineers better understand requirements, especially in a large project, by reducing the ambiguities of human understanding in analyzing requirements and by increasing the consistency of the final function models when multiple engineers are involved. Besides, this thesis is founded by EBD theory; meanwhile, it enriches the approach of Environment Analysis in EBD.

10.2 **Future Work**

In this present thesis, an Environment-based requirement roadmap is proposed to support requirements elicitation; and a new approach by ROM analysis is presented for transformation from natural language to conceptual models. Through a few case studies, the results have shown they are effective and feasible to support requirements modeling. The following work can be continued in the future.

1) As can be seen from this thesis, our current approach largely depends on the capability and capacity of the ROMA system, which captures the semantics of natural language text. Therefore, the accuracy of ROMA is of a critical importance. Although ROMA is already very robust, it is still under further development.

2) Another problem that needs to be dealt with is the study of the structure of large requirement documents so that they can be pre-processed by the ROMA system.

3) It must be pointed out that the examples used in this thesis are short paragraphs. A more complex text may increase the size of ROM diagram. Though theoretically, the present algorithms will work for ROM diagrams of any complexity, future research is needed for how to efficiently transform large text into a set of shorter paragraphs.

4) The rules for transforming a ROM diagram to a conceptual model should be further validated through a more comprehensive system test based on statistical analysis.

5) Transformation from ROM diagram to other conceptual models such as class diagram and ER model should be conducted.

References

Akao, Y. and Glenn, M., 2003. The leading edge in QFD: past, present, and future. International Journal of Quality and Reliability Management, 20(1): 20-35.

Al-Safadi, L.A.E., 2009. Natural language processing for conceptual modeling. International Journal of Digital Content Technology and its Applications 3(3): 47-59.

Alexander, I., 2003. Misuse cases help to elicit non-functional requirements. Computing & Control Engineering 40-45.

Alvarez, R., 2002. Discourse analysis of requirements and knowledge elicitation interviews, Proceedings of the 35th Hawaii International Conference on System Sciences, Big Island.

Ambriola, V. and Gervasi, V., 2006. On the systematic analysis of natural language requirements with CIRCE. Automated Software Engineering, 13(1): 107-167.

Amyot, D., 2003. Introduction to the User Requirements Notation: Learning by Example, SITE, University of Ottawa.

Andreou, A.S., 2003. Promoting software quality through a human, social and organisational requirements elicitation process. Requirements Engineering, 8(2): 85-101.

Aranda, G.N., Vizcaíno, A. and Piattini, M., 2010. A framework to improve communication during the requirements elicitation process in GSD projects. Requirements Engineering, 15(4): 397-417.

Arthur, J.D. and Gröner, M.K., 2005. An operational model for structuring the requirements generation process. Requirements Engineering, 10(1): 45-62.

Asimow, M., 1962. Introduction to design. Prentice Hall.

Bhatta, S. and Goel, A., 1994. Model-based discovery of physical principles from design experiences. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 8(2): 113-123.

Bhatta, S. and Goel, A., 1997. Learning generic mechanisms for innovative design adaptation. Journal of Learning Sciences, 6(4): 367-396.

Borland, 2011. CaliberRM Enterprise software requirements management.

Carod, N.M. and Cechich, A., 2009. A Classification framework for Software Requirements Prioritization Approaches. Revista Colombiana de Computación, 10(2).

Casamayor, A., Godoy, D. and Campo, M., 2009. Semi-Supervised Classication of Non-Functional Requirements: An Empirical Analysis. Artificial Intelligence, 44(35-45).

Cascini, G., 2012. TRIZ-based Anticipatory Design of Future Products and Processes. Transactions of the SDPS: Journal of Integrated Design and Process Science, 16(3).

Chase, S.C. and Liew, P., 2001. A framework for redesign using FBS models and grammar adaptation. In: B.d. Vries, J.v. Leeuwen and H. Achten (Editors), Computer aided architectural design futures 2001. Kluwer Academic Publishers, pp. 467-477.

Chen, L. and Zeng, Y., 2009. Automatic generation of UML diagrams from product requirement requirements described by natural language, The 2009 ASME International Design Engineering Technical Conferences (IDETC) and Computers and Information in Engineering Conference, San Diego, USA.

Chen, P.P.-S., 1983. English sentence structure and entity-relationship diagrams. Information Sciences, 29(2): 127-149.

Chen, Z., Yao, S., Lin, J., Zeng, Y. and Eberlein, A., 2007. Formalization of product requirements: from natural language descriptions to formal specifications. Int. J. Manufacturing Research, 2(3): 362-387.

Chen, Z.Y. and Zeng, Y., 2006. Classification of product requirements based on product environment. Concurrent Engineering: Research and Applications, An International Journal, 14(3): 219-230.

Christel, M.G. and Kang, K.C., 1992. Issues in requirements elicitation, Software Engineering Institute, Carmegie Mellon University, Pittsburgh, Pennsylvania.

Christophe, F., Wang, M., Coatanéa, E., Zeng, Y. and Bernard, A., 2011. Grammatical and semantic disambiguation of requirements at elicitation and representation stages, Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Washington, DC, USA (accepted).

Clausing, D., 1998. Total quality development. American Society of Mechanical Engineers, New York.

Cleland-Huang, J., Settimi, R., Zou, X. and Solc, P., 2007. Automated classification of non-functional requirements. Requirements Engineering, 12(2): 103-120.

Conklin, Jeff and Yakemovic, K.C.B., 1991. A process-oriented approach to design rationale. Human-Computer Interaction, 6(3): 357-391.

Coughlan, J. and Macredie, R.D., 2002. Effective communication in requirements elicitation: a comparison of methodologies Requirements Engineering, 7(2): 47-60.

Cysneiros, L.M., Leite, J.C.S.d.P. and Neto, J.d.M.S., 2001. A rramework for integrating non-functional requirements into conceptual models. Requirements Engineering, 6(2): 97-115.

Darlington, M.J. and Culley, S.J., 2002. Current research in the engineering design requirement. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 216(3): 375-388.

Davis, M.D., Sigal, R. and Weyuker, E.J., 1994. Computability, complexity and languages: fundamentals of theoretical computer science. Academic Press.

Deng, Y.M., 2002. Function and behavior representation in conceptual mechanical design. AI EDAM, 16(05): 343-362.

Diev, S., 2006. Use cases modeling and software estimation: applying use case points. SIGSOFT Software Engineering Notes, 31(6): 1-4.

Erden, M.S. et al., 2008. A review of function modeling: Approaches and applications. AI EDAM, 22(02): 147-169.

Eris, O., 2004. Effective inquiry for innovative engineering design. Kluwer Academic Publishers, Stanford University, 154 pp.

Esmaeilsabzali, S., Day, N.A., Atlee, J.M. and Niu, J., 2010. Deconstructing the semantics of big-step modelling languages. Requirements Engineering, 15(2): 235-265.

Fabian, B., Gürses, S., Heisel, M., Santen, T. and Schmidt, H., 2010. A comparison of security requirements engineering methods. Requirements Engineering, 15(1): 7-40.

Fantechi, A. et al., 1994. Assisting requirement formalization by means of natural language translation. Formal Methods in System Design, 1994. 4: p. 243–263., 4(3): 243-363.

Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R., 1996. Advances in knowledge discovery and data mining. Menlo Park, Calif. : AAAI Press : MIT Press, c1996.

Fowler, M., 2003. UML distilled: a brief guide to the standard object modeling language. Object technology series. Addision-Wesley.

Friedenthal, S., Steiner, R. and Moore, A., 2008. A practical guide to SysML: the systems modeling language. Amsterdam; Elsevier/Morgan Kaufmann OMG Press, Boston.

Gero, J.S. and Kannengiesser, U., 2007. A function-behavior-structure ontology of processes. AI EDAM, 21(04): 379-391.

Gershenson, J.K. and Stauffer, L., A., 1999. A taxonomy for design requirements from corporate customers. Research in Engineering Design, 11(2): 103-115.

Gnesi, S., Lami, G., Trentanni, G., Fabbrini, F. and Fusani, M., 2005. An Automatic Tool for the Analysis of Natural Language Requirements. International Journal of Computer Systems Science and Engineering, 20(1).

Goel, A.K., Rugaber, S. and Vattam, S., 2009. Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language. AI EDAM, 23(Special Issue 01): 23-35.

Gorschek, T. and Wohlin, C., 2006. Requirements abstraction model. Requirements Engineering, 11(1): 79-101.

Greenspan, S. and Feblowitz, M., 1993. Requirements engineering using the SOS paradigm, 1st International symposium on requirements engineering, San diego, USA, pp. 260-263.

Hands, K., Peiris, D.R. and Gregor, P., 2004. Development of a computer-based interviewing tool to enhance the requirements gathering process. Requirements Engineering, 9(3): 204-216.

Hickey, A. and Davis, A., 2004. A unified model of requirements elicitation. Journal of Management Information Systems, 20(4): 65-84.

Hubbard, R., Mead, N. and Schroeder, C., 2000. An assessment of the relative efficiency of a facilitator-driven requirements collection process with respect to the conventional interview method, International Conference on Requirements Engineering. IEEE Computer Society Press, Los Alamitos, CA.

Hull, E., Jackson, K. and Dick, J., 2005. Requirements Engineering. Springer, London.

IBM, 2011a. IBM Rational DOORS.

IBM, 2011b. IBM Rational RequisitePro.

Johnson, J. and Henderson, A., 2011. Conceptual Models Core to Good Design. Morgan & Claypool Publishers.

Kanda, A., Teegavarapu, S., Summers, J. and Mocko, G., 2008. Patent Driven Design: Exploring the Possibility of Using Patents to Drive New Design, Tools and Methods for Competitive Engineering Conference, Izmir, Turkey.

Kang, K.C., Cohen, S.G., Hess, J.A., Novack, W.E. and Peterson, A.S., 1990. Feature-oriented domain analysis feasibility study. Software Engineering Institute, Pittsburgh, PA.

Kean, L., 1997. Feature-oriented domain analysis, Carnegie Mellon, Software Engineering Institute.

Lano, K., 2007. Formal specification using interaction diagrams, SEFM'07 proceedings.

Lano, K., 2009. UML 2 semantics and applications. Wiley, Hkboken, N.J.

Lano, K. and Clark, D., 2007. Direct semantics of extended state machines, TOOLS'07.

Larman, C., 2004. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Addison Wesley Professional.

Lecoeuche, R., Mellish, C. and Roberston, D., 1998. A framework for requirements elicitation through mixed-initiative dialogue, International Conference on Requirements Engineering: Putting Requirements Engineering to Practice.

Leffingwell, D. and Widrig., D., 2003. Managing Software Requirements: A Use Case Approach, Second Edition. Addison-Wesley, Boston, MA.

Leite and Cesar, J., 1987. A survey on requirements analysis, Department of Information and Computer Science, University of California at Irvine.

Lin, J. and Katz, B., 2003. Question Answering from the Web Using Knowledge Annotation and Knowledge Mining Techniques, Conference on Information and Knowledge Management, Proceedings of the twelfth international conference on Information and knowledge management, New Orleans, LA, USA pp. 116-123.

Liu, D., Subramaniam, K., Eberlein, A. and Far, B.H., 2004. Natural language requirements analysis and class model generation using UCDA. Innovations in Applied Artificial Intelligence, Lecture Notes in computer Science, 3029: 295-304.

Loucopoulos, P. and Karakostas, V., 1995. System requirements engineering. McGraw-Hill.

Loucopoulos, P. and Kavakli, E., 1995. Enterprise modelling and the teleological approach to requirements engineering. International journal of intelligent and cooperative information systems, 4(1): 45-79.

Luh, D.-B., Ma, C.-H., Hsieh, M.-H. and Huang, C.-Y., 2012. Using the Systematic Empathic Design Method for Customer-centered Products Development. Transactions of the SDPS: Journal of Integrated Design and Process Science, 16(2).

Macaulay, L., 1996. Requirements engineering. Springer, London.

Mala, G.S.A. and Uma, G.V., 2006. Automatic construction of object oriented design models [UML diagrams] from natural language requirements specification, Pricai 2006:

Trends in Artificial Intelligence, Proceedings. Lecture Notes in Artificial Intelligence, pp. 1155-1159.

Maletz, M., 2008. Integrated requirements modeling A contribution towards the integration of requirements into a holistic product lifecycle management strategy, Graz, Austria.

Marefat, M. and Kashyap, R.L., 1990. Geometric Reasoning for Recognition of Three-Dimensional Object Features. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(10): 949-965.

Markovic, S. and Barbosa, L.S., 2008. Semantics of OCL specified with QVT. Software and Systems Modelling, 7(4).

Martin, J., 1987. Information engineering, Vol 3: Meeting the needs of users more directly. Savant Institute, Carnforth, U.K.

Mead, N.R., 2006. Requirements elicitation introduction, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Medyna, G., Nonsiri, S., Coatanéa, E. and Bernard, A., 2012. Modelling, Evaluation and Simulation during the Early Design Stages: Toward the Development of an Approach Limiting the Need for Specific Knowledge. Transactions of the SDPS: Journal of Integrated Design and Process Science.

Meyer, B., 1985. On formalism in specifications. IEEE Software, 2(1): 6-26.

Mich, L., Franch, M. and Inverardi, P.N., 2004. Market research for requirements analysis using linguistic tools. Requirements Engineering, 9(1): 40-56.

Minneman, S.L., 1991. The social construction of a technical reality: empirical studies of group engineering design practice, Stanford University.

Moreno, A.M., 1997. Object Oriented Analysis from Textual Specifications, Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering, Madrid.

Morkos, B., Shankar, P. and Summers, J.D., 2012. Predicting requirement change propagation, using higher order design structure matrices: an industry case study. Journal of Engineering Design, 23(12): 905-926.

Moroz, A., 2011. Environment-Based Design of software: an agile software design mothod, Concordia Universtiy Montreal.

Mullery, G.P., 1979. CORE: a method for controlled requirements specification, Proceedings of the 4th International Conference on Software Engineering (ICSE-4). CA: IEEE Computer Society Press, Munich, Germany, pp. 126-135.

Nuseibeh, B. and Easterbrook, S., 2000. Requirements engineering: a roadmap, Proceedings of the Conference on The Future of Software Engineering ACM Press Limerick, Ireland pp. 35-46.

OMG-SysML, 2011. OMG SysML. (http://www.omgsysml.org/, Accessed on 2 May 2011).

OMG-UML, 2011. OMG UML. (http://www.omg.org/spec/UML/, Accessed on 2 May 2011).

Osborne, M. and MacNish, C.K., 1996. Processing natural language software requirement specifications, 2nd IEEE International Conference on Requirements Engineering. IEEE Press.

Overmyer, S., Lavoie, B. and Rambow, O., 2001. Conceptual modeling through linguistic analysis using LIDA, In Proceedings of 23rd International Conference on Software Engineering (ICSE 2001), Toronto, Canada.

Oxman, R., 2004. Think-maps: teaching design thinking in design education. Design Studies, 25(1): 63-91.

Ozaydin, B. and Tanik, M.M., 2011. System Moldeling and Analysis Using Communication Channels. Transactions of the SDPS: Journal of Integrated Design and Process Science, 15(1): 1-33.

Prabhakar, S. and Goel, A.K., 1998. Functional modeling for enabling adaptive design of devices for new environments. Artificial Intelligence in Engineering, 12(4): 417-444(28).

QFD-Institute, 2005. Frequently asked questions about QFD, (http://www.qfdi.org/, Accessed on 5 August 2007).

Rumbaugh, J., Jacobson, I. and Booch, G., 1998. The unified modeling language reference manual. Addison-Wesley.

Sajid, A., Nayyar, A. and Mohsin, A., 2010. Modern trends towards requirement elicitation, Proceedings of the 2010 National Software Engineering Conference (NSEC'10). ACM, New York, NY, USA.

Saleh, K. and Al-Zarouni, A., 2004. Capturing non-functional software requirements using the user requiremetns notation, International research conference on innovations in information technology.

Schach, S.R., 2002. Obect-oriented and classical software engineering. McGraw Hill.

Schiffrin, D., 1994. Approaches to discourse. Blackwell Publishers Ltd, Oxford, England.

Seresht, S.M., 2008. A methodology for semi-automatic assistance in elicitation and analysis of textual user requirements, Concordia University, Montreal, 122 pp.

Seresht, S.M. and Ormandjieva, O., 2008. Automated Assistance for Use Cases Elicitation from User Requirements Text, 11th Workshop on Requirements Engineering, Barcelona.

Simon, H., 1969. The sciences of the artificial. MIT Press, Cambridge, MA.

Soares, M.S. and Vrancken, L., 2008. Model-driven user requirements specification using SysML. Journal of Software, 3(6): 57-68.

Southwell, K. et al., 1987. Requirements definition and design, The STARTS Guide, Second Edition. National Computing Centre, pp. 177-313.

Storrle, H. and Hausmann, J., 2005. Towards a formal semantics of UML 2.0 activities. Software Engineering, 64.

Subramaniam, K., Far, B.H. and Eberlein, A., 2004. Automating the transition from stakeholders' requests to use cases in OOAD, Electrical and Computer Engineering, 2004. Canadian Conference on, pp. 515-518 Vol.1.

Tjoa, A.M. and Berger, L., 1993. Transformation of requirements specifications expressed in natural language into an EER model, Proceedings of the 12th International conference on Entity-Relationship Approach (ER'93), Airlington, Texas USA.

Tom, M. and Sitte, J., 2009. Future user requirement elicitation for technology investment: A formal approach, IEEE International Conference on Systems, Man, and Cybernetics. SMC San Antonio, TX, pp. 2116-2121.

Tseng, Y.-H., Wang, Y.-M., Juang, D.-W. and Lin, C.-J., 2005. Text mining for patent map analysis, IACIS Pacific 2005 Conference Proceedings, pp. 1109-1116.

Ullman, D.G., 2002. The mechanical design process. McGraw-Hill, New York.

Umeda, Y., Takeda, H., Tomiyama, T. and Yoshikawa, H., 1990. Function, behaviour, and structure. Applications of artificial intelligence in engineering: 177-193.

Umeda, Y. and Tomiyama, T., 1995. FBS modeling: modeling scheme of function for conceptual design, Proceedings of working papers of the 9th international workshop on qualitative reasoning about physical systems, Amsterdam, pp. 271-278.

Verstijnen, I.M., van Leeuwen, C., Goldschmidt, G., Hamel, R. and Hennessey, J.M., 1998. Sketching and creative discovery. Design Studies, 19(4): 519-546.

Volere, 2010. Requirements specification template. In: James and S. Robertson (Editors).

Wang, M., 2012. Project Report for Identifying Requirements in Energy Trading System, Concordia Institute for Information Systems Engineering, Montreal, Canada.

Wang, M. and Zeng, Y., 2009. Asking the right questions to elicit product requirements. International Journal of Computer Integrated Manufacturing, 22(4): 283-293.

Wang, M., Zeng, Y., Chen, L. and Eberlein, A., 2013. An algorithm for transforming design text ROM diagram into FBS model. Computers in Industry, 64: 499-513.

Weilkiens, T., 2007. Systems engineering with SysML/UML : Modeling, analysis, design. Amsterdam; Morgan Kaufmann OMG Press, Elsevier, Boston.

Weissman, A. et al., 2011. A computational framework for authoring and searching product design specifications. Advanced Engineering Informatics, 25(3): 516-534.

Wen, K., Tan, S., Wang, J., Li, R. and Gao, Y., 2013. A model based transformation paradigm for cross-language collaborations. Advanced Engineering Informatics(0).

Wen, K., Wang, J., Li, R. and Li, Y., 2011. Cross-Language Transformation based on Recursive Object Model in Understanding Product Requirements, SDPS, Jeju Island, South Korea.

Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C. and Bruel, J.-M., 2010. RELAX: A language to address uncertainty in self-adaptive systems requirement. Requirements Engineering, 15(2): 177-196.

Wiggins, G. and McTighe, J., 2005. Understanding by design. Association for Supervision and Curriculum Development (ASCD).

Wikipedia, 2011. Requirements. (http://en.wikipedia.org/wiki/Requirements, Accessd on 2 March 2011).

Wilson, R.H. and Latombe, J.-C., 1994. Geometric reasoning about mechanical assembly. Artificial Intelligence, 71(2): 371-396.

Wölkl, S. and Shea, K., 2009. A computational product model for conceptual design using SysML, International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, ASME 2009, San Diego, USA.

Wood, J. and Silver, D., 1995. Joint application development. Wiley, New York.

Wootton, A.B., Cooper, R. and Bruce, M., 1998. Requirements capture: theory and practice. The Engineering and Physical Sciences Research Council Technology Management Initiative, 18(8-9): 497-511.

Xu, L., Ziv, H. and Richardson, D., 2005. Towards modeling nonfunctional requirements in software architecture, In Proceedings of Aspect-Oriented Software Design, Workshop on AspectOriented Requirements Engineering and Architecture Design.

Yoshioka, M. et al., 2004. Physical concept ontology for the knowledge intensive engineering framework. Advanced Engineering Informatics, 18(2): 95-113.

Young, R.R., 2001. Effective Requirements Practices. Addison-Wesley, Boston, MA.

Zahniser, R.A., 1990. How to speed development with group sessions. IEEE Software: 109-110.

Zeng, Y., 2002. Axiomatic theory of design modeling. Transaction of SDPS: Journal of Integrated Design and Process Science, 6(3): 1-28.

Zeng, Y., 2004a. Environment-based design: process model, Concordia Institute for Information Systems Engineering, Concordia University, Montreal.

Zeng, Y., 2004b. Environment-based formulation of design problem. Transaction of SDPS: Journal of Integrated Design and Process Science, 8(4): 45-63.

Zeng, Y., 2008. Recursive Object Model (ROM) - Modeling of Linguistic Information in Engineering Design. Computers in Industry, 59(6): 612-625.

Zeng, Y., 2011. Environment-based Desgin (EBD), Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Washingtong DC, USA.

Zeng, Y. and Cheng, G.D., 1991. On the logic of design. Design Studies, 12(3): 137-141.

Zeng, Y. and Gu, P., 1999. A science-based approach to product design theory Part II: Formulation of design requirements and products. Robotics and Computer Integrated Manufacturing, 14(4): 341-352.

Zeng, Y. et al., 2004. Mathematical foundation for modeling conceptual design sketches. Transactions of the ASME: Journal of Computing and Information Science in Engineering, 4(2): 150-159.

Zeroual, K., 1989. An approach for automating the specification-acquisition process, In Proceedings of the Second International Workshop on Software Engineering and Its Applications, Toulouse, Nanterre, France, pp. 349-355.

Zhu, S., Yao, S.J. and Zeng, Y., 2007. A novel approach to quantifying designer's mental stress in the conceptual design process ASME DETC/CIE, Las Vegas, Nevada, USA, pp. DETC2007-35887.

Zielczynski, P., 2007. Requirements Management Using IBM Rational RequisitePro. IBM Press.