1-1-2012

Development of a computer-aided design supporting system for transformative product design

Keunho Choi
Wayne State University,

DEVELOPMENT OF A COMPUTER-AIDED DESIGN SUPPORTING SYSTEM FOR TRANSFORMATIVE PRODUCT DESIGN

by

KEUNHO CHOI

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2012

MAJOR: INDUSTRIAL ENGIENERING

Approved by:

____
Advisor Date

____

____

____

____

DEDICATION

I dedicate this dissertation to my wonderful family who loved and cared
me to collect sweet memories and had enjoyable moments in life.
Particularly to my thoughtful and patient wife, Yunsung,
who has put up with these many years of research, and
to our precious children Eden and Jayden who are the joy of our lives.

I must also thank my loving parents and my terrific in-laws
who have helped so much with baby-sitting and
have given me their fullest support, especially in prayer.

Finally, I dedicate my work to the friendship and memory of my church
members of Korean Presbyterian Church of Michigan, who supported and
encouraged my family to live far away from home in Jesus' love.

나의 사랑하는 가족에게 이 논문을 바칩니다.
특별히 박사과정 기간 동안 배려와 인내로 한결같이 옆에서 지원해 준
사랑스런 아내 윤성이와 우리 삶의 기쁨이 되어 준
두 자녀 이현이와 준현이에게 고맙습니다.
또한 끝까지 기도로 아낌 없이 지원해 주신
부모님과 장인, 장모님께 감사드립니다.
그리고 타향살이 미국생활에서 잘 정착하며 지낼 수 있도록
예수님의 사랑으로 보살펴 주신
미시간 한인 장로교회 성도님들에게도 감사드립니다.

ACKNOWLEDGEMENTS

In acknowledging my appreciation and thanks, I am most grateful to the Almighty God for His divine guidance and blessings, with which His has brought many wonderful people during the course of my research.

I received a lot of support from many people during the course of this research. I owe special thanks to my academic advisor, Prof. Kyoung-Yun Kim, for his advice, inspiration, and support in this work. My gratitude also goes to the entire faculty and staff of the Department of Industrial and Systems Engineering at Wayne State University, for the support and kindness they have shown me over the years. I thank Drs. Leslie Monplaisir, Kai Yang, Alper E. Murat, and Jon Brewster, for serving on my dissertation committee, and for providing invaluable advice and constructive criticism for my work.

I also would like to thank Prof. Ohbyung Kwon at Kyung Hee University, South Korea, for his encouragement and intellectual advice and support on my academic career and faith. My whole-hearted thanks go to all of the Computational Intelligence and Design Informatics (CInDI) laboratory members for their support and suggestions in this work.

Finally, I wish to thank to my parents who have always provided endless love throughout my life with their continuous prayer, and to my in-laws who have supported us with baby-sitting and faithful prayer. My intimate appreciation has been reserved for my wife Yunsung for all her love, firm and steady support, and continued prayers. Last but not the least, I would like to thank the little ones, Eden and Jayden. Thank you.

PREFACE

Modern product innovation is often delivered by fusing technologies from different domains. To meet the current speedy pressure of product innovation, the conceptual design is a very critical stage to develop an innovative product. While various research works have been reported, the industries still struggle to reveal promising concept (solution) principles for the emerging products. Traditional conceptual design tools have been supporting a single product, whether it is an incremental improvement or a radical innovation, not a cross-disciplinary technology or product. In addition, the concept (solution) principles should be able to combine unstable design working principles when encountering specification and/or requirement change in the conceptual design that may have a significant influence downstream to product development.

Transformative Product Design (TPD), that is a new design paradigm being introduced by this research, aims to design a new product from a combination out of multiple products, which have been developed. To achieve the transformation process, it is crucial to identify similarities (or dissimilarities) between products. Without this step, it would be virtually impossible to even determine a set of good candidate of existing products to combine. After recognizing the similarities, a transformative design concept should be determined, optimized and evaluated. However, current conceptual design tools could not support this paradigm.

As aforementioned, one of remarkable characteristics in the modern technology change is the transformation of an existing product with multiple cross-disciplinary technologies. According to the proposed design paradigm, TPD, this research presents methods to respond this phenomenon and a conceptual design supporting system to realize the paradigm. To facilitate autonomous computer assistance in conceptual design for transformative design, the following

research objectives should be fulfilled: (1) to represent a design in a systematic way to identify the similarities and dissimilarities between cross-domain design objects, (2) to generate and optimize transformative design alternatives to designers, and (3) to assist designers to explore the recommended design transformative design alternatives.

In TPD, transformation is a design process to produce a new product (i.e., transformative product) from a base product by adding and/or converting functions/features with reference products or technologies. To represent the relationships between functions and behaviors and between structures (i.e. forms) and behaviors, this research presents a new model, i.e., the interaction network. To effectively generate the interaction network, this research adopts functional requirements in design documents as the source of the interaction network by incorporating with the semantic technology. The proposed semantic approach aims to diminish the human intervention to a minimum. Given the interaction network, to accomplish the transformation process, this research employs the network models and develops a new concept detection method to provide a systematic identification of a design relationship between existing technologies and products by supporting cross-disciplinary collaboration among various stakeholders. Finally, this research aims to present a new transformative conceptual design generation method based on the select concepts generated by the proposed concept detection technique as well as the corresponding evaluation method for the transformative design alternatives. The proposed methods will be modularized in the proposed prototype designer system to enhance collaboration in design process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Technology Fusion

To achieve cross-disciplinary technology advances, Kodama suggested technology fusion approach, which is described as a type of innovation that leads to advanced functions by combining at least two or more existing technologies into hybrid technologies [Kodama, 1986]. He addressed two possible approaches to (definitions of) R&D: (1) breakthrough approach and (2) technology fusion approach. The former focuses on the replacement of an older generation of technology. Due to the characteristics of the approach, it takes the strategy of technology substitution, which is linear and step-by-step. For instance, semiconductor has been invented to improve vacuum tube and Blu-ray Discs is replacing DVDs and CDs as CDs had replaced record albums.

On the contrary, technology fusion approach takes nonlinear, complementary and cooperative strategy of technology substitution. Nonlinear strategy means neither sequential development nor second-generation type development. Thus it excludes routine design or incremental design. The second strategy takes into account the characteristics of current existing technologies in terms of compatibility. In other words, innovative technology would depend on how much they are compatible with the existing technologies. It would be not successful when a technology is not acceptable in a society even if a technology is innovative and improved in the technology perspective. The last strategy emphasizes the importance of collaboration across disciplinary. Not only are the cross-disciplinary (technology) developers important, but also the end users of a technology should be considered all the time during the development phase. The main purpose of technology fusion is to provide the right product that customer wants, by integrating existing technologies, not to develop simply an innovative product that customer might not want.

In a sense, he emphasized the significance of between-industry sectors for technology fusion. While mechatronics and optoelectronics in the manufacturing industry are examples of traditional research within an industry sector, he provided a successful example of technology fusion with a carbon fiber product, which is a golf club shaft initially developed by Toray Inc. Later, the company had participated in multidisciplinary research projects for structural airframe and developed a fourth generation carbon fiber based material for airframe structure of Airbus A320. However, he just mentioned the necessity of technology fusion for the entrepreneurial strategic competitiveness.

1.2. Importance of Technology Fusion in Design and Limitations of Current Innovation Approach

Recently, hybrid technology has become a promising research domain in industry that has been facing the challenges such as sustainability and alternative energy technology. “One technology, one industry” no longer applies. Cross-disciplinary technology advances are noticeable phenomena in modern technology and products.

The Diffusion of Innovations theory considers how a society adopts a technology as a characteristic of innovation rather than how improved a technology is [Rogers, 1995]. To realize the success of design for cross-disciplinary technology advances, an innovative design should be evaluated from a societal perspective, instead of being evaluated from the perspective of product functions or technology advances. Innovative products, having generally greater relative advantages, compatibilities, and fewer complexities, will be adopted more rapidly than other innovations [Sosa and Gero, 2007].

Current conceptual design tools tend to have been developed to support the relative advantage and reduced complexity of product design and to focus on a breakthrough approach to solve a specific engineering problem or on radical innovation. Many conceptual design tools have supported designers during initial information gathering stage, creating functions and structures or optimizing design in later design stage where the concept finalized [Cari *et al*., 2005; Gupta and Okudan, 2008]. In recent times, researchers have tried to support automated concept generation such as A-Design that can produce multiple concepts using design agents [Campbell *et al*., 1999] and approaches using genetic algorithm (e.g. Jin and Li, 2007). However, an individual technology advance does not guarantee the success of an innovative product. Even if they put their efforts toward describing design activities in an object world to generate conceptual design, they are not satisfactory to assist cross-disciplinary technology advances. However, the compatibility characteristic has often been overlooked. Leon (2009) addressed the importance of compatible innovation because customers or societies do often not accept radical innovation. Compatible innovation has been reported as a method to improve the acceptance level of the innovation.

Other researchers have also addressed the importance of compatibility issues for innovation. While original ideas may require freedom at many levels, constraints such as compatibility with previous solutions and infrastructures can benefit creativity and innovation [Simonton, 2004; Partridge and Rowe, 1994]. The adoption of new ideas may increase as they mature. High quality and commercial success are usually found not in radical innovative ideas, but in more compatible modifications [Petroski, 1992]. High compatibility may cause technological innovations to be more successful due to higher degrees of adoption and diffusion [Rogers, 1995]. There is a high possibility for hybrid technology to succeed in a society in comparison with a radical approach toward a technology advance. As reported by Kodama (1986, 1992) and No and Park (2010), breakthrough approaches and radical innovation are often inadequate on their own to ensure the success of a design. Furthermore, cross-disciplinary technology advances are challenging and difficult to achieve by the breakthrough approach and radical innovation.

1.3. Existing Research in Conceptual Design

The conceptual design is critical, particularly in designing a new and innovative product or generating a completely new design for an existing product [Wang *et al*., 2002]. Conceptual design begins with high-level descriptions of requirements and continues with a high-level identification of a solution [McNeil *et al*., 1998]. In other words, conceptual design starts with formulation of abstract ideas with approximate concrete representations [Takala, 1989]. Then the generation of ideas is eventually evaluated against general requirements. With additional data incorporated later, decisions will be allowed to select design alternatives.

There have been a number of methods and techniques for representing conceptual design. Basically, many conceptual design representation methods establish functions and structures. Kitamura *et al*. (2004) define functional models as a representation of a part of (but not all of) the designer's intentions, so called design rationale. Umeda and Tomiyama (1995) consider function as a bridge between human intention and physical behavior of artifacts. They state that "there is no clear and uniform definition of a function, and moreover, it seems impossible to describe function objectively." Because of the subjective characteristics of functions, functions play an intermediate role between intention and objective.

Another approach is to regard function as a relationship between input and output of energy, material, and information, and this definition is widely accepted in design research [Pahl and Beitz, 1988; Welch and Dixon, 1992]. This research aims to follow the functional representation modeling approach to understand conceptual design and proposes a new, network-based representation model for conceptual design in order to support the technology fusion approach for cross-disciplinary technology advances in product design in a systematic manner.

1.4. Current Conceptual Design Tools

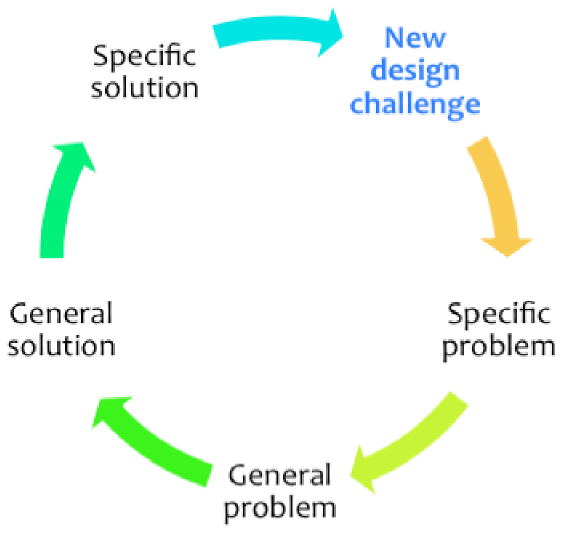
Conceptual design activities mainly consist of the following design processes. First of all, to facilitate computer-aided design tools, design rationale and design thinking should be translated and enhanced to computers from designers. Thus there has been challenging, for example, how to investigate cognitive behavior [Huang, 2008], how to capture design rationale [Bracewell *et al*., 2009], and how to stimulate creative design thinking [Salah and Abdalla, 2008]. Secondly, with the captured design intention, computer aided tools for conceptual design should be able to provide design alternatives. For the purpose of suggesting design alternatives, various research contributions have been recognized. Function primitives help designers describe and classify functions and the relations to other concepts in product design. One of objectives of researchers in functional modeling (FM) is to develop function primitives varied in terms of the domain of applications. (e.g., design of mechanics [Chandrasekaran, 2005]). Qin *et al*. (2000) presented a way of capturing sketching intentions and generating the corresponding 2D geometric primitives. To analyze multiple purposes of conceptual design due to the non-determined characteristics, multi-attribute utility analysis has been developed [Malak Jr. *et al*., 2009].

Most common techniques used in the conceptual design include problem solving strategies, genetic algorithms, case-based reasoning, and agent technology. Wang *et al*. (1994) applied a DAER (design-analysis-evaluation-redesign) model for conceptual design, combining numerical calculation with symbolic reasoning. Some researchers identified the fact with the help of machine learning, that product developers must, at an early design stage, take into account all the life-cycle concerns such as manufacturing, reliability, marketing and distribution, to achieve highest return on investments. Using agents and machine learning techniques, Co-Designer partially realized the requirements [Hague and Taleb-Bendiab, 1996]. At conceptual stage, information is very fuzzy and incomplete, which makes the design process quite difficult and challenging. Genetic algorithms or evolutionary algorithms have been well proposed to deal with this challenge, aimed at assisting designers during the end of the conceptual design stage and dealing with often vague and imprecise information [Renner and Ekart, 2003; Leon, 2004; Ng and Leng, 2002; Bentley and Wakefield, 1996].

Current conceptual design tools have been proposed to include several tasks: search for working principles; combine working principles; select suitable combinations; and firm up into principle solution variants [Pahl and Beitz, 1988]. To support such tasks, conceptual tools have developed various and successful optimization methodologies and problem solving techniques with computational intelligence. However, they have mainly taken the *problem-solving* approach to conceptual design, while the technology fusion approach tends to seek a new application area that can apply existing technologies. In order to overcome the limitation of current conceptual design tools and techniques for technology fusion, a new design paradigm should be taken into consideration. This research presents a new design paradigm in the following section.

1.5. Transformative Product Design

In order to meet the cross-disciplinary technology advances and product design, this research presents a new design paradigm, called Transformative Product Design (TPD). TPD aims to design a new product from a combination out of multiple products, which have been developed. Current innovation theories, such as TRIZ, OTMS-TRIZ, Axiomatic Design, General Theory of Innovation, Mind Mapping, Brain Storming, and Lateral Thinking, provide ways of the generation of creative design concepts involving resolution by elimination of contradictions [Leon, 2009]. They determine the current design approach as an inventive way from identifying a specific design problem through abstracting it as a typical problem and finding an associated typical solution and to finally generating specific solution. Figure 1.1 (a) shows the problem solving process in the traditional product design. In contrast, transformative product design already has a specific solution in a specific domain. TPD is required to combine two different complete design models into hybrid product models with various alternatives. Figure 1.1 (b) shows the design process in transformative product design.



(a) Traditional product design (b) Transformative product design

Figure 1.1 Design approach comparison

Transformative product design is to focus on combination of existing technologies or products (specific solutions) into hybrid technologies or products (for a new problem). Assuming that existing technologies or products are represented in individual design knowledge that is not easily shared with others, it is too difficult to see any relationship between technologies or products. To achieve TPD, it is significant to discover similarities (or dissimilarities) out of existing technologies or products. Otherwise, it might be hardly to determine a set of good candidate of existing technologies to combine. However, current conceptual design tools have focused on the design process within a design team (or unit) having a complete knowledge of products such as functional catalogs, product platform, modular design, etc. [Erden *et al.*, 2008; Yoshioka *et al.*, 2004; Zha and Sriram, 2006]. Therefore, the research motivation comes from the demands of a new conceptual design supporting system for transformative product design.

1.6. Research Objectives

The contribution of this research is in many folds. A major contribution of this research focuses on the development of a computer-aided conceptual design decision supporting tool that fulfills the proposed transformative product design paradigm with the following questions: (1) how a system systematically understands design knowledge in consideration of teleological knowledge and physical artifacts by reducing the human intervention to a minimum in an efficient manner, and (2) how a system systematically transforms a product from existing design knowledge. To answer the research questions, the objectives and the importance of this research are outlined as follows:

(a) Many researchers have contributed to develop various design representation models that can understand teleological knowledge and physical knowledge of products; however, there are few methods to systematically represent design knowledge with the aid of computers so that human efforts decrease to a minimum. Transformative product design, by nature, has to take cross-disciplinary domain knowledge into consideration. This challenge is not a trivial task for domain experts, who have their strengths on their own specialties, to understand things in different domains. In other words, it is difficult for them to digest design knowledge and extract teleological knowledge and physical knowledge of products in different domains. The first objective of this research is to systematically capture existing product design knowledge from functional requirements in an informal format in a natural language and translate into a machine-readable design representation model, which is called *Interaction Network*, in consideration of the two perspectives of design knowledge.

(b) The second objective of this research aims to provide semantics for the design transformation process in TPD. One of key characteristics in TPD is *multi-disciplinary*. A designer is not able to provide appropriate keyword to search design knowledge in different domains. In other words, designers might be frustrated to think of a way beyond their expertise. Moreover, if designers select any technology advance in a product for the design transformation, it is not a simple process to identify which other functions would be related to the select technology advance. In order to fulfill this research challenge, a proposed similarity score aims to compare two phrases or sentences and to measure the similarity between the two.

(c) One of the biggest challenges and the beauty of this research is the design transformation. The ultimate goal of transformative product design is to guide existing technology advance(s) to a new direction of the design concepts across product domains. As briefly introduced in the previous sections of this chapter, current conceptual design tools have focused on a single product innovation or improvement. There have been limitations to deal with cross-disciplinary design consideration. Attention is given to this challenge for transformative product design. By incorporating with the above two research objectives, this research also presents a novel algorithm to systematically compare the select functional concept (or *technology advance*) for the design transformation with other product designs across domains. In other words, this algorithm gives the capability of exploration of new applications of existing design solutions in consideration products in different domains.

1.7. Scope of Research and Limitations

This research develops a computer-aided design tool for a conceptual design according to a newly proposed design paradigm, which has never been introduced in research and industries. To facilitate the proposed design paradigm, it is necessary to understand customer needs, functions of products, and forms of products. Among the three important characteristics for product design and development, this research focuses on the functional perspective for the design transformation. Transformative product design, by nature, identifies a new direction of existing technologies or product, instead of targeting to minimize the gap between customer needs and current design solutions. Consequently, to lead a conceptual design alternative for the design transformation, this research considers the teleological perspective as a starting point for the concept generation in order to determine a set of new directions of existing technologies or products. Meanwhile, the *Interaction Network* based design representation is able to address physical knowledge of products too. This might be able to enhance this research later for the further improvement by taking a different approach to the concept generation for the design transformation from the view of object world. For the semantic capability of the proposed system, this research does not implement a new semantic network for a natural language, but adopt a representative semantic English network, i.e., Word-Net, since this research aims to show the feasibility of the semantic approach to the design transformation. Consequently, the semantic performance totally depends on the performance of Word-Net.

1.8. Dissertation Layout

The remainder of this research is structured as follows:

Chapter 2 summarizes the literature review related to conceptual design and natural language processing with object identification methods in software engineering.

Chapter 3 presents a new model, i.e., *Interaction Network*. To effectively generate the interaction network, this research adopts functional requirements in design documents as the source of the interaction network by incorporating with the semantic technology.

Chapter 4 develops a new *concept detection* method to provide a systematic identification of a design relationship between existing technologies and products by supporting cross-disciplinary collaboration among various stakeholders.

Chapter 5 presents a new *transformative conceptual design generation* method based on the select concepts generated by the proposed concept detection technique. Transformation in TPD is a key design process to produce a new product (i.e., transformative product) from multiple products by adding and/or converting functions/features.

Chapter 6 modularizes the proposed methods in a proposed prototype designer system to enhance collaboration in design process, analyzes the system development requirements, and finally implements a computer-aided design supporting system for transformative product design.

Web technologies, such as Java and Apache Tomcat Server, NLP tools with Java APIs, and database technology have been employed.

Chapter 7 validates how the prototype designer system satisfies the proposed transformative product design paradigm according to the convergence types classified by Stieglitz (2003).

Chapter 8 concludes the research by highlighting the contribution of this research to future innovative design and discussing future works for the further improvement.

1.9. Road Map

Chapter 2 gives readers the research background of this research, including conceptual design and object identification methods in software engineering. Readers might skip this chapter to go direct to the proposed methods for the transformative product design paradigm, which are presented in Chapters 3, 4, and 5.

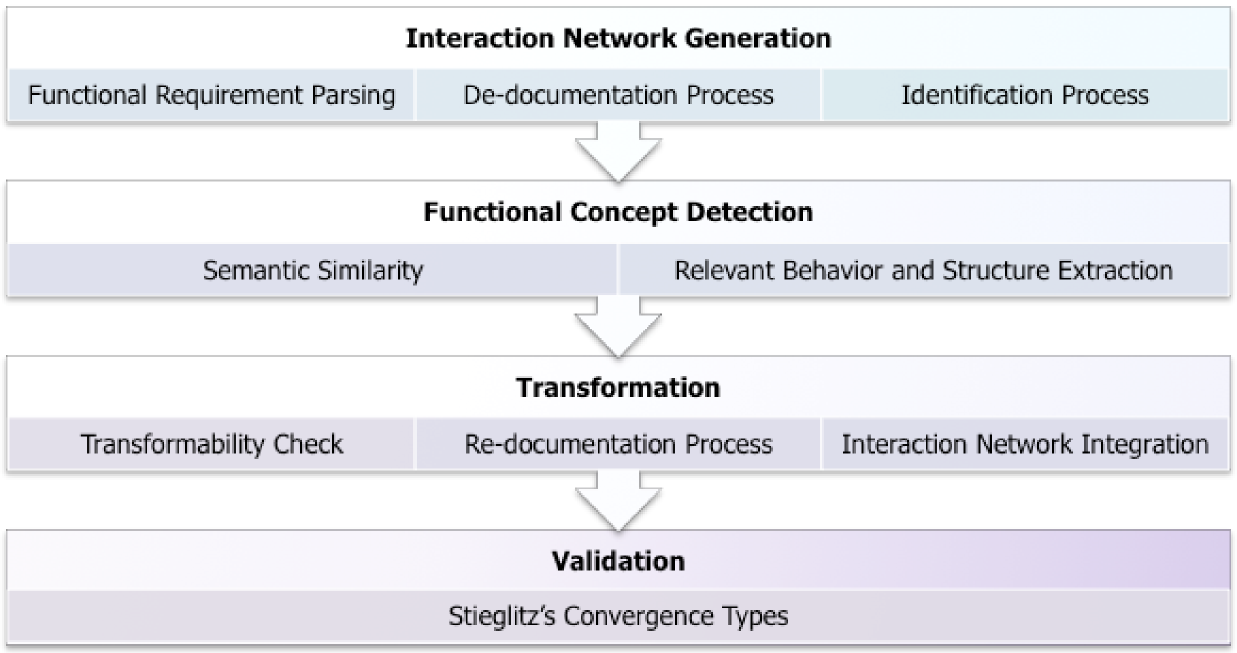

Interaction Network Generation
Functional Requirement Parsing
De-documentation Process
Identification Process
Functional Concept Detection
Semantic Similarity
Relevant Behavior and Structure Extraction
Transformation
Transformability Check
Re-documentation Process
Interaction Network Integration
Validation
Stieglitz's Convergence Types

Figure 1.2 Research process for Transformative Product Design (TPD)

For better understanding, it is recommended to read Chapters 3, 4, and 5 in their sequential order. The above figure illustrates the main flow of this research from Chapter 3 to Chapter 6 with detail processes. Chapters 6 and 7 are stand-alone and can be read in any order. Lastly, Chapter 8 should be read when all the other chapters have been read as it summarizes the research contributions and future direction.

CHAPTER 2

RESEARCH BACKGROUND AND LITERATURE REVIEW

2.1. Conceptual Design

The conceptual design is critical, particularly, in designing a new and innovative product or generating a completely new design for an existing product [Wang *et al*., 2002]. Figure 2.1 shows the design opportunity of high impact in early design stage.

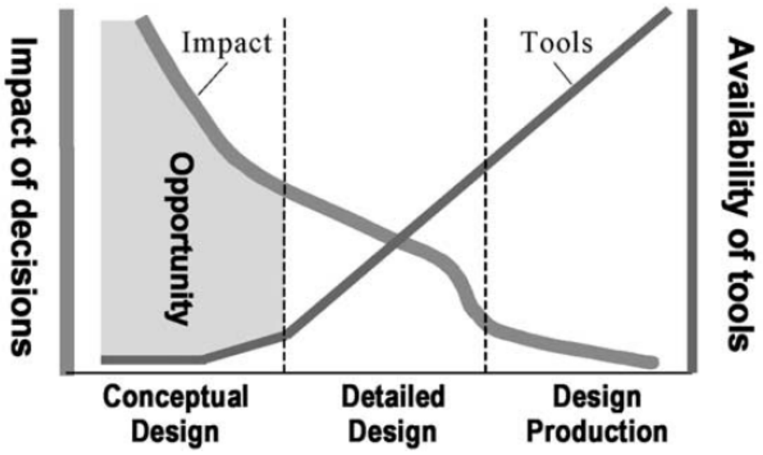


Figure 2.1 Opportunity in early design stage [Wang *et al*., 2002]

McNeil *et al*. (1998) consider conceptual design as a process of identifying high-level descriptions of requirements and seeking a high-level of a solution. Formulation of abstract ideas with approximate concrete representations is the very beginning process of conceptual design [Takala, 1989]. Then designers eventually evaluate the generation of ideas against general requirements. Design alternatives will be determined with additional data incorporated later.

To represent conceptual design, a number of methods and techniques have been reported. They are basically established on the basis of functional approach to conceptual design. Functional models can be defined as a representation of a part of (but not all of) the designer's intentions (i.e., design rationale) [Kitamura *et al*., 2004]. A function can be considered as a bridge, which plays an intermediate role, between human intention and physical behavior of artifacts [Umeda and Tomiyama, 1995]. Umeda and Tomiyama (1995) also highlighted the subjective characteristics of functions, given the fact that there is no clear and uniform definition of a function. Another approach toward functional models is to consider function as a relationship between input and output of energy, material, and information [Pahl and Beitz, 1988; Welch and Dixon, 1992]. This approach has been widely accepted in design research.

Transformative product design aims to support designers in a systematic way by improving efficiency in order to transform existing products into a new application. To do so, design knowledge extraction for the transformation process is definitely necessary and is not a trivial task for designers. In fact, this task is very cumbersome, erroneous, and even highly dependent on designers' understanding of design knowledge. To fulfill the demands of the systematic design knowledge extraction, the following section studies the details of the aforementioned research approaches and current conceptual design tools first. Next, this chapter reviews natural language processing techniques in order to present a systematic method for identifying representation model by reducing human intervention to a minimum.

2.1.1. Design Modeling

Engineering design can be viewed as an articulate process composed of phases, where each phase represents a combinatorial action on the parts the composite object is constituted of [Colombo *et al.*, 2007]. Various perspectives could differentiate engineering design. Before addressing any specific design modeling method in conceptual design in this section, epistemological categories of knowledge have been studied, which consists of mainly two types of design knowledge: subjective vs. objective knowledge (Figure 2.2).

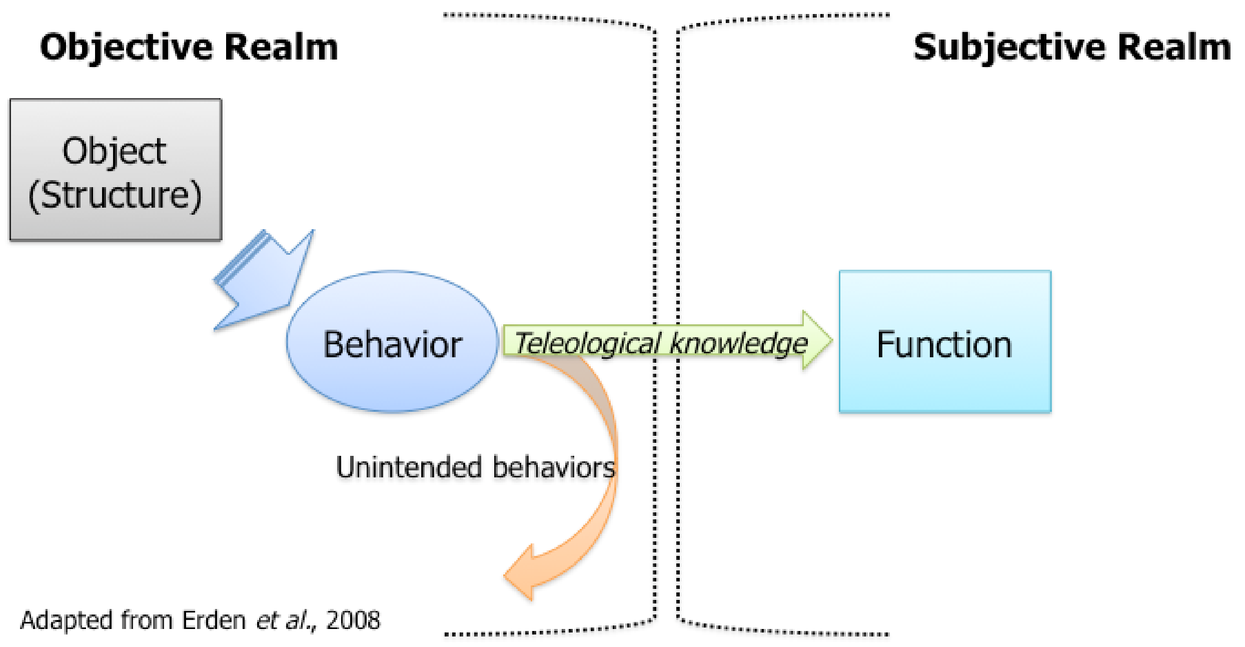


Figure 2.2 General view of function modeling in design engineering

The former one represents teleological knowledge, which is knowledge about purpose. Teleological knowledge has been represented in functional knowledge [Umeda and Tomiyama, 1997]. The later knowledge illustrates physical knowledge such as structures of physical existence and behaviors of physical principles [Brown and Chandrasekaran, 1986].

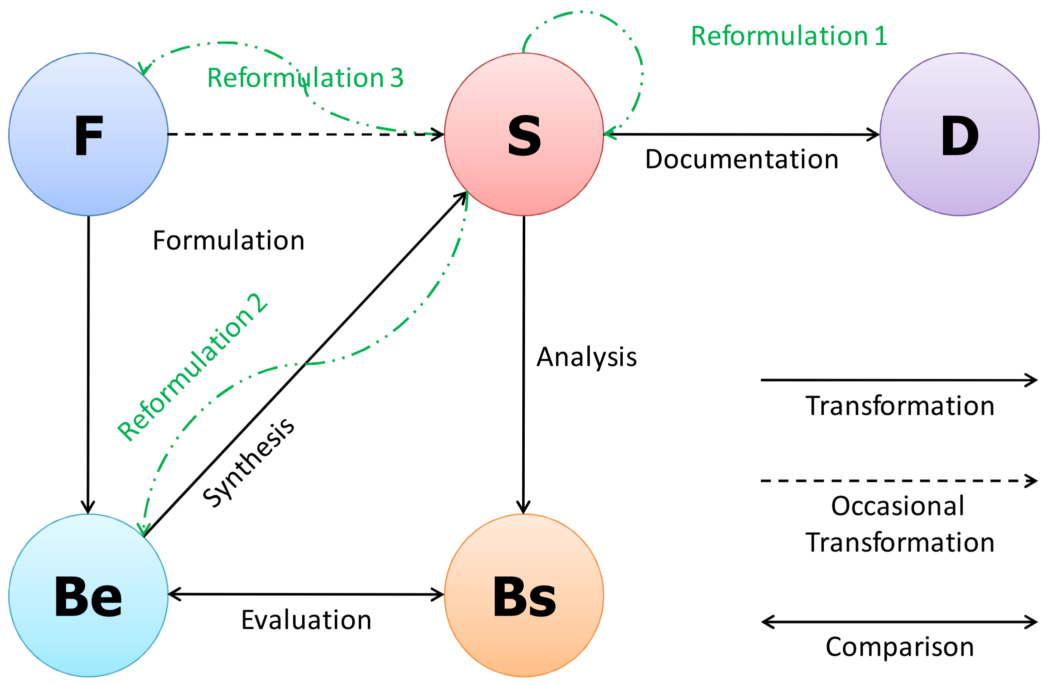
There are two representative design models based on the epistemological taxonomy of design knowledge: the Function-Behavior-Structure (FBS) model developed by Gero (1990) and the Function-Behavior-State model proposed by Umeda and Tomiyama (1995, 1997). Umeda address there is no meaningful distinction between state and structure; the difference between the two in conventional usage is just a matter of duration: structures that change in a short time are generally called states. By the definition, structure would embrace state so that Gero's FBS model will be considered for this research.

FBS (Function-Behavior-Structure) Model

Gero developed the function–behavior–structure (FBS) model in 1990, which encapsulated design process. He defines function as an intermediate between the goal of human and the behavior of a system. FBS model focuses more on the design process, which is a transformation from intentions to the structure.

Gero and Kannengiesser (2007) describes three high-level categories for the properties of a design object: (1) *Function* (F) of an object is defined as its teleology, i.e. "what the object is for," (2) *Behavior* (B) of an object defined as the attributes that are derived or expected to derived from its structure (S), i.e. "what the object does," and (3) *Structure* (S) of an object defined as its components and their relationship, i.e. "what the object consists of." The structure (S) of most objects can be described in terms of geometry, topology and material.

Figure 2.3 shows the FBS design process schema. In the Gero's design process schema, the purpose of designing is to transform function F (where F is a set) into a design description D in such a way that the artifact being described is capable of producing these functions. Thus, a naive model of design is F → D, where → is some transformation (this addresses simply a conversion process from the former to the latter, that is different from the design process in TPD). There is, however, no direct transformation capable of achieving this result. A design description represents the artifact's elements and their relationships, labeled structure S. Computer-aided design systems have become the means by which structure is transformed into a design description; that is, S → D, so called ***documentation***. A direct transformation from function to structure occurs at the element level of an artifact and is not considered designing; that is F → S.



Note. *F* represents Function, which is intended use. *Be* represents Expected Behavior. *Bs* represents Behavior derived from Structure. Structure *S* includes geometry, material and topology.

Figure 2.3 FBS model [Gero, 1990]

Function is defined as the relation between the goal of a human user and the behavior of a system. Gero classified behaviors in two ways: the behavior of the structure Bs (where Bs is a set), the expected behavior Be (where Be is a set). The first behavior is directly derivable from structure: S $\rightarrow$ Bs. This process is that of ***analysis***, which is structural knowledge (structural requirements on the predicted behavior), and presumes the description of which behaviors to determine. As for the second behavior, it provides the syntax by which the semantics represented by function can be achieved: F $\rightarrow$ Be. This process is that of ***formulation*** or *specification* in design, which is teleological knowledge.

The comparison between the predicted behavior of the structure and the expected behavior required to determine if the structure synthesized is capable of producing the functions is termed ***evaluation*** in design: Be $\leftrightarrow$ Bs, where $\leftrightarrow$ is a comparison. Here is another model of design: F $\rightarrow$ Be, Be $\rightarrow$ S(Bs). The function is transformed to expected behavior. Then this expected behavior is used in the selection and combination of structure based on a knowledge of the behaviors produced by this structure. This process is called ***synthesis***, which is behavioral knowledge (behavioral requirements on structure).

As shown in Figure 2.3, Gero classifies three different ***reformulation*** processes: (1) Reformulation Type One, which is a modification of the structural values decided for an object, (2) Reformulation Type Two, a modification of the behavioral requirements on structure, and (3) Reformulation Type Three, a modification of the teleological requirements on expected behavior (design object intended use).

There are many research works using the FBS model to provide design alternatives. For example, an analogical approach is a computational reasoning method to reason and select design alternatives by using characteristics of the FBS model. Gero and Maher (1991) present dependency networks and associated design prototypes. They employ analogical reasoning method based on genetic transformation such as mutation.

To implement analogical reasoning, Gero and Maher (1991) introduce two technical methods: (1) the memory indexing based on structure variables and (2) the transformation from one structure variable to another structure. The limitation of the dependency networks is to focus on structure variables. In practice, conceptual design is used to start from subjective characteristics of design, which are functions as well as the intention of products. Furthermore, to fertilize the transformative product design, designers have to be able to even start from uncertain design requirements. At this time, physical objects may not be available completely. Thus, it may be better to take a teleological approach to explore cross-disciplinary technology advances.

Additionally, the FBS model considers still design process within a single product development. All the three reformulation processes are related to the early phase before the design documentation. On the other hand, the transformative product design assumes that design knowledge is documented already. For the transformation process in TPD, it is required to reuse the existing design knowledge. Consequently, a new design representation has to be able to embrace the reverse design engineering from the documentation. Interestingly, there is a few research reported in such a reverse manner. Goel *et al*. (2009) used the structure-behavior-function (SBF) model as a programming language for automated design and problem solving. However, they developed too strictly structured modeling language to stimulate the design transformation process in consideration of the proposed design paradigm in this research.

Functional Interrelationship

Pahl and Beitz (1988) define function as the general input/output relationship of a system whose purpose is to perform a task. A function becomes an abstract formulation of a specific task, independent of any particular solution. It represents a flow of energy, materials, or signals (information). Functions are decomposed into sub-functions and usually have their descriptions in the *verb-plus-noun* form. The decomposition is presented as a block diagram of sub-functions (see Figure 2.4).

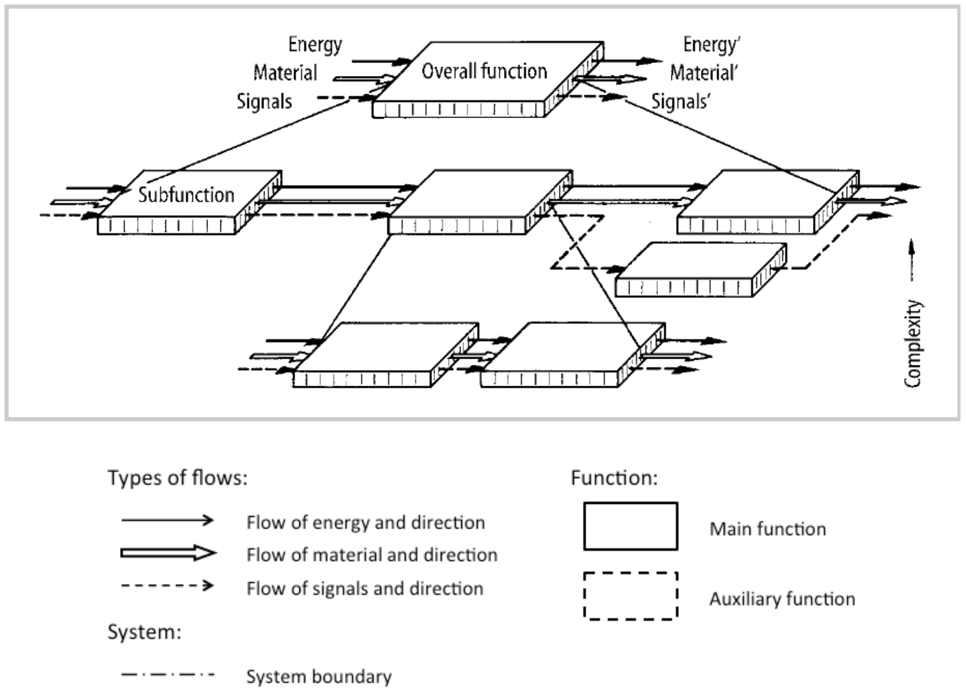


Figure 2.4 Example of establishing a function structure [Pahl and Beitz, 1988]

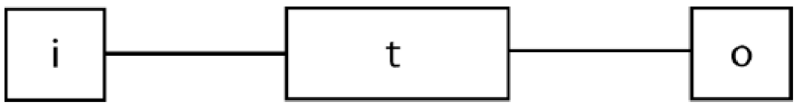


Figure 2.5 The structure of a process

Gero and Kannengiesser (2007) describe the function diagram of input/output and transformation as process structure. They identified the structure into mainly components and relationships. The components are an input (i), a transformation (t) and an output (o). The relationships connect the input and the transformation (i – t) and the transformation and the output (t – o) (see Figure 2.5).

In this structure of a process, the input (i) and the output (o) structure elements represent properties of other entities in terms of their variables and/or their values. In a sense, they classified two different processes: (1) homogeneous process and (2) heterogeneous process. The former represents that the input (i) and output (o) contain the same variables with the only change of their values. The later, in contrast, use disparate variables as input (i) and output (o). For the homogeneous process example, the process of transportation changes only the values for the location of a physical object (e.g. the values of its *x*-, *y*-, and *z*-coordinates). For the heterogeneous, the process of electricity generation takes mechanical motion as input (i) and produces electrical energy as output (o). In such a function block diagram, input and output may refer not only to (properties of) objects but also to (properties of) other processes. Other examples than the function block diagram are Functional Analysis Systematic Technique (FAST) diagrams, which forms a hierarchical tree to represent functional relationships, functional flow charts and functional logic diagrams [Sturges *et al.*, 1993].

The function block diagram approach is well applied to solve a complex problem by decomposing it into simpler sub-problems. It is also valuable to understand the subjective perspectives of a system with different points of view (i.e., material, energy, and signal). However, it has been limited to cope with the perspective of objective artifacts in design engineering. Thus, even if this approach has been adopted in many research works in design supporting tools, it is not appropriate to analyze the existing design documents that may include subjective and objective design worlds. Therefore, this research concentrates on the design representation model according to the epistemological taxonomy rather than the functional interrelationship-based design representation.

2.1.2. Conceptual Design Tools

The early or conceptual stage of the design process is dominated by the generation of ideas, which are subsequently evaluated against general requirements criteria [Wang *et al.*, 2002]. As shown in Figure 2.1, there are many opportunities in early design stage that can impact significantly on later design phases with relatively less cost. As more tangible evidences of function are derived, additional data is incorporated later so as to allow better decisions to be made between competing alternatives. Current conceptual design tools have tried to support this challenge to maximize the utilization of the opportunities for better decisions.

Conceptual design activities mainly consist of the following design processes. First of all, to facilitate computer-aided design tools, design rationale and design thinking should be translated and enhanced to computers from designers. Many researchers have come up with the challenging issues, for example, how to determine functional interdependencies in consideration of customers' needs [McAdams *et al.*, 1999], how to investigate cognitive behavior [Huang, 2008], how to capture design rationale [Bracewell *et al.*, 2009], and how to stimulate creative design thinking [Salah and Abdalla, 2008].

Secondly, with the captured design intention, computer aided tools for conceptual design should be able to provide design alternatives. For the purpose of suggesting design alternatives, various research contributions have been recognized. Function primitives help designers describe and classify functions and the relations to other concepts in product design. One of objectives of researchers in functional modeling is to develop function primitives varied in terms of the domain of applications. (e.g., design of mechanics [Chandrasekaran, 2005]). Qin *et al.* (2000)

presented a way of capturing sketching intentions and generating the corresponding 2D geometric primitives. To analyze multiple purpose of conceptual design due to the non-determined characteristics, multi-attribute utility analysis has been developed [Malak Jr. *et al.*, 2009].

Design repository is one of major issues in design engineering to capture and represent design knowledge. For conceptual design, research works have focused on functional modeling. Recent representative examples of functional basis approach are ontology-based functional formulations [Garbacz *et al.*, 2011] and formal structures for design information such as database [Bohm *et al.*, 2008]. However, these contributions turned out to be intensively related to teleological knowledge with the lack of consideration of objective knowledge. To facilitate existing design knowledge comprehensively, a conceptual design tool is also required to recognize the physical artifacts of product design. Otherwise, design knowledge analysis would be restricted to partial perspective of products.

Current conceptual design tools have been proposed to involve several tasks: search for working principles; combine working principles; select suitable combinations; and firm up into principle solution variants [Pahl and Beitz, 1988]. To support such tasks, conceptual tools have developed various and successful optimization methodologies and problem solving techniques with computational intelligence. Most common techniques used in the conceptual design include problem solving strategies, genetic algorithms, case-based reasoning, and agent technology. Wang *et al.* (1994) applied the DAER (design-analysis-evaluation-redesign) model for conceptual design, combining numerical calculation with symbolic reasoning.

Some researchers identified the fact with the assistance of machine learning, that product developers must, at an early design stage, take into account all the life-cycle concerns such as manufacturing, reliability, marketing and distribution, to achieve highest return on investments. Using agents and machine learning techniques, Co-Designer partially realized the requirements [Hague and Taleb-Bendiab, 1998]. At conceptual stage, information is very fuzzy and incomplete, which makes the design process quite difficult and challenging. Genetic algorithms or evolutionary algorithms have been well proposed to deal with this challenge, aimed at helping designers during the end of the conceptual design stage and managing often vague and imprecise information [Renner and Ekart, 2003; Leon, 2004; Ng and Leng, 2002; Bentley and Wakefield, 1996].

Current approach toward conceptual design tools and methods resulted from the assumption that the identification of new design challenges is the starting point of a new design. Consequently, they aimed to propose methods and strategies to solve their design challenges covering the life-cycle concerns. However, for transformative product design, it is more imperative to present a new method in order to comprehensively capture existing design knowledge and to discover different, possible directions of the existing design to new applications. In other words, while current conceptual design tools and methods have focused on the *problem-solving* process after identifying design challenges from existing products, transformative product design pays further attention to the *discovery of various new applications* of existing design solutions.

2.1.3. Collaborative Conceptual Design

When a product is designed through the collective and joint efforts of many designers, the design process may be called Collaborative Design (also be called Co-operative Design, Concurrent Design and Interdisciplinary Design) [Wang *et al.*, 2002]. This would include those functions as disparate as design, manufacturing, assembly, test, quality and purchasing as well as those from suppliers and customers [Spow, 1992]. Traditional design systems have used a sequential model for design generation, which breaks the design task into sub-tasks that are serially executed in a predefined pattern. This design pattern has been changed since researchers found that sequential design is brittle and inflexible and often requires numerous iterations, which make the design expensive and time-consuming, and also limit the number of design alternatives. Collaborative design tries to address these problems concurrently. A number of technologies including distributed objects, agents and the Internet and Web technologies have been proposed to implement collaborative design systems.

In the late 1990's, the Internet and Web technologies had rapidly been emerging in the market as well as had been applied to design supporting tools with collaboration. Web-based collaborative design tools should have the following primary functions [Wang *et al*., 2002]:

- Access to catalogue and design information on components and sub-assemblies
- Authenticated access to design tools, services and documents
- Communication among multidisciplinary design team members in multimedia formats

Most Web-based collaborative design systems in the emerging era of the Web technologies are developed using Java, whereas few others are developed by the use of Common Lisp (e.g. WWDL by Zdrahal and Dominque (1997)), Prolog (e.g. WebCADET by Rodgers *et al.* (1999)). HTML and Java Applets are widely used for developing the client side user interfaces, in addition to VRML. A number of Web-based design systems use VRML as a neutral representation in their geometric models [Allen *et al.*, 1999; Roy *et al.*, 1997; Klein, 1997]. Saridakis *et al* (WebCo2Des, 2005) has proposed recent research works related to conceptual design. A web-based integrated system for collaborative product development and an interactive real-time design platform have also been published [Tseng *et al.*, 2008; Liang and Wei, 2006].

In sum, collaborative design tools have been well developed with various information and graphic technologies. Current collaborative design tools have supported the collaboration with designers. To generate conceptual design alternatives of a product, design activities have been classified into several sub-tasks and multi-disciplinary specialties of designers are incorporated with collaborative technologies. However, current research achievements of collaborative design tools have also been indicated to focus on design activities of a domain-specific engineering design problem and the associated detail design optimization and selection. To provide a new collaborative environment for transformative product design across multi-disciplinary stakeholders, this research presents a web-based conceptual design supporting system that stimulates a general environment, where cross-disciplinary designers can work together, and eventually is able to integrate with graphical design tools for physical properties of product design.

2.1.4. Summary

In conceptual design, there have been two representative functional modeling methods: Function-Behavior-Structure and Function Block Diagram. The two representative models are appropriate to represent conceptual design that highlights teleological knowledge of products. The former covers two different perspectives of design engineering, i.e., subjective and objective views of products. However, design processes described in the model do not specify the steps to obtain design knowledge from design documentation in a reverse manner, which is a key characteristic to realize the transformative product design in order to understand existing design knowledge. Meanwhile, the latter model is limited in teleological knowledge representation of product design so that it is not appropriate to translate possible physical artifacts from existing design knowledge. Hence, this research adopts the former approach to extract design knowledge that covers two perspectives of design.

Current conceptual design tools also follow the traditional philosophy of designing by the identification of new design challenges. As aforementioned in the design processes of FBS, most conceptual design tools aimed to propose methods and strategies to solve their challenging problems. However, the transformative product design is more likely to pay attention to the knowledge discovery from the existing design, which can direct to different, possible new applications across domains.

To propose a design representation model for transformative product design, this research reverses the design processes presented in the FBS model to acquire existing design knowledge from design documentation, which takes a different approach by identifying new applications from existing design solutions. In addition to that, to incorporate multiple design stakeholders and stimulate a general environment for cross-disciplinary design, not focusing on any specific design domain, this research develops a web-based prototype system for conceptual design support. To enable these objectives, this research adapts the object identification methods in software engineering by applying natural language processing technique, reviewed in the following section.

2.2. Requirement Analysis with Natural Language Processing

To facilitate conceptual design by analyzing existing design knowledge, a documentation of functional requirements is a good option to start with. First of all, to promote conceptual design for transformative product design, it is necessary to understand functional design of products as well as objective artifacts of products. Since transformative product design requires a reverse design engineering to extract design knowledge from documented design, the digitalized drawings can be an alternative to start with. However, such drawings have been reported to lack of design intentions or rationale easily [Kim *et al*., 2006; Kim *et al*., 2009]. In other words, it does not encounter any functional perspectives of the design. Hence, design drawings are not a good candidate for the reverse design engineering.

Functional requirements are well documented to describe products in the two different perspectives of design. They contain the teleological knowledge of the design and object artifacts of it. Additionally, functional requirements require less complicated description in details of design knowledge than detail designs. At least, functional requirements are not necessary to describe geometric information of designs that may delay the design decision time. To promote conceptual design for the transformative product design, therefore, a functional requirement document can be a good starting point.

The challenging issue of using a functional requirement document is the analytical process of understanding natural languages, for example, English. This section studies the natural language processing (NLP) techniques that have recently become promising research tools for the informal text analysis as NLP tools have been significantly improved for the last decade [Elbendak *et al*., 2011].

2.2.1. Natural Language Processing

Natural language processing (NLP) technologies have been adopted in various fields of computer science, artificial intelligence, information retrieval, and linguistics to enable the interaction between computers and natural languages [Bird *et al*., 2011; Lewis, 1996; Charniak and McDermott, 1985]. NLP aims to extract meaningful information from documents written in natural languages or to produce natural language based output [Warschauer and Healey, 1998].

One of promising NLP application area in software engineering is the requirement analysis [Elbendak *et al.*, 2011]. Specifically in software design, the requirement analysis process requires critical and difficult tasks due to the input written in a natural language such as English.

Elbendak *et al*. (2011) addressed many problems and difficulties often associated with the use of natural language (NL). They highlighted the ambiguity and complexity of NL as major problems in requirements specifications. Such challenging characteristics of NL may cause misunderstanding between the parties involved. Customer satisfaction with the implementation of requirement analysis is significantly associated with how much the understanding is clear. Furthermore, the significant cost difference of any errors, mistakes, or inconsistencies incurs at the specification analysis stage from the time at later phases, especially when a system has already been implemented. It has been reported that the cost difference to correct an error in the early stage compared with leaving it till the end is 1:100 [Boehm, 1983; Dong Liu *et al*., 2004; Li *et al*., 2005].

In the requirement analysis, another challenging issue when implementing NL techniques is the accuracy. Automatic identification of objects and classes with their relationships is possibly faster than manual identification: however, the automatic process may loss accuracy throughout the identification process. Moreover, there is no standard method for the automatic identification process. Some may think of standards in the requirement specifications such as UML (Unified Modeling Language) to assist better machine-interpretable specifications. The use case description may be more useful and effective than free form English sentences. However, they are still written in a natural language. The following sections study how the product design research applies the NLP techniques for what purposes and how the NLP techniques are employed in software design in order to support the object and the associated behavioral relationship identification process.

2.2.2. Natural Language Applications in Product Design

Ur-Rahman (2010) summarized possible applications by employing information retrieval and natural language processing techniques for industrial knowledge management, including product development. He highlighted the potential of discovering knowledge from semi-structured or unstructured database and disclosing useful patterns of information and knowledge in textual data formats. Yang *et al*., (2005) also highlighted the importance of information in the engineering design process. They also addressed that gaining access to design knowledge is critical to effective design decision-making.

Dong and Agogino (1997) addressed a challenging issue in sharing design information across functional workgroups, which is the complex expression of associations between design elements. The ultimate goal of their research was to reduce the complexity by means of a technique for inducing a representation of the design based upon the syntactic patterns contained in the corpus of design documents. They contributed to identify the association between the design and the representation for the design on the basis of the representation on terminological patterns at the design text. To realize the research methods, a dictionary of noun-phrases found in the text corpus was created, inter-term dependencies were discovered in clusters of words, and a Bayesian belief network described a conceptual hierarchy specific to the domain of the design.

Nanda *et al*. (2006) also pointed out the lack of a comprehensive and systematic methodology for constructing product ontologies. To develop ontologies for design artifacts, they proposed a methodology, Product Family Ontology Development Methodology (PFODM), to develop formal product ontologies using the Semantic Web paradigm. They focused especially on ontologies for product family design. Their research contribution resulted in a hierarchical conceptual clustering of related design artifacts so that they were able to share and reuse parts, processes, and intentional information by reducing complexity, lead-time, and development costs.

Researchers have reported the limitations of the use of ontologies for information sharing in design engineering [Dong and Agogino, 1997; Nanda *et al*., 2006]. They pointed out that it is more expensive than the design itself when formalizing ontologies before the design begins, establishing the knowledge sharing agreements, or mapping out the design hierarchy. We agree with their argument of the difficulties of ontology formulation and knowledge sharing and mapping. Therefore, to broadly utilize the prototype system for transformative product design, this research follows their research direction, especially Dong and Agogino’s approach, to extract design knowledge and any interactions with respect to the FBS model in the previous section.

Ahmed *et al*. (2007) employed natural language processing tools to build thesaurus of terms and refine integrated taxonomy for the proposed six-stage methodology that creates engineering ontologies. In the middle of the methodology, they consider the design document retrieval activities of designers by submitting natural language queries or navigating the ontology space. Their development of the engineering taxonomy aims to facilitate the capture and exchange of function information and to apply it to a broad variety of engineering artifacts. With natural language based descriptions, they consider verbs as functions and nouns as flows from the function list. The ultimate goal of the taxonomy development is to allow researchers and industry to create ontologies for their particular purpose and a thesaurus for the terms within the ontology.

Similarly, Yang *et al*. (2005) focused on the development of thesauri for design context. They examined several approaches to building thesauri, including manual and automated methods. Even if Yang *et al*. proposed a manual framework to index information in consideration of the evolutionary nature of design, they concluded that automated machine methods for constructing thesauri may provide a feasible, low overhead alternative for information retrieval in design context.

Yang (2009) addressed the quality of ideas in the process of design. Assuming that the idea generation is related to the conceptual design phase in product design, according to her research work, one possible approach to improving the quality of concepts is through increasing their quantity. She found statistically significant correlations between the quantity of brainstormed ideas and design outcome, some (but not all) correlations between the quantity of morphological alternatives and design outcome, and correlations between the volumes of dimensioned drawings generated during the early-to-middle phases of design and design outcome. Transformative product design requires a new design environment in an easy way of generating design knowledge to expedite the design transformation process. NLP tools are able to enable the requirement of TPD by systematically extracting design knowledge from unstructured design documents.

2.2.3. Object Identification Methods

In software design, the process of requirements identification is considered one of the most critical and difficult tasks. Most of the input to this process is written in natural languages, which are inherently ambiguous. Developers need to interact with users in their own language. Also they need to review and analyze documents written in a natural language. Many research works have reported different techniques in natural language processing systems that attempt to transform natural language into conceptual models are considered for their effectiveness [Elbendak *et al*., 2011]. Some examined the rules to convert English sentences into entity relationship (ER) and enhanced entity relationship (EER) diagrams to determine entity types, attribute types and relationship types.

In software engineering, they consider functional specifications come from users in their own language and the translation of the functional requirements into object/class is the responsibility of developers to implement the requirements. To efficiently identify objects/classes and relationships from requirements/specifications written in a natural language such as English and to reduce human errors in dealing with complex requests, they have employed natural language processing tools and techniques [Elbendak *et al*, 2011]. These techniques for either informal free text or the more formalized description (i.e., use case description), which can be considered functional description in product design, support the object-oriented analysis phase, which can be structure analysis in product design.

Elbendak *et al*. (2011) summarized many challenging issues and difficulties often associated with the use of natural language (NL) as follows:

- The ambiguity and complexity of natural language
- Errors, mistakes or inconsistences incurred at this stage, which can be highly costly at later stages
- The analysis process itself due to the input in a natural language
- Less accuracy of automatic identification of objects/classes and relationships
- Little or no adoption of standards for expressing requirements specification (RS) for the purpose of object identification

- No standard method for the automatic object identification from NL sentences
- No standard template for requirement specifications

According to these challenges, for the last few years, there was a pause in development. However, recently there are two factors that encourage a renewal of efforts in this area. The first factor is the dictionary advance with ontologies. Dictionaries have become more sophisticated, so that dynamic content targeted to a particular subject area can be used. Secondly, the syntactic capabilities of the NLP tools have enhanced. Now they are able to classify words in sentences (nouns, verbs, gerunds, etc.). The advanced dictionaries and the enhanced capabilities of the NLP tools made the synergy for the efforts in this area.

These word classes are utilized to determine the design knowledge with respect to teleological knowledge and physical knowledge in the proposed methods. Chiu and Shu (2007) reported that the use of language, specifically verbs, could stimulate concept generation. Stone and Wood (2000) also identified the combination of the relationship between verb and object as the function-flow interaction.

There have been reported to provide some rules for mapping natural language elements to object-oriented concepts. The initial study of this was Abbott (1983)'s first suggestion indicating the nouns as classes and objects while verbs denotes relationships. Chen (1983) developed rules for determining entity types. For example, an English gerund "corresponding to a relationship converted entity type gerund may indicate an entity type which is relationship converted entity type" in an entity-relationship diagram. However, Elbendak *et al*. (2011) argued that there are a few focusing on the formalization and implementation of the methodology for the object model creation process.

Tjoa and Berger (1994) proposed a rule-based design tool, Data Model Generator (DMG) to maintain rules and heuristics in several knowledge bases. They employed a parsing algorithm to access information from the grammar using a lexicon, by retrieving necessary information using the rules and heuristics to set up a relationship between linguistic and design knowledge. They translated natural language into EER concepts. Another rule-based system, ER generator, was developed by Gomez *et al*. (1999) to generate ER models from natural language specifications. Bajwa and Choudhary (2006) developed additional rule-based system that employs NLP techniques. Al-Safadi (2009) presented a semi-automated approach for designing databases in EER diagram notation. This work concentrated on the conversion of a natural language description into a conceptual data model, i.e., an EER model.

Harmain and Gaizauskas (2003) developd a natural language based CASE tool, CM-Builder. Their tool aims to support the analysis stage of software development in an object-oriented framework by producing UML conceptual models. There is another research work related to UML, which is UML-Generator [Bajwa *et al*., 2009]. Their tool extracts requisite information from a natural language text and then converts this into a UML class diagram. There is also a system to generate an entity-relationship model from the input text and thereafter convert the entity-relationship model to Vienna Development Method (VDM) data types [Meziane and Vadera, 2004; Meziane, 1994] by means of NLP techniques.

While many researchers have reported novel methods to build various conceptual models, such as UML, EER, and ER, from natural language descriptions, it is still a challenging issue to identify classes, objects and their particular methods and attributes, which were used to be supplemented with the human intervention. Mich (1996) and Mich and Garigliano (2002) developed an NL-based prototype system, NL-OOPS, which enables object-oriented analysis from natural language specifications.

Giganto (2008) developed an algorithm to extract use case sentences from the requirements, validate the functional specifications of each sentence and reuse the validated domain dependent use cases to supply the missing functional specifications that may contain the participating classes. Dennis *et al*. (2009) presented text analysis rules for implying an aggregation or association relationship and determining behavioral operation.

Elbendak *et al*. (2011) tried to use multiple input sources: a set of functional requirements, use case descriptions or a problem statement description in natural language. They proposed the parsed use case description (PUCD) as an intermediate representation for capturing the output of the parsing stage. Next, they extracted *part-of-speech*, which is word class in sentences, to identify objects/classes, attributes and the relationships among them.

The research studied so far in this section has provided comprehensive understanding of how NLP can be applied to support the analysis and design stages of software development. Elbendak *et al*. (2011) tabulated the comparison of historical object identification proposals in software engineering.

In early history of object identification methods, there were limitations in terms of object-oriented analysis and design. In addition to the lack of the non-OO approach, they have not produced working systems as the proof of concepts of their methods [Abbott, 1983; Booch, 1991; Booch *et al*., 2005]. Meziane (1994) and Meziane and Vadera (2004) developed workable systems with the limited user interactions.

As previously described in this section as well as the previous ones, there has been a research approach to the use of natural language for modeling or understanding design knowledge in product design [e.g., Chiu and Shu, 2007; Stone and Wood, 2000]. However, current NL-based design research has limitations to adopt as is, for the transformative product design, since the desired transformative design environment has to be flexible so as to fertilize the natural language based semantic design activities throughout the design transformation process, including the design knowledge acquisition.

2.2.4. Summary

As mentioned in early of this section, software design has great similarities with product design. They also need to understand functional requirements and then translate them to the object world to implement them into a system. Even though natural language processing techniques had a pause in development due to their challenging issues, with the advanced technologies in dictionaries and the NLP tools themselves, recently many software researchers have shed new light on the natural language processing based approach to automatically and systematically identify objects/classes.

Among many research works, this research applies the approach proposed by Elbendak *et al*. (2011), specifically the parsed use case descriptions as an intermediate representation for the design capture. They are basically an extension of Mich and Garigliano’s (2002) NL-based prototype system, NL-OOPS. Nonetheless, they are able to identify the relationships between objects in a semi-automatic manner, which can be translated into behavioral interaction between structures in product design. Another benefit to employ the natural language processing techniques is to enhance the semantic approach for the transformative product design. This aspect will be presented in the later proposed research methods of this research.

CHAPTER 3

INTERACTION NETWORK GENERATION FROM FUNCTIONAL REQUIREMENTS

3.1. Introduction

This research presents a new design paradigm that takes a different approach from the traditional innovative product design methods. Current product design approaches have shown the following procedures: (1) identification of new design challenges, (2) formulation of the challenges into specific problems, (3) generalization of specific problems, (4) detection of general solutions, and finally (5) application of general solutions for specific problems. These approaches have started with the identification process of new design challenges, seeking a way of solutions to problems found through design analogy activities. Transformative product design, however, pays attention to applications of existing design solutions on the assumption that there are many advanced technologies or design solutions that can be applied across multiple domains. This is not a trivial design activity; rather, it requires creative, inspired thinking and experience. This research aims to provide a transformative product design environment that fulfills the requirements.

To provide such transformative product design environment, design knowledge acquisition is one of significant design activities. One of goals of this research is to present research methods in a systematic manner, in order to reduce human intervention to a minimum. To pursue the goal, this chapter describes how to systematically generate a design model for any product design across multiple disciplines.

To fulfill new product design demands for cross-disciplinary technology advances, it is imperative to understand existing design knowledge for the identification of possible new applications that can reuse the knowledge, rather than to classify a new design challenge. In other words, while the traditional design representation models and tools have focused on the identification of new design challenges for the single product development, a new design paradigm for cross disciplinary technology advances requires the initial process that can analyze existing design knowledge from available resources. Thus, this research selects functional requirement documents as existing design knowledge, which are usually written in a natural language (i.e., English in this research).

3.2. Interaction Network Identification Process

A functional requirement document has challenging issues to analyze because there is no standard method or template for the requirement documents. Functional description may be written in either informal free text or more formalized description (i.e., use case description). To embrace functional descriptions in either form, the proposed design representation model has to be a general representation model to cover different kinds of description in a natural language.

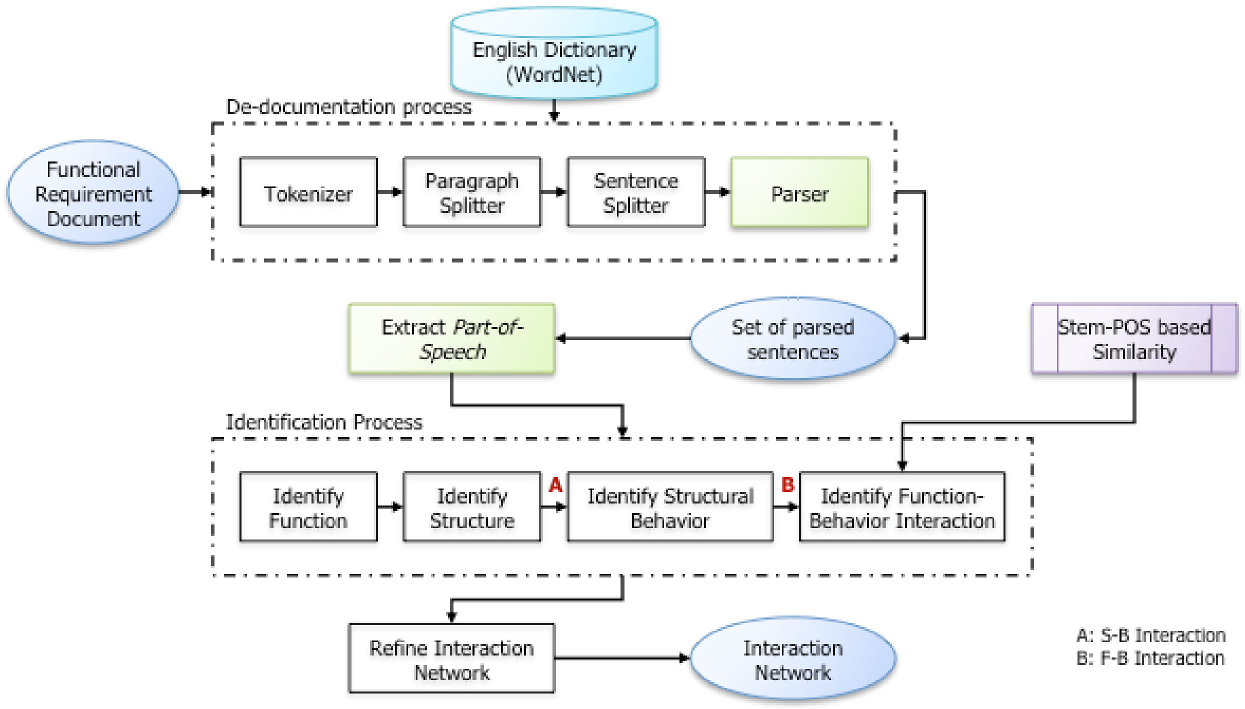


Figure 3.1 Interaction network identification process

To implement this process, this research employs natural language processing techniques for the identification of design elements from functional descriptions written in English. The design elements determined from this process include functions, structures, and behaviors. Figure 3.1 shows the interaction network identification process for the generalized design model.

First, the identification process starts with the reverse documentation process from an informal free text-based functional requirement document by tokenizing every single word. In this process, there are two splitting processes to identify paragraphs and sentences from the functional requirement document. Each sentence is decomposed by the word classifier, which can parse every single word with respect to its *part-of-speech*, the word class of the sentence. This is the initial pre-processing step of the reverse conceptual design engineering that extracts design knowledge from functional descriptions written in English, in the perspective of the FBS model studied in Chapter 2. Examples of entire identification process will be given in the next case study section of this chapter.

The second process is the identification process for the design model from the parsed document. Basically there are two identification processes: the function and structure identification and their interaction identification. For the function detection, the purposive verb phrases are considered because the function is associated with the concept of “what the object (or product) is for”; while structures are classified out of noun (phrases) since they are to represent “what the object (or product) consists of”. Gero (1990) presents the aforesaid definitions of each design component, including the following concept. The next step is to determine their interactions, which are behaviors with the definition of “what the object (or product) does”. Basically, every verb between structures is considered as behavioral interaction. The design model assumes that there is only one type of behaviors because the finalized design document means that there is no gap between the synthesized expected behaviors and structural behaviors of the FBS model. Thus, after identifying the (structural) behaviors between structures, the function-behavior interaction is determined from the association of functions to structural behaviors. The next subsection presents the definitions used for this process and the algorithm of the identification process.

3.2.1. Functional Requirement Analysis

When considering a computer-aided design tool, many research works have tried to do something with digital design drawings. However, such drawings have been reported to lack of design intentions or rationale easily [Kim *et al*., 2006; Kim *et al*., 2009]. In other words, it does not encounter any functional perspectives of the design. Hence, design drawings are not a good candidate for the reverse design engineering.

Functional requirements are appropriate to facilitate conceptual design by analyzing existing design knowledge since the description of the requirements comprehensively expresses teleological knowledge and (partial) physical knowledge. Additionally, functional requirements require less complicated description in details of design knowledge than detail designs do. At least, functional requirements are not necessary to describe geometric information of designs that may delay the design decision time and raise obstacles to acquire design knowledge. Recent research works have reported to explore the use of language for conceptual design (e.g., Stone and Wood (2000) and Chiu and Shu (2007)).

Functional requirement analysis is also a promising area in software design [Elbendak *et al*., 2011], which is similar to the product design processes, in order to improve a systematic assistance for computers' capability to understand the software design knowledge by reducing human intervention and errors if possible. To facilitate this systematic assistance, software design researchers have adopted the state-of-the-art technology, natural language processing for the last decade, as the NLP tools have been improved so that they can be applied for informal free text analysis research [Elbendak *et al*., 2011].

Similarly, research in product design has also reported to adopt the NLP technologies to expedite information retrieval in design knowledge [Ur-Rahman, 2010; Ahmed *et al*., 2007; Nanda *et al*., 2006; Yang *et al*., 2005; Dong and Agogino, 1997]. Thus, this research also participates in the research trend to employ the use of language for conceptual design.

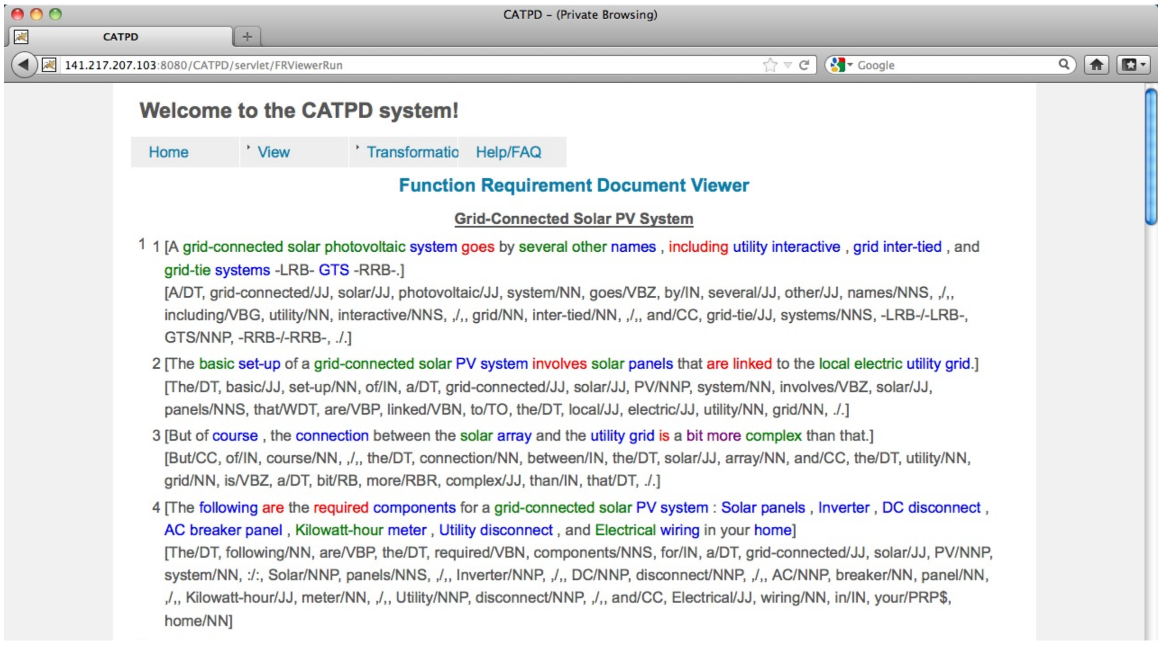
3.2.2. Identification Algorithm

For the proposed interaction network identification process, this research defines key elements of the functional requirement document as follows:

DEFINITIONS

- $FR := D$, where FR is the functional requirement and D is a text-based functional requirement document.
- $D := P = \{p_1, p_2, \ldots, p_i\}$, where p_i is a paragraph and P is a set of paragraphs in D.
- $p := S = \{s_1, s_2, \ldots, s_j\}$, where s_j is a sentence and S is a set of sentences in p.
- $s := W = \{tw_1, tw_2, \ldots, tw_k\}$, where tw_k is a pair of a word and its linguistic structure tag according to *part-of-speech* (i.e., $<w, t>$, where w is a word and t is a tag) and W is a set of pairs in s.

According to the de-documentation process in Figure 3.1, a functional requirement document is decomposed into paragraphs and their sentences. Next, each decomposed sentence is also tokenized by the part-of-speech of each word in according to the above definitions. Thus each word consists of a pair of stem word, which is the origin of the word, and its linguistic structure *part-of-speech* tag.


CATPD – (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/FRViewerRun
Google
Home
View
Transformatio
Help/FAQ

Welcome to the CATPD system!

Function Requirement Document Viewer

Grid-Connected Solar PV System

1 1 [A grid-connected solar photovoltaic system goes by several other names , including utility interactive , grid inter-tied , and grid-tie systems -LRB- GTS -RRB-.]
[A/DT, grid-connected/JJ, solar/JJ, photovoltaic/JJ, system/NN, goes/VBZ, by/IN, several/JJ, other/JJ, names/NNS, ,/,, including/VBG, utility/NN, interactive/NNS, ,/,, grid/NN, inter-tied/NN, ,/,, and/CC, grid-tie/JJ, systems/NNS, -LRB-/-LRB-, GTS/NNP, -RRB-/-RRB-, ./.]

2 [The basic set-up of a grid-connected solar PV system involves solar panels that are linked to the local electric utility grid.]
[The/DT, basic/JJ, set-up/NN, of/IN, a/DT, grid-connected/JJ, solar/JJ, PV/NNP, system/NN, involves/VBZ, solar/JJ, panels/NNS, that/WDT, are/VBP, linked/VBN, to/TO, the/DT, local/JJ, electric/JJ, utility/NN, grid/NN, ./.]

3 [But of course , the connection between the solar array and the utility grid is a bit more complex than that.]
[But/CC, of/IN, course/NN, ,/,, the/DT, connection/NN, between/IN, the/DT, solar/JJ, array/NN, and/CC, the/DT, utility/NN, grid/NN, is/VBZ, a/DT, bit/RB, more/RBR, complex/JJ, than/IN, that/DT, ./.]

4 [The following are the required components for a grid-connected solar PV system : Solar panels , Inverter , DC disconnect , AC breaker panel , Kilowatt-hour meter , Utility disconnect , and Electrical wiring in your home]
[The/DT, following/NN, are/VBP, the/DT, required/VBN, components/NNS, for/IN, a/DT, grid-connected/JJ, solar/JJ, PV/NNP, system/NN, :/:, Solar/NNP, panels/NNS, ,/,, Inverter/NNP, ,/,, DC/NNP, disconnect/NNP, ,/,, AC/NNP, breaker/NN, panel/NN, ,/,, Kilowatt-hour/JJ, meter/NN, ,/,, Utility/NNP, disconnect/NNP, ,/,, and/CC, Electrical/JJ, wiring/NN, in/IN, your/PRP$, home/NN]

Figure 3.2 An example of decomposed and tokenized functional requirement document

The decomposed and tokenized data will be accumulated in the functional requirement base. Figure 3.2 shows an example of a grid-connected solar PV system after the de-documentation process. The completely decomposed document is attached in Appendix A.

After the de-documentation process, the identification process can start for either functions or structures. Practically, that does not matter at all which one becomes the first step for the identification process in terms of the interaction network generation. This chapter assumes that designers start from the function identification.

INTERACTION NETWORK IDENTIFICATION ALGORITHM

Step 1. Detect functional sentence candidates from the functional requirement document.

Step 2. Determine functions from the functional sentence candidates.

Step 3. Detect non-duplicated nouns from the functional requirement document.

Step 4. Obtain the cardinality of each noun to determine its priority for the structure candidate.

Step 5. Detect structural sentence candidates, which include the structure candidates.

Step 6. Determine structures from the structural sentence candidates.

Step 7. Detect behavioral verbs between structures.

Step 8. Determine behaviors.

Step 9. Identify the interaction between functions and structural behaviors.

The above algorithm shows the identification process. The first step of the identification algorithm begins from purposive verb phrases in the functional requirement document to understand the concept of “what the product is for”. To determine the purposive verb phrases systematically, this research employed the two forms for the purposive phrases: To-Verb Base and For-Verb Present Participle. Based on these purposive verb phrases in the functional requirement document, the functional sentence candidates can be determined as follows:

$S^{fc} = \{s^{v}_{ij} \mid s^{v}_{ij} = 1\}$ Definition (3.1)

, where s^{v}_{ij} determines if a sentence j in a paragraph i includes a purposive verb phrase or not. Hence, s^{v}_{ij} is 1 only if it includes a purposive verb phrase; otherwise it is zero. At this point, designers are able to select appropriate functions by naming each function with the functional sentence candidates.

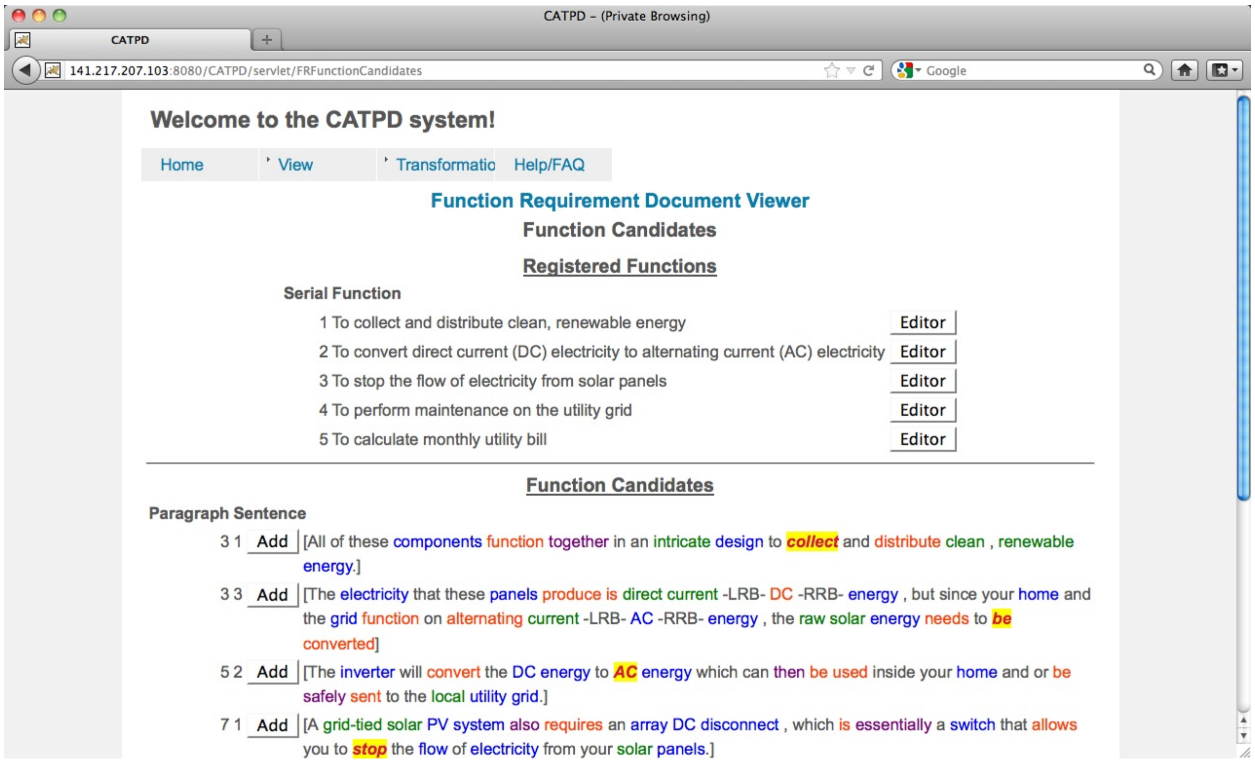
Functions determined by designers are defined as:

$\Lambda := \{\lambda_1, \lambda_2, \ldots, \lambda_l\}$ Definition (3.2)

, where λ_l is the schema of a function in *FR*. Function Schema is defined as:

λ_l = <ID, Function, S^{λ}> Definition (3.3)

, where $S^{\lambda} \subset S^{fc}$. ID is the key identifier of each function in *FR*. Function is the function name represented in a linguistic phrase (or sentence). Figure 3.3 shows an example of the function identification from the functional requirement document of a grid-connected solar photovoltaic (PV) system.


CATPD – (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/FRFunctionCandidates
Google
Home
View
Transformatio
Help/FAQ

Welcome to the CATPD system!

Function Requirement Document Viewer

Function Candidates

Registered Functions

Serial Function

1 To collect and distribute clean, renewable energy Editor
2 To convert direct current (DC) electricity to alternating current (AC) electricity Editor
3 To stop the flow of electricity from solar panels Editor
4 To perform maintenance on the utility grid Editor
5 To calculate monthly utility bill Editor

Function Candidates

Paragraph Sentence

3 1 Add [All of these components function together in an intricate design to collect and distribute clean , renewable energy.]

3 3 Add [The electricity that these panels produce is direct current -LRB- DC -RRB- energy , but since your home and the grid function on alternating current -LRB- AC -RRB- energy , the raw solar energy needs to be converted]

5 2 Add [The inverter will convert the DC energy to AC energy which can then be used inside your home and or be safely sent to the local utility grid.]

7 1 Add [A grid-tied solar PV system also requires an array DC disconnect , which is essentially a switch that allows you to stop the flow of electricity from your solar panels.]

Figure 3.3 An example of function identification

Next, the identification algorithm extracts nouns for structural candidates from the functional requirement document. The candidates in nouns or noun phrases are represented in the following definition:

$N = \{n_1, n_2, \ldots, n_p\}$ Definition (3.4)

, where n_p is a non-duplicated noun from D.

The cardinality of each candidate is defined as

$\text{Rank}N_p := |n_p|$ Definition (3.5)

, where |•| is the cardinality of •. The greater $\text{Rank}N_p$, the higher possibility of structure for n_p is. Given the structural candidates, the structural sentence candidates can be determined as follows:

$S^{sc} = \{s^n{}_{ij} \mid s^n{}_{ij} = 1\}$ Definition (3.6)

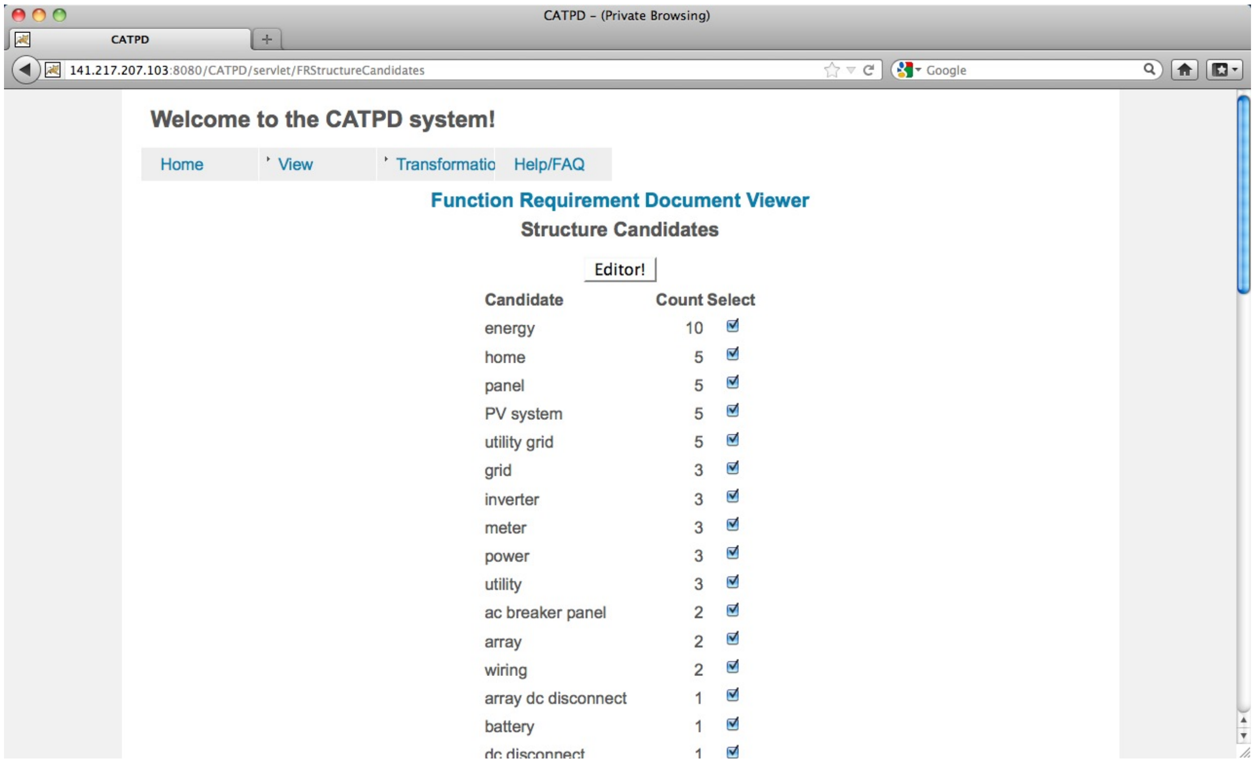
, where $s^n{}_{ij}$ determines if a sentence j in a paragraph i includes n_p or not. Hence, $s^n{}_{ij}$ is 1 only if it includes n_p; otherwise it is zero. Like the function selection, designers are able to select appropriate structures, which can be defined as

$\varDelta := \{\delta_1, \delta_2, \ldots, \delta_m\}$ Definition (3.7)

, where δ_m is the schema of a structure in FR. Structure Schema is

$\delta_m = \langle \text{ID, Structure}, S^{sc} \rangle$ Definition (3.8)

, where ID is the key identifier of each structure in FR and Structure is a noun n_p selected by designers. Figure 3.4 shows an example of the structure identification from the functional requirement document of a grid-connected solar PV system.


CATPD – (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/FRStructureCandidates
Google
Home
View
Transformatio
Help/FAQ
Editor!

Welcome to the CATPD system!

Function Requirement Document Viewer

Structure Candidates

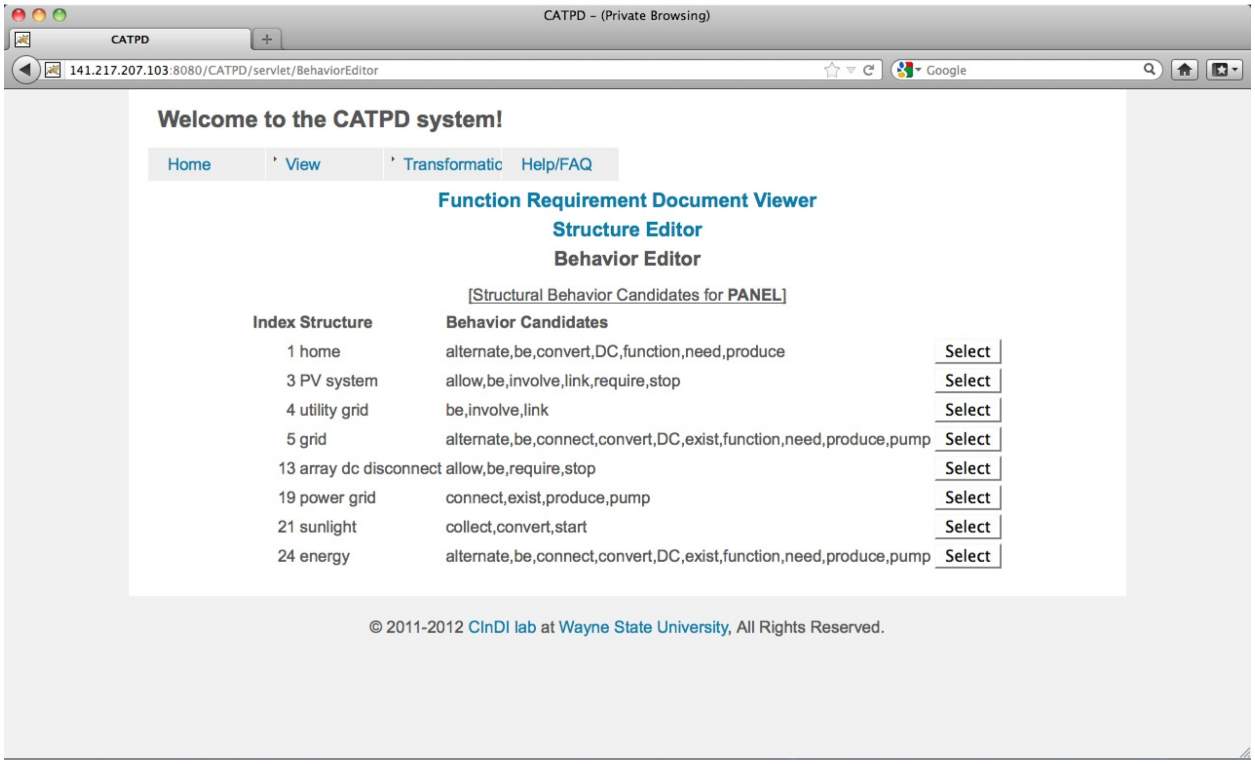
Candidate	Count	Select
energy	10	✓
home	5	✓
panel	5	✓
PV system	5	✓
utility grid	5	✓
grid	3	✓
inverter	3	✓
meter	3	✓
power	3	✓
utility	3	✓
ac breaker panel	2	✓
array	2	✓
wiring	2	✓
array dc disconnect	1	✓
battery	1	✓
dc disconnect	1	✓

Figure 3.4 An example of structure identification

The identification process continues to determine interactions between structures, which are (structural) behaviors. The behaviors can be extracted from the structural sentence candidates. Behaviors can be considered as operations, relationships, or inheritance in software design. Elbendak et al. (2011) classified them into four types (i.e., association, aggregation, composition or inheritance) by looking for clauses in the form “something contains something”, “something is part of something” and “something is made up of something”. They are likely related to verbal expression between objects (i.e., noun phrases). This research follows a similar approach to identify behavioral interaction between structures. The behavioral verbs between structures can be defined as

$$\varsigma^{xy} = \delta_x \bullet \delta_y = \{\mathrm{VB} \mid \forall s \in S^{sc}{}_x \cap S^{sc}{}_y\} \qquad \text{Definition (3.9)}$$

, where δ_x and δ_y are structure schemata, $1 \leq x, y \leq m$, $x \neq y$, and VB = $\{w \mid t$ is a verbal tag and $<w, t> \in s\}$.


CATPD – (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/BehaviorEditor
Google
Home
View
Transformatic
Help/FAQ
Function Requirement Document Viewer
Structure Editor
Behavior Editor

Welcome to the CATPD system!

[Structural Behavior Candidates for **PANEL**]

Index Structure	Behavior Candidates	
1 home	alternate,be,convert,DC,function,need,produce	Select
3 PV system	allow,be,involve,link,require,stop	Select
4 utility grid	be,involve,link	Select
5 grid	alternate,be,connect,convert,DC,exist,function,need,produce,pump	Select
13 array dc disconnect	allow,be,require,stop	Select
19 power grid	connect,exist,produce,pump	Select
21 sunlight	collect,convert,start	Select
24 energy	alternate,be,connect,convert,DC,exist,function,need,produce,pump	Select

© 2011-2012 CInDI lab at Wayne State University, All Rights Reserved.

Figure 3.5 An example of (structural) behavior identification

Finally, according to the given behavioral verbs, behaviors can be determined as

$\Sigma := \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ Definition (3.10)

, where $n \leq m \times m$ and σ_n is the schema of a behavior in *FR*. Behavior Schema is

$\sigma_n = <\text{ID}, \text{Behavior}, \delta_x, \delta_y, \varsigma^{xy}>$ Definition (3.11)

, where ID is the key identifier of each behavior in *FR* and Behavior is the behavior name for structures δ_x and δ_y. The last step of the identification process is to detect the function-behavior interaction. This step connects the functions and the structures through the structural behavior.

3.2.3. Interaction Network Generation and Visualization

According to the de-documentation process with the NLP technologies and the identification algorithm, a document of functional requirements is translated into the defined three schemata of functions, structures, and behaviors. Given the schemata, the translated design knowledge is acquired and stored in the interaction network repository. Detail explanation of different repositories, including the interaction network repository, will be described in the system architecture section in Chapter 7. As defined in the schemata, one schema is associated with another schema according to their components. Based on this relationship between schemata, this research develops a system module that converts the relationship into a graphic network file, i.e., Graph Exchange XML Format (GEXF). The following figure shows an example of the automatically converted network file of a grid-connected solar PV system.

<?xml version='1.0' encoding='UTF-8'?><gexf xmlns="http://www.gexf.net/1.2draft" xmlns:viz="http://www.gexf.net/1.2draft/viz" version="1.2"><meta lastmodifieddate="2012-07-12"><creator>CInDI.pro</creator><description>SBI Network: Grid-Connected Solar PV System</description></meta><graph defaultedgetype="undirected" idtype="string" mode="static"><nodes count="24"><node id="0" label="home"><viz:color r="255" g="69" b="0" /></node><node id="1" label="panel"><viz:color r="255" g="69" b="0" /></node><node id="2" label="PV system"><viz:color r="255" g="69" b="0" /></node><node id="3" label="utility grid"><viz:color r="255" g="69" b="0" /></node><node id="4" label="grid"><viz:color r="255" g="69" b="0" /></node><node id="5" label="inverter"><viz:color r="255" g="69" b="0" /></node><node id="6" label="meter"><viz:color r="255" g="69" b="0" /></node><node id="7" label="power"><viz:color r="255" g="69" b="0" /></node><node id="8" label="utility"><viz:color r="255" g="69" b="0" /></node><node id="9" label="ac breaker panel"><viz:color r="255" g="69" b="0" /></node><node id="10" label="array"><viz:color r="255" g="69" b="0" /></node><node id="11" label="wiring"><viz:color r="255" g="69" b="0" /></node><node id="12" label="array dc disconnect"><viz:color r="255" g="69" b="0" /></node><node id="13" label="battery"><viz:color r="255" g="69" b="0" /></node><node id="14" label="dc disconnect"><viz:color r="255" g="69" b="0" /></node><node id="15" label="panel system"><viz:color r="255" g="69" b="0" /></node><node id="16" label="PCU"><viz:color r="255" g="69" b="0" /></node><node id="17" label="power conditioning unit"><viz:color r="255" g="69" b="0" /></node><node id="18" label="power grid"><viz:color r="255" g="69" b="0" /></node><node id="19" label="Solar

panel"><viz:color r="255" g="69" b="0" /></node><node id="20" label="sunlight"><viz:color r="255" g="69" b="0" /></node><node id="21" label="utility bill"><viz:color r="255" g="69" b="0" /></node><node id="22" label="utility disconnect"><viz:color r="255" g="69" b="0" /></node><node id="23" label="energy"><viz:color r="255" g="69" b="0" /></node></nodes><edges count="152"><edge id="0" source="0" target="1" type="undirected" /><edge id="1" source="0" target="2" type="undirected" /><edge id="2" source="0" target="3" type="undirected" /><edge id="3" source="0" target="4" type="undirected" /><edge id="4" source="0" target="5" type="undirected" /><edge id="5" source="0" target="6" type="undirected" /><edge id="6" source="0" target="7" type="undirected" /><edge id="7" source="0" target="9" type="undirected" /><edge id="8" source="0" target="11" type="undirected" /><edge id="9" source="0" target="14" type="undirected" /><edge id="10" source="0" target="19" type="undirected" /><edge id="11" source="0" target="22" type="undirected" /><edge id="12" source="0" target="23" type="undirected" /><edge id="13" source="1" target="0" type="undirected" /><edge id="14" source="1" target="2" type="undirected" /><edge id="15" source="1" target="3" type="undirected" /><edge id="16" source="1" target="4" type="undirected" /><edge id="17" source="1" target="12" type="undirected" /><edge id="18" source="1" target="18" type="undirected" /><edge id="19" source="1" target="20" type="undirected" /><edge id="20" source="1" target="23" type="undirected" /><edge id="21" source="2" target="0" type="undirected" /><edge id="22" source="2" target="1" type="undirected" /><edge id="23" source="2" target="3" type="undirected" /><edge id="24" source="2" target="5" type="undirected" /><edge id="25" source="2" target="6" type="undirected" /><edge id="26" source="2" target="9" type="undirected" /><edge id="27" source="2" target="11" type="undirected" /><edge id="28" source="2" target="12" type="undirected" /><edge id="29" source="2" target="14" type="undirected" /><edge id="30" source="2" target="16" type="undirected" /><edge id="31" source="2" target="17" type="undirected" /><edge id="32" source="2" target="19" type="undirected" /><edge id="33" source="2" target="22" type="undirected" /><edge id="34" source="3" target="0" type="undirected" /><edge id="35" source="3" target="1" type="undirected" /><edge id="36" source="3" target="2" type="undirected" /><edge id="37" source="3" target="5" type="undirected" /><edge id="38" source="3" target="7" type="undirected" /><edge id="39" source="3" target="8" type="undirected" /><edge id="40" source="3" target="10" type="undirected" /><edge id="41" source="3" target="23" type="undirected" /><edge id="42" source="4" target="0" type="undirected" /><edge id="43" source="4" target="1" type="undirected" /><edge id="44" source="4" target="7" type="undirected" /><edge id="45" source="4" target="8" type="undirected" /><edge id="46" source="4" target="18" type="undirected" /><edge id="47" source="4" target="23" type="undirected" /><edge id="48" source="5" target="0" type="undirected" /><edge id="49" source="5" target="2" type="undirected" /><edge id="50" source="5" target="3" type="undirected" /><edge id="51" source="5" target="6" type="undirected" /><edge id="52" source="5" target="9" type="undirected" /><edge id="53" source="5" target="11" type="undirected" /><edge id="54" source="5" target="14" type="undirected" /><edge id="55" source="5" target="16" type="undirected" /><edge id="56" source="5" target="17" type="undirected" /><edge id="57" source="5" target="19" type="undirected" /><edge id="58" source="5" target="22" type="undirected" /><edge id="59" source="5" target="23" type="undirected" /><edge id="60" source="6" target="0" type="undirected" /><edge id="61" source="6" target="2" type="undirected" /><edge id="62" source="6" target="5"

type="undirected" /><edge id="63" source="6" target="7" type="undirected" /><edge id="64" source="6" target="9" type="undirected" /><edge id="65" source="6" target="10" type="undirected" /><edge id="66" source="6" target="11" type="undirected" /><edge id="67" source="6" target="14" type="undirected" /><edge id="68" source="6" target="19" type="undirected" /><edge id="69" source="6" target="21" type="undirected" /><edge id="70" source="6" target="22" type="undirected" /><edge id="71" source="7" target="0" type="undirected" /><edge id="72" source="7" target="3" type="undirected" /><edge id="73" source="7" target="4" type="undirected" /><edge id="74" source="7" target="6" type="undirected" /><edge id="75" source="7" target="8" type="undirected" /><edge id="76" source="7" target="10" type="undirected" /><edge id="77" source="7" target="21" type="undirected" /><edge id="78" source="7" target="23" type="undirected" /><edge id="79" source="8" target="3" type="undirected" /><edge id="80" source="8" target="4" type="undirected" /><edge id="81" source="8" target="7" type="undirected" /><edge id="82" source="8" target="23" type="undirected" /><edge id="83" source="9" target="0" type="undirected" /><edge id="84" source="9" target="2" type="undirected" /><edge id="85" source="9" target="5" type="undirected" /><edge id="86" source="9" target="6" type="undirected" /><edge id="87" source="9" target="11" type="undirected" /><edge id="88" source="9" target="14" type="undirected" /><edge id="89" source="9" target="19" type="undirected" /><edge id="90" source="9" target="22" type="undirected" /><edge id="91" source="10" target="3" type="undirected" /><edge id="92" source="10" target="6" type="undirected" /><edge id="93" source="10" target="7" type="undirected" /><edge id="94" source="10" target="21" type="undirected" /><edge id="95" source="11" target="0" type="undirected" /><edge id="96" source="11" target="2" type="undirected" /><edge id="97" source="11" target="5" type="undirected" /><edge id="98" source="11" target="6" type="undirected" /><edge id="99" source="11" target="9" type="undirected" /><edge id="100" source="11" target="14" type="undirected" /><edge id="101" source="11" target="19" type="undirected" /><edge id="102" source="11" target="22" type="undirected" /><edge id="103" source="12" target="1" type="undirected" /><edge id="104" source="12" target="2" type="undirected" /><edge id="105" source="13" target="23" type="undirected" /><edge id="106" source="14" target="0" type="undirected" /><edge id="107" source="14" target="2" type="undirected" /><edge id="108" source="14" target="5" type="undirected" /><edge id="109" source="14" target="6" type="undirected" /><edge id="110" source="14" target="9" type="undirected" /><edge id="111" source="14" target="11" type="undirected" /><edge id="112" source="14" target="19" type="undirected" /><edge id="113" source="14" target="22" type="undirected" /><edge id="114" source="16" target="2" type="undirected" /><edge id="115" source="16" target="5" type="undirected" /><edge id="116" source="16" target="17" type="undirected" /><edge id="117" source="17" target="2" type="undirected" /><edge id="118" source="17" target="5" type="undirected" /><edge id="119" source="17" target="16" type="undirected" /><edge id="120" source="18" target="1" type="undirected" /><edge id="121" source="18" target="4" type="undirected" /><edge id="122" source="18" target="23" type="undirected" /><edge id="123" source="19" target="0" type="undirected" /><edge id="124" source="19" target="2" type="undirected" /><edge id="125" source="19" target="5" type="undirected" /><edge id="126" source="19" target="6" type="undirected" /><edge id="127" source="19" target="9" type="undirected" /><edge id="128" source="19" target="11" type="undirected" /><edge id="129" source="19" target="14" type="undirected" /><edge id="130" source="19" target="22" type="undirected" /><edge id="131" source="20" target="1"

type="undirected" /><edge id="132" source="21" target="6" type="undirected" /><edge id="133" source="21" target="7" type="undirected" /><edge id="134" source="21" target="10" type="undirected" /><edge id="135" source="22" target="0" type="undirected" /><edge id="136" source="22" target="2" type="undirected" /><edge id="137" source="22" target="5" type="undirected" /><edge id="138" source="22" target="6" type="undirected" /><edge id="139" source="22" target="9" type="undirected" /><edge id="140" source="22" target="11" type="undirected" /><edge id="141" source="22" target="14" type="undirected" /><edge id="142" source="22" target="19" type="undirected" /><edge id="143" source="23" target="0" type="undirected" /><edge id="144" source="23" target="1" type="undirected" /><edge id="145" source="23" target="3" type="undirected" /><edge id="146" source="23" target="4" type="undirected" /><edge id="147" source="23" target="5" type="undirected" /><edge id="148" source="23" target="7" type="undirected" /><edge id="149" source="23" target="8" type="undirected" /><edge id="150" source="23" target="13" type="undirected" /><edge id="151" source="23" target="18" type="undirected" /></edges></graph></gexf>

Figure 3.6 An example of GEXF file of Figure 3.7

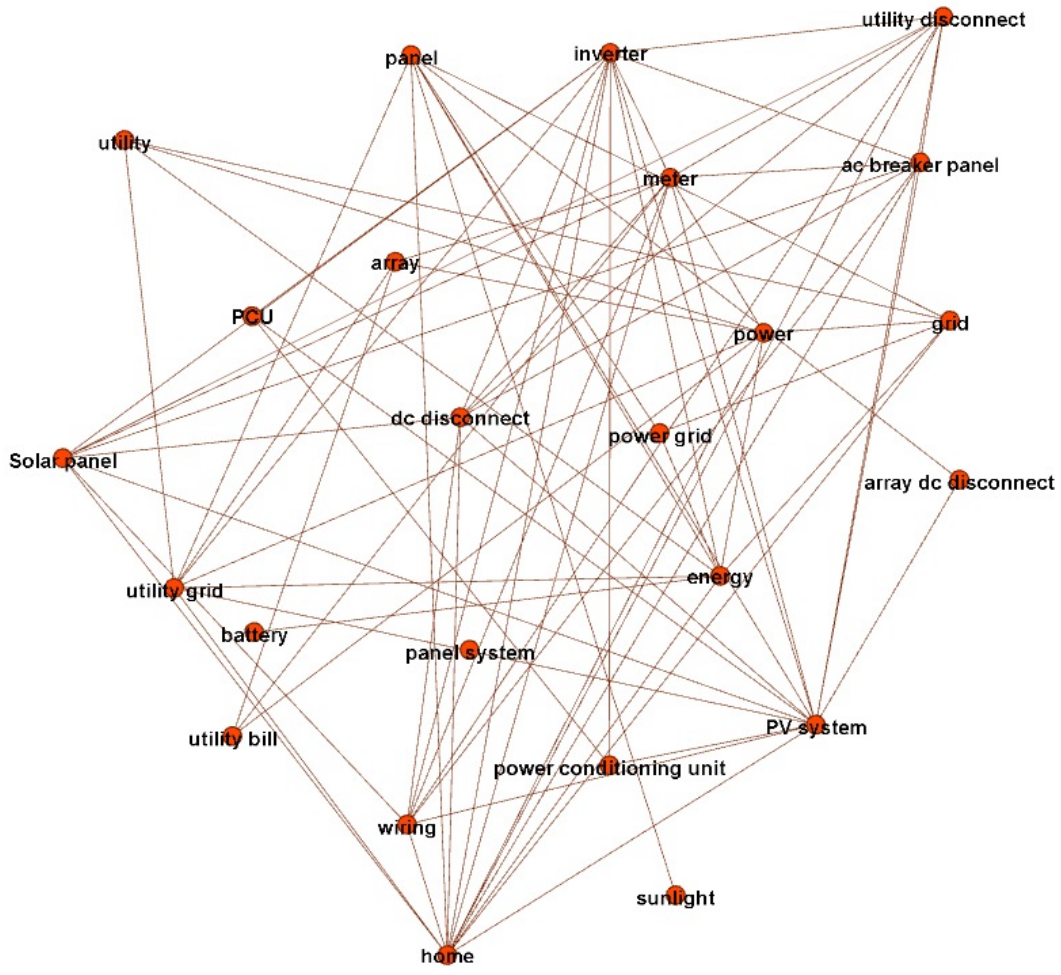


Figure 3.7 An example of Structure-Behavior Interaction (SBI) network

The exported interaction network in a GEXF file can be rendered in a network visualization tool, Gephi. By using the visualization tool, designers are able to see the behavioral interactions between structures as shown in Figure 3.7. As visualized in the figure, each node denotes a structure of a grid-connected solar PV system, while each edge between nodes represent the behavioral interaction. In the visualization, structures are colored in orange, while functions in blue. Figure 3.7 draws only the structure-behavior interactions.

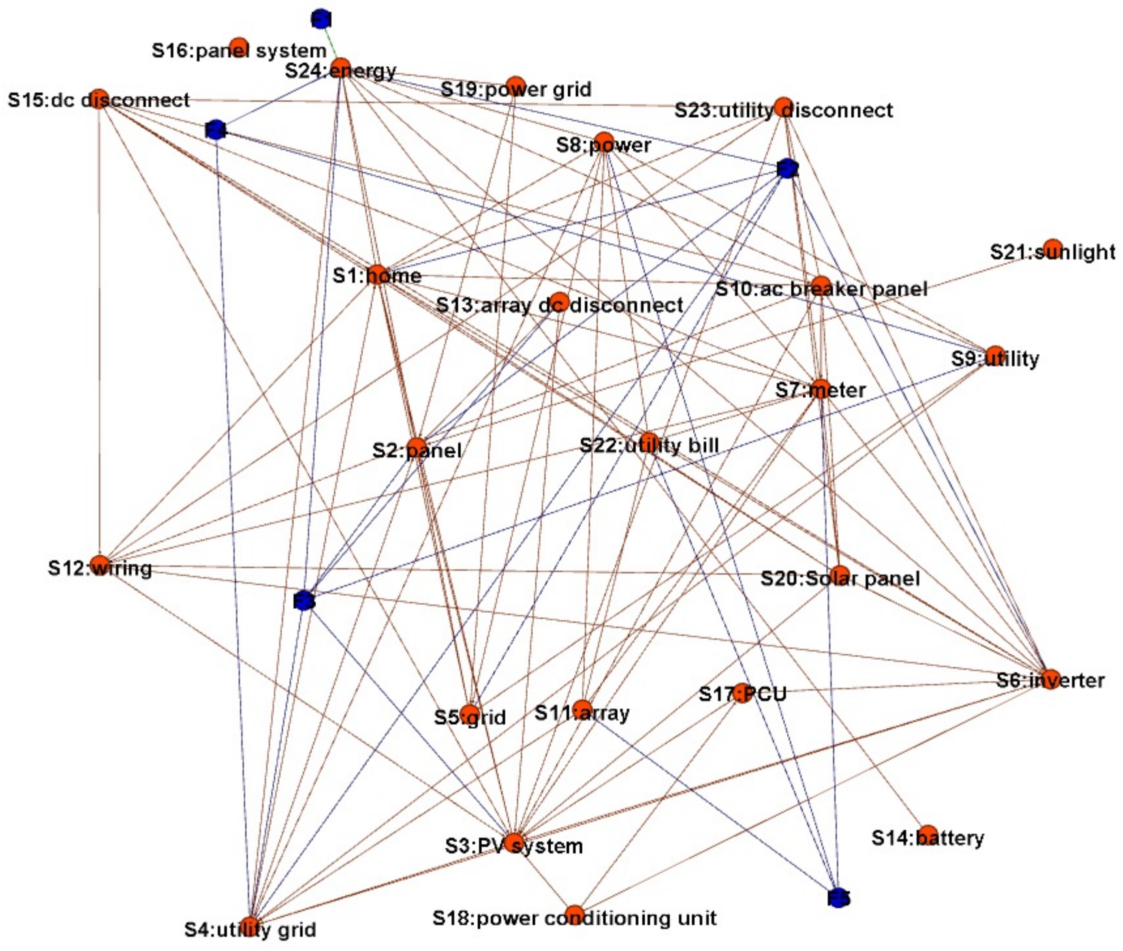


Figure 3.8 An example of interaction network of a grid-connected solar PV system

Figure 3.8 completely illustrates the entire interaction network of a grid-connected solar PV system by including functions and their associated interactions with structures. The GEXF file of Figure 3.8 is disclosed in Appendix G.

3.3. Conclusion

3.3.1. Summary

This chapter proposes a design representation model, i.e., interaction network, by adopting the FBS model in design engineering in order to fulfill new product design demands for cross-disciplinary technology advances. The difference between the FBS model and the proposed interaction identification method mainly lies in the approach to understand the design process. The FBS model focuses on the internal design process between functions and structures via behavioral relations in order to finalize the product design documentation, which is design knowledge creation. On the contrary, the proposed method pays attention to the reuse of existing design knowledge. Furthermore, to efficiently recapture design knowledge into the structured interaction network model, natural language processing techniques are adopted.

In software design, many research works contributed to the object identification, which translate functional descriptions in a natural language such as English. A functional requirement document has challenging issues to analyze because there is no standard method and template for the functional description. Therefore, this chapter presents an identification algorithm to obtain the teleological knowledge and objective artifacts as well as their interactions from an informal free text based functional description. The proposed identification method is to reduce efficiently human intervention in the design model generation, which can be eventually applied to the transformation process in the following chapters. For the transformation process, Chapter 4 proposes a method to detect functional concept from a product design. The final transformative product design alternative generation will be covered in Chapter 5.

3.3.2. Contribution

To realize the transformative product design environment, the very first step is to utilize existing design knowledge since the design transformation process aims to search new applications of existing advanced technologies or design solutions. In a sense, the de-documentation process in this chapter expedites a systematic process to extract design knowledge from a natural language based functional requirement description with minimalized human intervention and efforts. Additionally, the proposed algorithm pursues the object identification methods in software design in order to represent the extracted design knowledge into machine-interpretable network based design model. This approach allows the following research methods to manipulate transformative design alternatives, by extracting and integrating design knowledge from the network models.

CHAPTER 4

FUNCTIONAL CONCEPT DETECTION FOR TRANSFORMATION

4.1. Introduction

In the previous chapter, we have seen the utilization of existing design knowledge as the foundation for the transformative product design paradigm by employing the de-documentation process with the natural language processing technologies and the interaction network identification method. By utilizing the interaction network model, seeking a new direction to apply existing advanced technologies or design solutions is possible.

Before generating transformative design alternatives that could be applications of existing design solutions, there is a prerequisite step to identify functional concept for the design transformation process. For this step, to present designers the maximum freedom of design knowledge retrieval, it would be beneficial when taking a semantic approach. This chapter provides how to semantically retrieve functions for the functional concept that will be used for the later transformation process.

In addition to such a semantic strategy to retrieve functional concept, this research aims to present also a structural design knowledge retrieval feature. Even if designers are able to semantically detect design knowledge for the functional concept, they could miss some important design knowledge that cannot be determined by such a semantic approach. To prevent this situation, another algorithm that practices the interaction network is proposed. Thus, this chapter consists of the two complementary concept detection methods: (1) primitive concept detection and (2) extended concept detection.

4.2. Concept Detection Process

The transformation process in TPD is to fuse existing design knowledge across multiple products by identifying necessary technology advances from them. In the previous chapter, the identification method investigates functions out of a functional requirement document, which is written in informal free texts in a natural language. However, it is not a trivial task for designers to find appropriate functions for any concept.

Furthermore, if designers select any function from the design knowledge, he/she would not be sure whether all the relevant functions for the intended concept are determined. Even though the purpose of the interaction network is to assist this challenging problem, it would take too much time to concentrate to understand and utilize the interaction network.

Last but not least, if someone looks up function from the interaction network and does not provide the same functional phrase stored in the database, it is impossible to detect any relevant functions from the interaction network. This chapter, therefore, proposes a method to detect a functional concept out of the interaction network with the given functional search keyword (or phrase). The following figure shows the process of the proposed concept detection method.

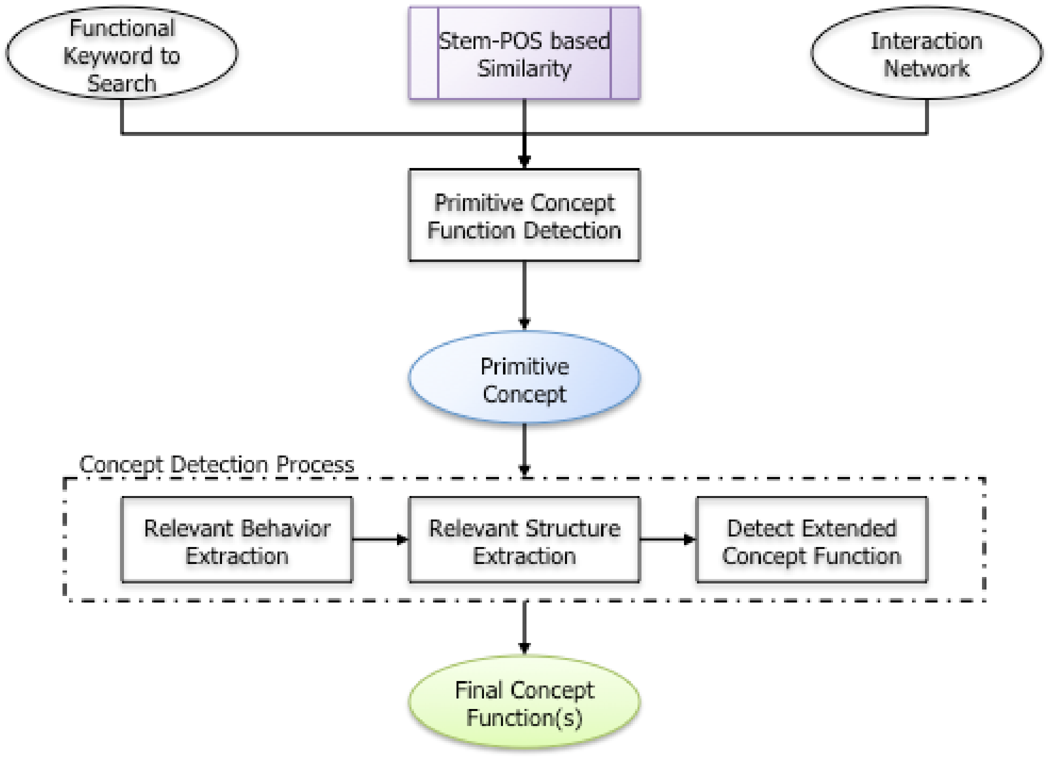

Functional Keyword to Search
Stem-POS based Similarity
Interaction Network
Primitive Concept Function Detection
Primitive Concept
Concept Detection Process
Relevant Behavior Extraction
Relevant Structure Extraction
Detect Extended Concept Function
Final Concept Function(s)

Figure 4.1 Concept detection process

As shown in Figure 4.1, there are two major processes in this method: (1) primitive concept detection and (2) extended concept detection. The primitive concept detection starts from the given functional keyword (or phrase) to search functions from an interaction network. As aforementioned, to enhance the search performance for the primitive concept, this research develops a similarity score to compare two phrases in a natural language (or sentences), called Stem-POS based Similarity (SPS) Score. This score will be also applied to the validation process of this final product design. The details of the score are described in Section 4.3.

Next, the extended concept detection is to investigate functions associated with the primitive concept functions. This technique utilizes the interaction network to identify relationships between functions and structures. Section 4.4 presents the proposed technique in detail.

4.3. Primitive Concept Detection

The primitive concept detection enriches a semantic approach to identify concept functions. For the last few years, advances in dictionaries and the natural language processing tools have reported a remarkable progress. Dictionaries include semantic relationships between words, such as synsets (i.e., sets of synonyms), hypernyms (i.e., generalization), hyponyms (i.e., specialization), meronyms (part-whole), and holonyms (whole-part). Many researchers have developed various similarity scores to compare two words, which are eventually adopted in the NLP tools, for example, the Leacock-Miller-Chodorow similarity and the Wu-Palmer similarity.

To realize the semantic capability of the proposed design supporting system, this research utilizes WordNet and the relevant application programming interfaces (APIs). However, Passos and Wainer (2009) addressed that WordNet-based similarity measures do not seem to help clustering of similar concept. They showed that a measure that does not depend on the topology of WordNet or hybrid measures with semantic relationships in WordNet results in higher performance. Thus, this research aims to develop another similarity score to embrace the simple measure and the hybrid measures based on semantic relationships in WordNet, which produces reasonable similarity scores for the proposed system. The next subsection shows the proposed similarity score.

4.3.1. Stem-POS based Similarity (SPS) Score

The proposed similarity score intends to utilize the origin (i.e. stem word) of each word in the given keywords (or phrases/sentences) for the word match and the *part-of-speech* (i.e. word class) of each word for the structure match. For the word match, Doran et al. (2004) applied classification weights related to lexical cohesive relationship between words. They set the weight of 1.0 for the exact match, 0.9 for the synonym relationship, and 0.7 for the other semantic relationship.

When applying stem words to compare via the WordNet dictionary, it is possible to assume that two words match exactly if they have the identical sets of synonyms and other lexical relationships. Thus, the proposed similarity score does not have the weight for the exact match. In addition to that, in WordNet, Passos and Wainer (2009) showed that the shared words in the specialization and generalization relationships of two words resulted in a highly reliable performance too. Unlike Doran et al. (2004) who tried to develop the weights, the proposed similarity score separate the specialization and generalization relationships from the part-whole and whole-part relationships. For the structure match, there will be a penalty weight when the *part-of-speeches* (POSes) of two words are different, for example, verb-noun.

The following definition defines the Stem-*Part-of-Speech* based Similarity (SPS) Score. The score formulates to calculate weighted semantic similarity of a pair of words in two phrases. The value of the score falls between 0 and 1; 1 represents they are the same phrases, while 0 means there is no similarity between the two.

DEFINITION: THE STEM-POS BASED SIMILARITY (SPS) SCORE

$SPS\ Score(p_x,p_y) = 1/2 \bullet \{w \bullet (m / |p_x|) + w \bullet (m / |p_y|)\}$, where

- p_x, p_y: phrases (or sentences) to compare for primitive concept detection
- $|p|$: the number of words in the phrase p
- m is the number of common words (or similar words by semantic match)
- $w = (\Sigma_{i=1\ldots m}\ sps^i_{x,y}) / m$, where semantic similarity score $sps_{x,y} = [0, 1]$
- $sps_{x,y} = \max\{sim\_syn_{x,y}, sim\_hyp_{x,y}, sim\_hypo_{x,y}, sim\_hol_{x,y}, sim\_mer_{x,y}\}$, where

  sim_semantictype is the Stem-POS based similarity score with respect to each semantic type and is defined as:

  $sim\_semantictype = pos_{x,y} \bullet \max\{|sim_{x,y}| / |stem_x|, |sim_{x,y}| / |stem_y|\} \bullet w^{sem}$, where $|sim|$ is the cardinality of the common semantic words between two stem words, $stem_x$ and $stem_y$, $|stem|$ is the cardinality of each stem word, $pos_{x,y}$ is the weight for the structure match and it is set to 1 when the *part-of-speeches* of the two stem words are identical and otherwise 0.5, w^{sem} is the weight with respect to the lexical cohesive relationship.

The pseudo code in Figure 4.2 is to implement the similarity score in the definition.

```
double function getSimilarityOfSemanticWords (Word1, Word2, Type of Semantic) {
set stems1 to stem words of the first word to compare
set stems2 to stem words of the second word to compare

for stem1 = the first stem word of stems1 to the last stem word of stems1 {
    set synset1 to the synset of stem1 in WordNet
    set semantic_words1 to a set of semantic words of synset1 in WordNet
    set card1 to the accumulated number of sets of semantic_words1 for the first word

    for stem2 = the first stem word of stems2 to the last stem word of stems2 {
        set synset2 to the synset of stem2 in WordNet
```

```
            set semantic_words2 to a set of semantic words of synset2 in WordNet
            set card2 to the accumulated number of sets of semantic_words2 for the first word

            for semantic_word1 := the first semantic word of semantic_word1
                  to the last semantic word of semantic_word1 {

                  for semantic_word2 := the first semantic word of semantic_word2
                        to the last semantic word of semantic_word2 {

                        if synonym1 is equal to synonym2
                              set common_card to common_card + 1
                  }
            }
      }
}

set sim1 to (common_card / card1)
set sim2 to (common_card / card2)

set similarity to Max{sim1, sim2}

if part-of-speech1 is not equal to part-of-speech2
      set similarity to (similarity * 0.5)

if type of semantic is hypernym or hyponym
      set similarity to (similarity * 0.9)

if type of semantic is holonym or meronym
      set similarity to (similarity * 0.7)

return similarity
}

double function getSimilarityOfWords (Word1, Word2) {
      set sim_syn to getSimilarityOfSemanticWords(Word1, Word2, Synonym)
      set sim_hyp to getSimilarityOfSemanticWords(Word1, Word2, Hypernym)
      set sim_hypo to getSimilarityOfSemanticWords(Word1, Word2, Hyponym)
      set sim_hol to getSimilarityOfSemanticWords(Word1, Word2, Holonym)
      set sim_mer to getSimilarityOfSemanticWords(Word1, Word2, Meronym)

      set similarity to Max{sim_syn, sim_hyp, sim_hypo, sim_hol, sim_mer}

      return similarity
}
```

Figure 4.2 Pseudo code of Stem-POS based Similarity Score for the comparison of words

Given the functional keywords (phrases or sentences) to search a primitive concept, the primitive concept detection process continues. Every function of a product design is calculated according to the proposed SPS score. The greater the SPS score, the higher possibility of primitive concept is. Examples of the primitive concept detection are given in the following section.

4.3.2. Example of Primitive Concept Detection

Tables in this section are examples of that the results of the primitive concept detection for an example of streetlight product with respect to different functional keywords (phrases): (1) To transfer power source to light, (2) To transmit power to light, and (3) To transmit power from grid to light. The first functional phrase shows the highest SPS score since the phrase use the identical words used in the function description of the example product, Streetlight (Table 4.1).

Table 4.1 Primitive concept detection example 1: “To transfer power source to light”

No.	Function	SPS Score
1	To support luminaires	0.16875
2	To mount a steel cover	0.00000
3	To provide enough strength for anchor bolts	0.10542
4	To facilitate installation of the pole with a handling hook or ring	0.00000
5	To accommodate luminaires	0.00000
6	To prevent from vehicle impact	0.00691
7	To attach pole to transformer base	0.01776
8	To operate high pressure sodium lamps	0.11527
9	To filter air entering the optical assembly	0.00000
10	To connect poles and other devices to grounding electrodes	0.15276
11	**To connect streetlights to power source or controller**	**0.56250**
12	To prevent inducing permanent set or injury to the cable	0.01062
13	To indicate information on reel	0.00625

Table 4.2 Primitive concept detection example 2: “To transmit power to light”

No.	Function	SPS Score
1	To support luminaires	0.00000
2	To mount a steel cover	0.00000
3	To provide enough strength for anchor bolts	0.09968
4	To facilitate installation of the pole with a handling hook or ring	0.00000
5	To accommodate luminaires	0.00000
6	To prevent from vehicle impact	0.06000
7	To attach pole to transformer base	0.00000
8	To operate high pressure sodium lamps	0.02462
9	To filter air entering the optical assembly	0.00000
10	To connect poles and other devices to grounding electrodes	0.12707
11	**To connect streetlights to power source or controller**	**0.40000**
12	To prevent inducing permanent set or injury to the cable	0.05182
13	To indicate information on reel	0.00714

Table 4.3 Primitive concept detection example 3: “To transmit power from grid to light”

No.	Function	SPS Score
1	To support luminaires	0.00000
2	To mount a steel cover	0.00000
3	To provide enough strength for anchor bolts	0.08434
4	To facilitate installation of the pole with a handling hook or ring	0.00000
5	To accommodate luminaires	0.00000
6	To prevent from vehicle impact	0.05250
7	To attach pole to transformer base	0.00000
8	To operate high pressure sodium lamps	0.02077
9	To filter air entering the optical assembly	0.00000
10	To connect poles and other devices to grounding electrodes	0.10589
11	**To connect streetlights to power source or controller**	**0.56250**
12	To prevent inducing permanent set or injury to the cable	0.04318
13	To indicate information on reel	0.00625

Similarly, the second keyword phrase changes the verb from transfer to transmit (Table 4.2). It still gives the identical priority out of the semantic search results. The third phrase changes the verb and includes another object (Table 4.3). It enriches the semantic search results with more descriptive explanation for the primitive concept. Accordingly, the eleventh function of a streetlight example, *'To connect streetlights to power source or controller'*, can be selected to be a primitive concept.

4.4. Extended Concept Detection Process

4.4.1. Extended Concept Detection Algorithm

The concept detection process is to disclose every possible function for the transformation process in a systematic manner to reduce human mistakes and errors to a minimum. In a sense, the previous section presents a semantic method to investigate the primitive concept according to the given functional keyword (phrases or sentences) from designers.

The next step is to identify all the associated functions to the primitive concept to fulfill the purpose of the concept detection process. For this step, the interaction network can be used to determine the related structural behaviors and structures in terms of the primitive concept. Then the proposed algorithm of this section will discover functions connected to the primitive concept function(s). Therefore, the final output of the concept detection includes the primitive concept function(s) and the extended concept function(s). The following algorithm describes how to determine extended functions from the primitive concept function(s).

EXTENDED CONCEPT DETECTION ALGORITHM

Step 1. Identify relevant structural behaviors that are directly associated with primitive concept function(s).

Step 2. Identify relevant structures related to the primitive concept function-based structural behaviors.

Step 3. Identify indirect structural behaviors.

Step 4. Determine all the functions associated with the indirect structural behaviors.

The first step of this algorithm is to identify relevant structural behaviors that are directly associated with primitive concept function(s). The following definition shows a set of relevant structural behaviors associated with primitive concept functions.

$\Sigma^{\lambda} = \{\sigma \mid s^{\varsigma} \in S^{\lambda}\}$ Definition (4.1)

, where σ is Behavior Schema from Definition 4.11, s^{ς} is a set of sentences that each behavior belongs to, and S^{λ} is a set of sentences that each primitive concept function belongs to.

Next, the algorithm is to detect relevant structures that are associated with the relevant structural behaviors. These structures can be found from each behavior schema from Definition 4.11. A behavior schema consists of two structures, where the behavior is associated. At this point, all the structures from these behavior schemata will be determined as follows:

$\Delta^{*} = \{\forall\delta_x \cup \forall\delta_y\}$ Definition (4.2)

, where δ_x , δ_y are structures that belong to σ, and $\sigma \in \Sigma^{\lambda}$.

This is the last step before determining the extended concept function from the primitive concept. From the previous definitions, all the relevant structures associated with the primitive concept function(s) are driven. The following definition shows the process to gather indirect structural behaviors according to the relevant structures.

$\Sigma^* = \{\sigma \mid \delta_x \text{ or } \delta_y \in \Delta^*\}$ Definition (4.3)

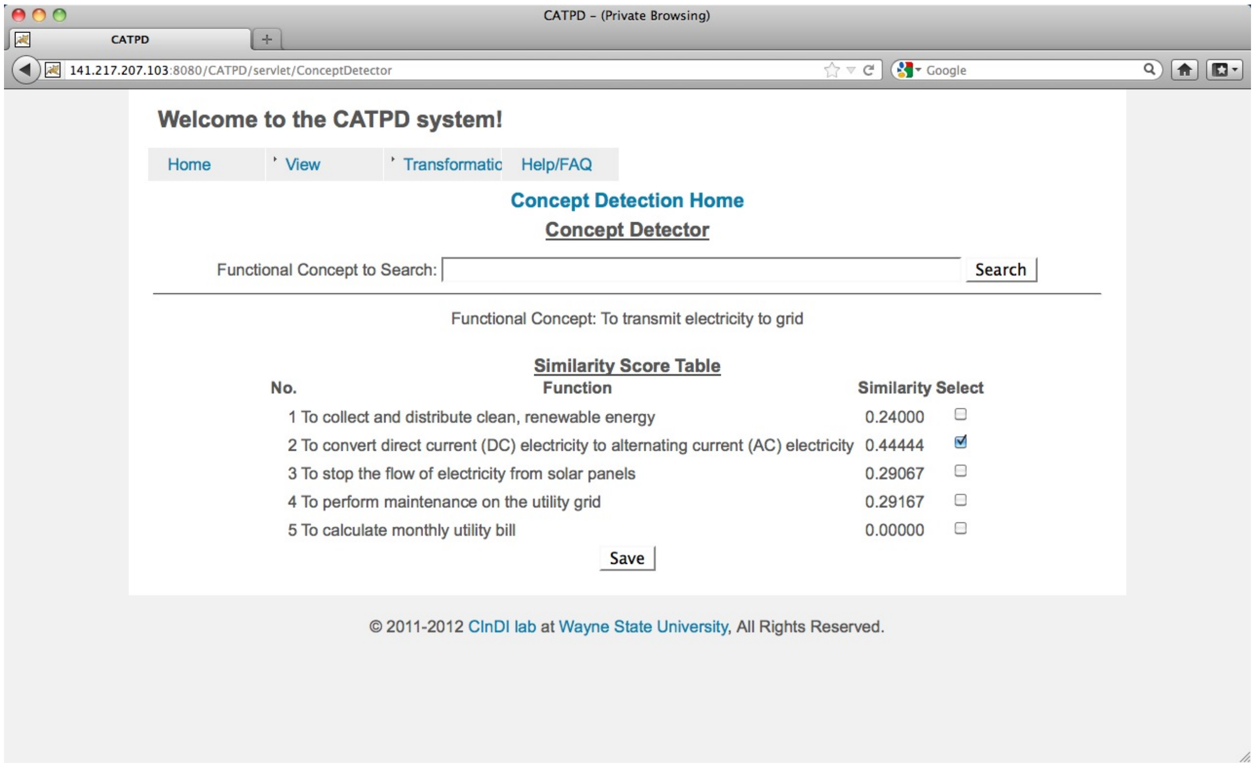
, where σ is a behavior schema, δ is a structure schema that belongs to the behavior schema σ, and Δ^* is a set of structures identified by the relevant structural behaviors. Finally, this algorithm determines extended concept functions that are associated with direct and indirect structural behaviors as follows:

$\Lambda^* = \{\lambda \mid S^{\lambda} \subseteq S^{\Sigma^*} \}$ Definition (4.4)

, where S^{λ} is a set of functional candidate sentences that belong to a function λ, and S^{Σ^*} is a set of structural candidate sentences that belong to the indirect structural behavior Σ^*. The following section presents an example of the extended concept functions by applying the proposed extended concept detection algorithm.

4.4.2. Example of Extended Concept Detection

For the illustration, this section takes another example for the extended concept detection with a grid-connected solar PV system. The tested functional keyword here is *'To transmit electricity to grid'*. This example includes the entire concept detection process from the primitive concept detection. According to the SPS score, the second function of the grid-connected solar PV system has been selected for the primitive concept (see Figure 4.3). Given this primitive concept function, the extended concept detection process continues.


CATPD – (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/ConceptDetector
Google
Home
View
Transformatic
Help/FAQ
Search
Save

Welcome to the CATPD system!

Concept Detection Home

Concept Detector

Functional Concept to Search:

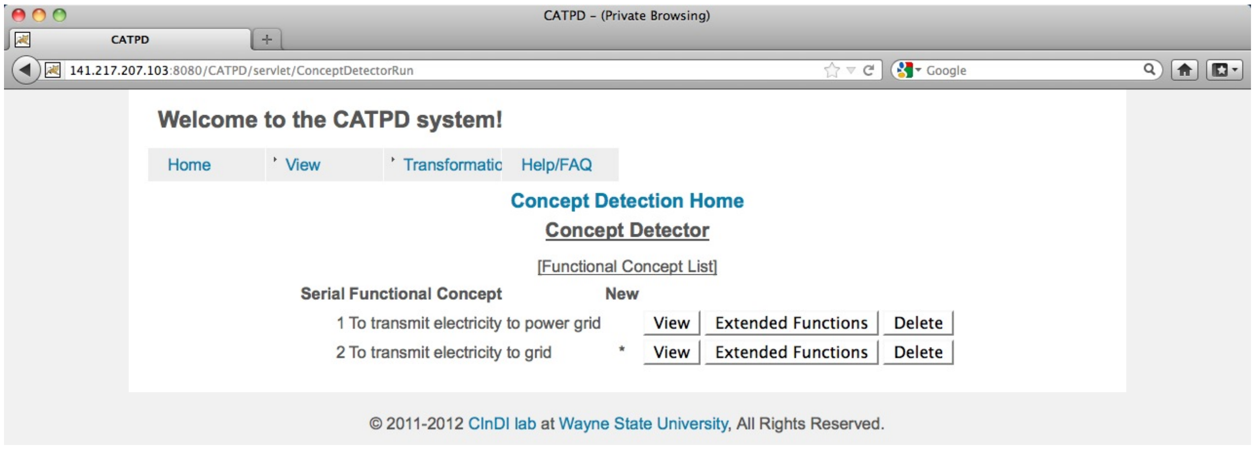
Functional Concept: To transmit electricity to grid

Similarity Score Table

No.	Function	Similarity	Select
1	To collect and distribute clean, renewable energy	0.24000	☐
2	To convert direct current (DC) electricity to alternating current (AC) electricity	0.44444	✓
3	To stop the flow of electricity from solar panels	0.29067	☐
4	To perform maintenance on the utility grid	0.29167	☐
5	To calculate monthly utility bill	0.00000	☐

© 2011-2012 CInDI lab at Wayne State University, All Rights Reserved.

Figure 4.3 Example of primitive concept detection


CATPD – (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/ConceptDetectorRun
Google
Home
View
Transformatic
Help/FAQ
View
Extended Functions
Delete

Welcome to the CATPD system!

Concept Detection Home

Concept Detector

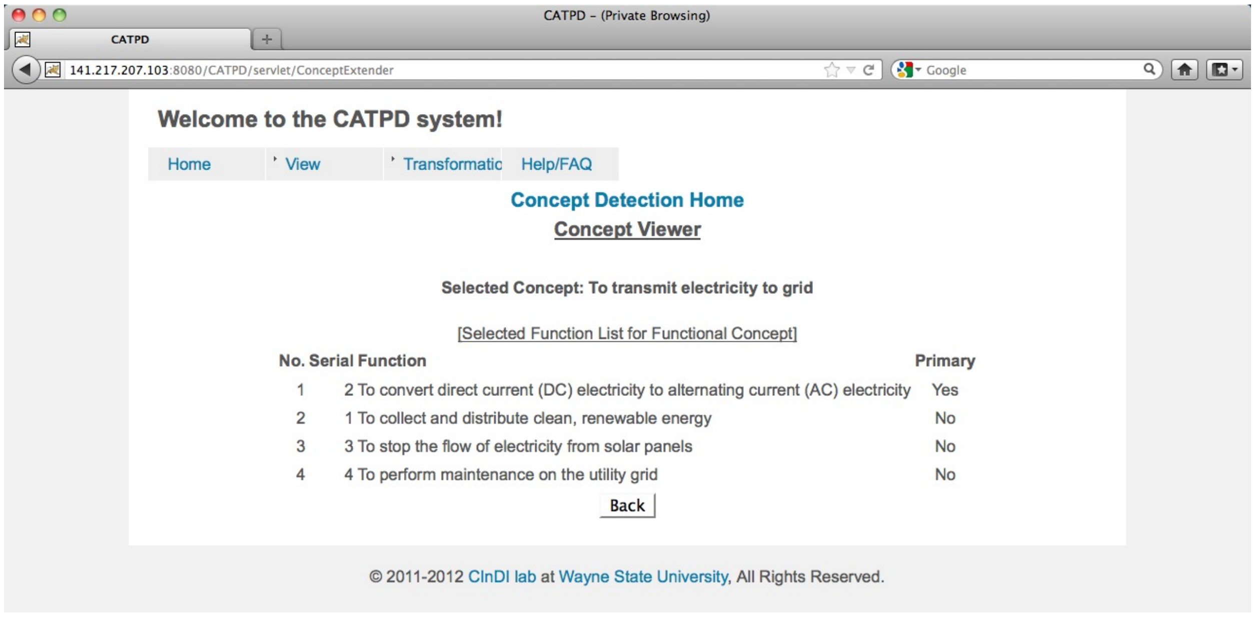
[Functional Concept List]

Serial	Functional Concept	New
1	To transmit electricity to power grid	
2	To transmit electricity to grid	*

© 2011-2012 CInDI lab at Wayne State University, All Rights Reserved.

Figure 4.4 List of detected concepts

Figure 4.4 lists the classified functional concept, including the newly generated concept. In this example, there was only one functional concept previously. The second one represents this is the newly generated functional concept. When clicking the button of Extended Functions of each functional concept, the system systematically generates the extended functions related to the primitive function for the selected functional concept by applying the proposed algorithm. Figure 4.5 shows an example of extended concept detection from the primitive concept function in Figure 4.3.


CATPD – (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/ConceptExtender
Google
Home
View
Transformatic
Help/FAQ
Back

Welcome to the CATPD system!

Concept Detection Home

Concept Viewer

Selected Concept: To transmit electricity to grid

[Selected Function List for Functional Concept]

No.	Serial Function	Primary
1	2 To convert direct current (DC) electricity to alternating current (AC) electricity	Yes
2	1 To collect and distribute clean, renewable energy	No
3	3 To stop the flow of electricity from solar panels	No
4	4 To perform maintenance on the utility grid	No

© 2011-2012 CInDI lab at Wayne State University, All Rights Reserved.

Figure 4.5 Example of extended concept detection

According to the proposed algorithm, multiple structures of a grid-connected solar PV system are identified, which are utility grid, grid, energy, panel, inverter, and home as shown in Figure 4.6. Given these structures, the algorithm detects the indirect structural behaviors and connected functions, which become the extended concept functions. In this example, there are three functions determined by the algorithm: (1) *To collect and distribute clean, renewable energy*, (2) *To stop the flow of electricity from solar panels*, and (3) *To perform maintenance on the utility grid*.

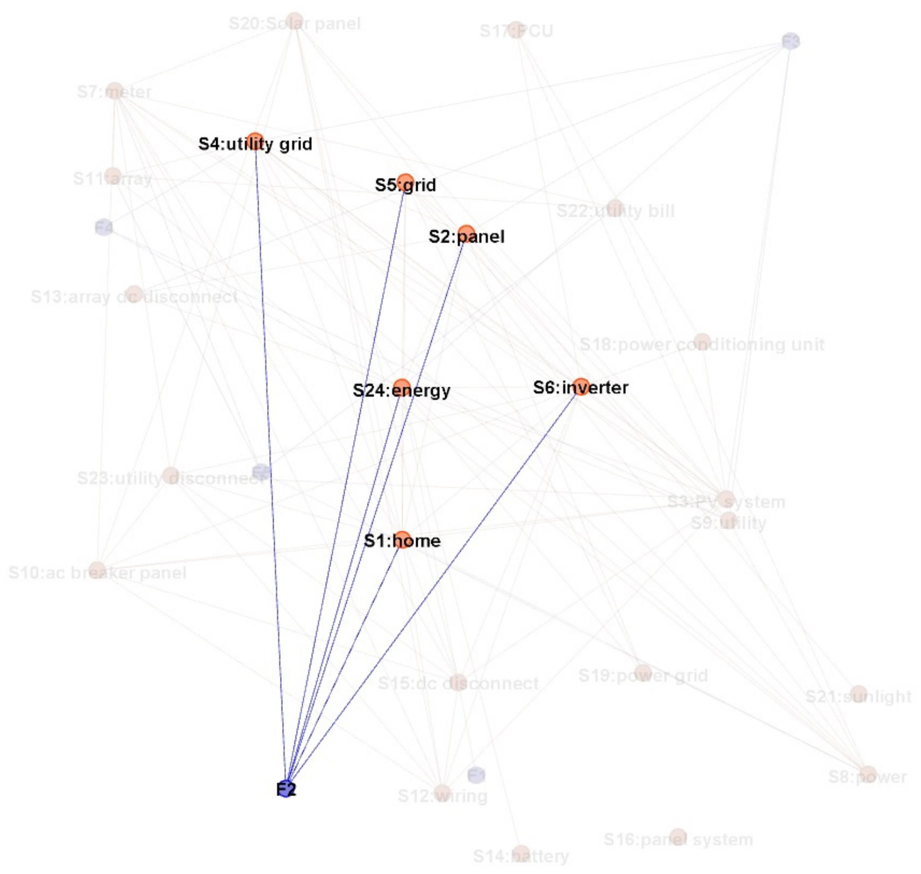


Figure 4.6 Structures associated with the primitive concept function

4.5. Conclusion

4.5.1. Summary

This chapter presents the concept detection method with the interaction network to identify relevant functions for transformative product design. The proposed concept detection process is an important process because it realizes the reusability of existing design knowledge to discover a new direction of applications from existing products or technologies.

The proposed technique intends not only to identify primitive concept functions, which are directly associated with functional keyword (phrases or sentences) to search from the interaction network in a semantic manner, but also to disclose all the relevant functions extended from the primitive concept function(s). Appropriate examples with a grid-connected solar PV system and a streetlight are given to illustrate the feasibility of the proposed methods.

4.5.2. Contribution

To fulfill the concept detection process to utilize existing design knowledge and become the source of the later transformation process, a new similarity score based on the combination of stem words and the *part-of-speech* of each word is proposed here, called Stem-POS based Similarity (SPS) Score. A semantic strategy of the score takes into account two perspectives of the subject words to compare: word match and structure match. This strategy enables the semantic comparison, even if words are different with respect to their word classes, by following the recommendation of the previous research works in natural language processing applications. All the similarity score is dependent on the cardinality of possible matched semantic words such as synonyms, hypernyms, hyponyms, holonyms, and meronyms.

Another contribution of the proposed method is to provide a complementary detection technique in consideration of the structure of interaction network. Even if the semantic strategy to retrieve design knowledge allows more freedom to designers, it could drop an important concept function that has to be associated. To prevent this challenging problem, the extended concept function detection algorithm is presented.

CHAPTER 5

TRANSFORMATIVE DESIGN GENERATION

5.1. Introduction

As addressed in the research objectives, the final step is the design transformation process, which is the beauty of this research. We have utilized existing design knowledge and have detected functional concept out of the extracted design knowledge in previous chapters. They are preconditions to continue this transformation process. The journey from the identification of design knowledge written in a natural language has produced the outcome of the functional concept to apply for new transformative applications across domains. Therefore, this chapter has to provide the comparison of design knowledge to accelerate the design transformation process.

In the information sharing approach in product design, they have focused on building a thesauri or ontologies for domain experts in different arrears to communicate with and interpret design knowledge in different terms [Ahmed *et al*., 2007; Nanda *et al*., 2006; Yang *et al*., 2005]. Their research efforts might be beneficial to this research if they provide input sources for the semantic approach this research takes. However, their efforts have still narrowed down to specific domains.

To realize a transformative design environment, this research adopts a general lexical dictionary that contains semantic words such as synonyms, hypernyms, hyponyms, and meronyms, across domains. To complement the limitation of such a semantic approach, this research also takes into account the proposed design representation model to analyze the structural interpretation of design knowledge with the semantic results. Thus, the research strategy taken in Chapter 4 continues in this chapter too. The most challenging issue in this chapter is to check the degree of transformability of other products in comparison with the detected functional concept in Chapter 4. Details are introduced in the following sections.

5.2. Transformation Process

The transformation process in TPD starts from the concept detection process. In the transformation process, functional concepts from other domains will be checked for the transformability that presents how those functional concepts are associated with the selected functional concept systematically. When recalling the motivation of the transformative product design, one of key characteristics is the reusability of existing design knowledge for a new, innovative application by converging individual excellent technology advances out of multiple products with a minimum human intervention in a systematic manner.

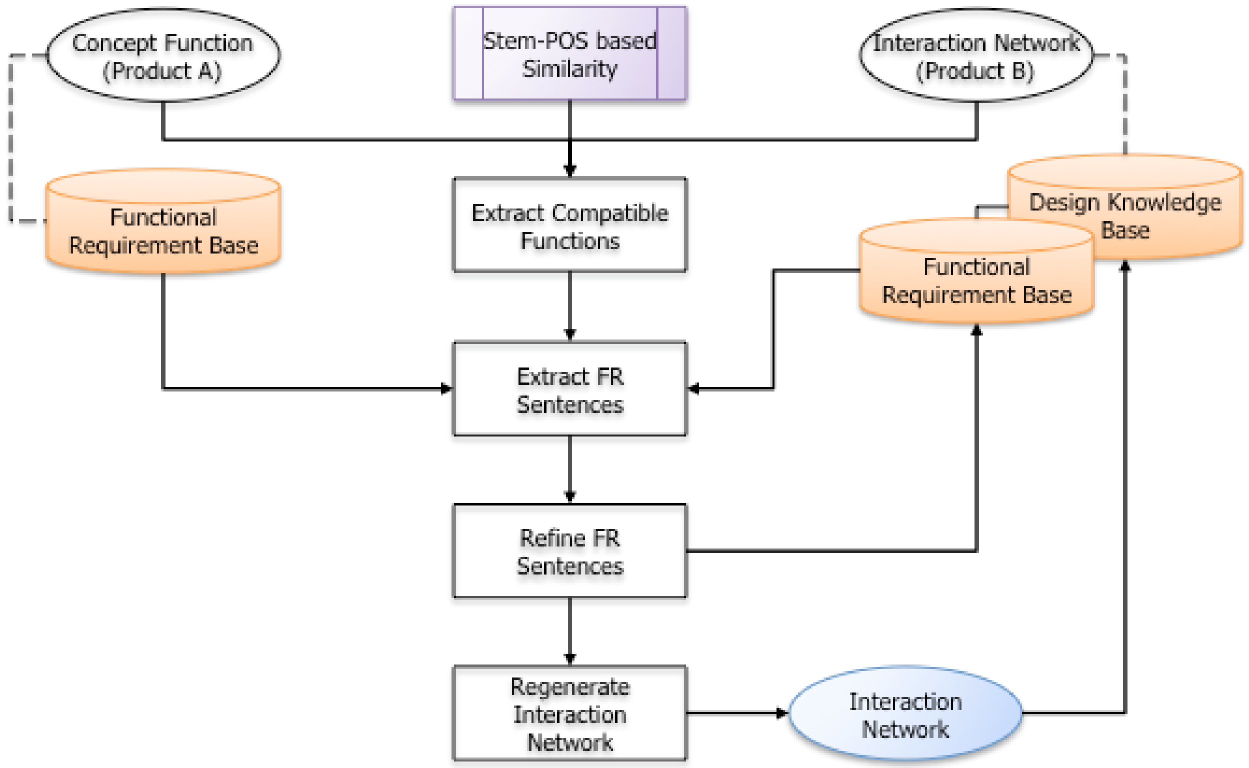

Concept Function (Product A)
Stem-POS based Similarity
Interaction Network (Product B)
Functional Requirement Base
Extract Compatible Functions
Design Knowledge Base
Functional Requirement Base
Extract FR Sentences
Refine FR Sentences
Regenerate Interaction Network
Interaction Network

Figure 5.1 Transformation process

Another challenging issue is of how to gather and connect functions and structures from different concepts. The transformation process is not a simple process. Especially, the assembled physical artifacts for the transformative product design have a possibility of being represented in different objects, even if they are the same or interchangeable. This chapter, therefore, proposes a method to assist the challenge in order to refine the functional requirement document in the transformation process in consideration of the compatibility issue. Figure 5.1 illustrates the transformation process that starts from the concept detection outcomes.

5.2.1. Transformability

As shown in Figure 5.1, the concept detection outcomes can be compared with the interaction network of a product. They also can be compared another concept detection outcomes. In this comparison, the proposed algorithm will first confirm if they are compatible each other. If there is no compatible function for the two outcomes, it may require an additional concept function that can weave them.

The transformability check is a key characteristic of the transformative algorithm in the next section. The transformability is the degree of how much one conceptual function is associated with and compatible to another concept function(s) or plain functions for the transformation process. In order to continue the transformation process, this transformability check process is the very first, significant process. To enable this transformability check, this research takes the semantic strategy taken in Chapter 4 again.

Let's recall the previous Stem-POS based Similarity (SPS) score. One of a pair of phrases to compare in the previous score is replaced with the select functional concept (Λ^*_i) for the design transformation, while the other is replaced with functions or functional concepts (Λ^{*k}_j) of other products (D_j) in the interaction network repository. Among the results from the comparison, the maximum *SPS* score will be used to determine the *degree of transformability*. Consequently, the prototype system implements this degree of transformability to retrieve all functions or functional concepts from the interaction network repository.

DEFINITION: DEGREE OF TRANSFORMABILITY

Degree of Transformability(Λ^*_i, D_j) = max{*SPS Score*(Λ^*_i, Λ^{*k}_j)}, where

- Λ^*_i is the selected functional concept (i.e., a new technology for the design transformation) of the ith product in the design repository.
- D_j is the jth product design in the design repository and $i \neq j$.
- Λ^{*k}_j is the kth functional concept in D_j.
- *SPS Score*(Λ^*_i, Λ^{*k}_j) = max{*SPS Score*($\lambda_{i,q}$, $\lambda_{j,r}^k$)}, where

  $\lambda_{i,q}$ is the qth function in Λ^*_i and $\lambda_{j,r}^k$ is the rth function in Λ^{*k}_j.

One thing to notice here is the transformation threshold (κ), which is pre-determined. Thus, when designers request to retrieve transformation candidates, every product with greater degree of transformability than the threshold (*Degree of Transformability*(Λ^*_i, D_j) $\geq \kappa$) becomes the transformative candidates.

5.2.2. Transformation

After the compatibility (i.e. transformability) check, the proposed algorithm continues to extract the functional requirement sentences and rearrange them into a new functional requirement document, which will be eventually the input resource for the transformative interaction network. Finally, the refined document drives the algorithm to generate the interaction network as its design model, which will be placed into the functional requirement base and design knowledge base for future transformation knowledge. The following algorithm represents the transformation process.

TRANSFORMATION ALGORITHM

Step 1. Check the transformability between the concept functions.

Step 2. Determine which concept functions will be used for the transformation.

Step 3. Extract the functional requirement sentences from the original interaction networks.

Step 4. Rearrange the sentences into a new design or the selected design.

Step 5. Determine the interaction network based on the refined sentences.

First, recall the definition of concept functions in Chapter 4 as shown below:

$\Lambda^{*} = \{\lambda \mid S^{\lambda} \subseteq S^{\Sigma *} \}$ Definition (4.4)

, where S^{λ} is a set of functional candidate sentences that belong to a function λ, and $S^{\Sigma *}$ is a set of structural candidate sentences that belong to the indirect structural behavior Σ^{*} (from Definition 4.3). Based on this definition, assume that there are two concept functions, Λ^{1*} and Λ^{2*}. In order to check their transformability, the proposed similarity score in Chapter 4 is revisited. It can be said that the two concept functions are transformable when $SPS\ Score(\lambda_i^1, \lambda_i^2) \geq \kappa$, where κ is a predetermined threshold for the transformability.

The next step is the reverse process for the extended function identification in Chapter 4 in order to find all the direct and indirect functional candidate sentences. The indirect functional candidate sentences are the structural candidate sentences associated with the concept functions. For this step, recall the definitions of all the schemata in Chapter 3.

λ_l = <ID, Function, S^{λ}> Definition (3.3)

δ_m = <ID, Structure, S^{sc}> Definition (3.8)

σ_n = <ID, Behavior, δ_x, δ_y, ς^{xy}> Definition (3.11)

ID is the identifier of each schema and Function, Structure, and Behavior represent descriptive names of each schema. As depicted in the definitions, each schema contains its relevant functional requirement sentences. The Behavior Schema in Definition 3.11 can track its behavior candidate sentences from its behavioral verb property, ς^{xy}. The behavioral verb is defined as follows:

$\varsigma^{xy} = \delta_x \bullet \delta_y = \{\mathrm{VB} \mid \forall s \in S^{sc}{}_x \cap S^{sc}{}_y\}$ Definition (3.9)

Accordingly, the algorithm extracts all the functional requirement sentences from Definition 4.1 of a set of relevant structural behaviors associated with primitive concept functions through Definition 4.4 of the concept function in a reverse manner. The final step of the algorithm follows all the process of the interaction network identification algorithm in Chapter 3. The following section presents a case study with a grid-connected solar PV system to show the feasibility of the proposed methods from Chapter 3 to Chapter 5.

5.3. Case Study with a Solar PV System

Recently renewable energy applications have obtained significant attentions for better sustainable environment, including products and architectures. A solar photovoltaic system is a promising technology out of famous renewable technologies. Thus this section takes a grid-connected solar photovoltaic system as an example. To remind, the ultimate goal of this process is to find a new application. As shown in Figure 5.2, the transformative design alternative generation process starts from the design knowledge extraction. The results of this design knowledge extraction have illustrated in Chapter 3.

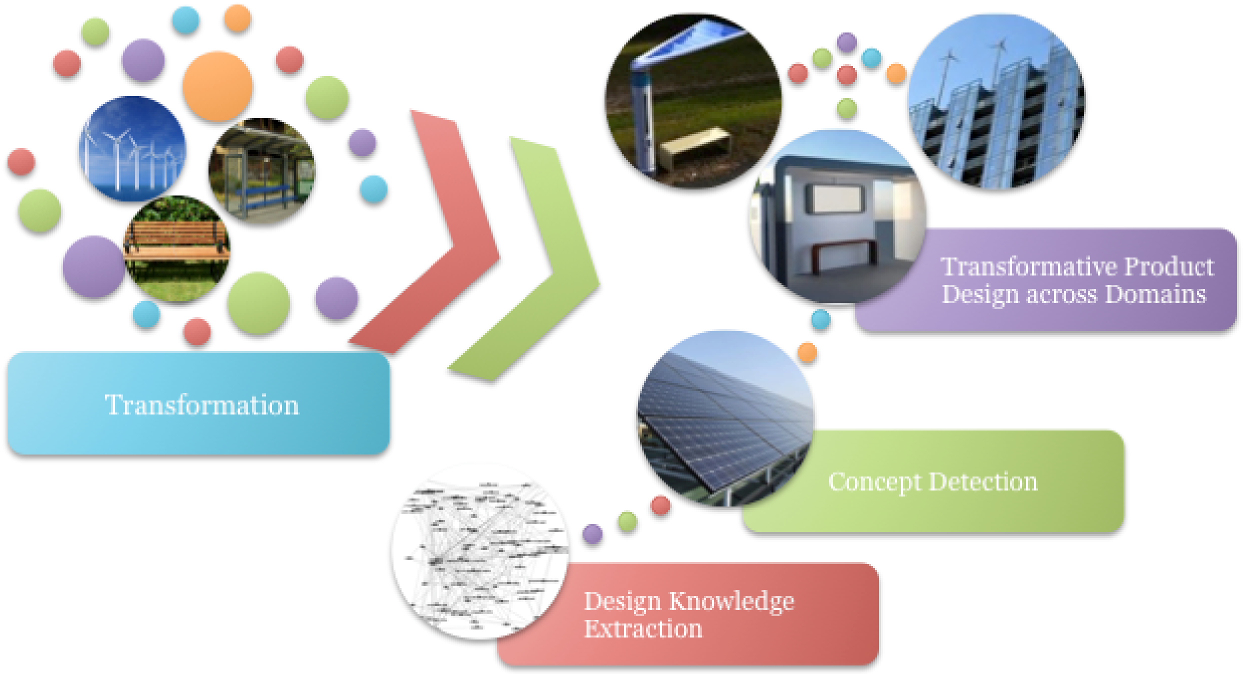
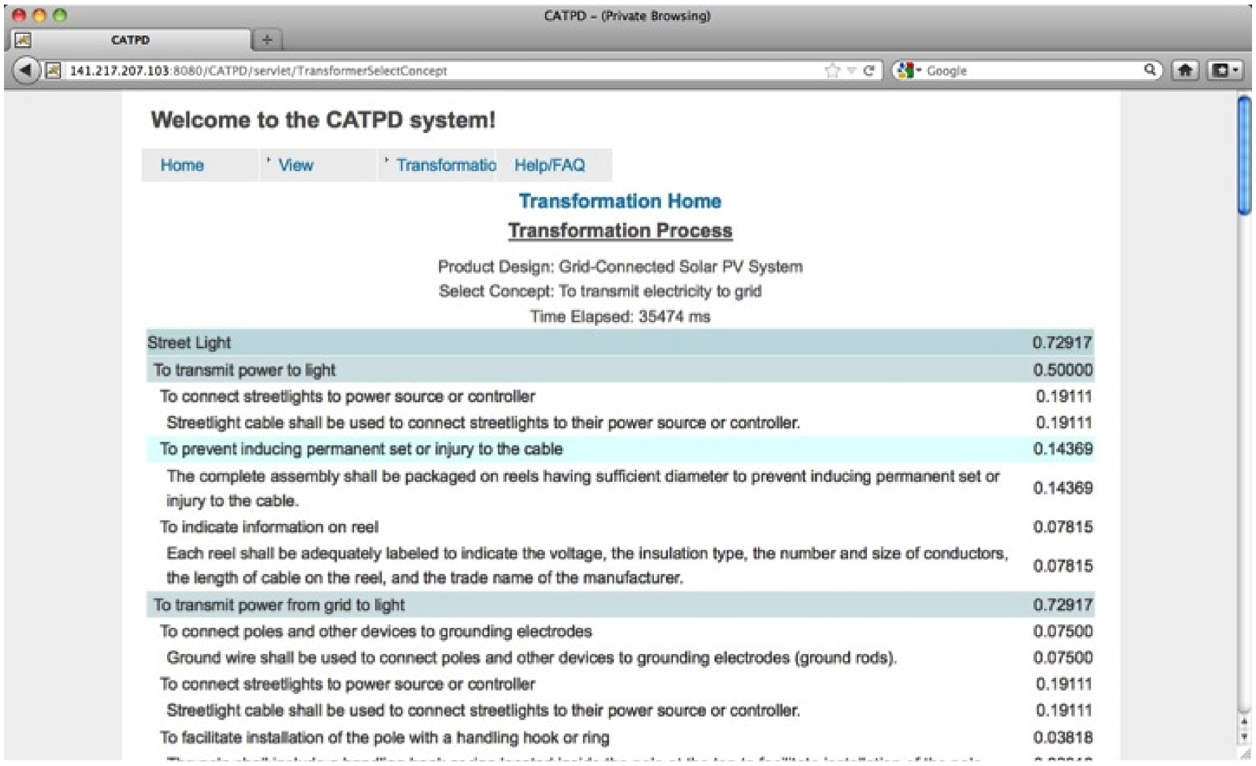

Transformation
Transformative Product Design across Domains
Concept Detection
Design Knowledge Extraction

Figure 5.2 Transformative product design example scenario

Once the interaction network of the selected product design, designers are able to detect functional concepts of excellent, highly advanced technologies or design solutions by using the proposed concept detection techniques. The outcome of this process is represented in Chapter 4. In the previous chapter, there are two functional concepts: (1) *To transmit electricity to power grid*, and (2) *To transmit electricity to grid*. To continue, the second functional concept has been selected here. (Technically, there is no difference between the two functional concepts because they have detected same functions.) Figure 5.3 is the outcome of the transformability check with the select functional concept when the threshold is set to 0.70.


CATPD – (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/TransformerSelectConcept
Google
Home
View
Transformatio
Help/FAQ

Welcome to the CATPD system!

Transformation Home

Transformation Process

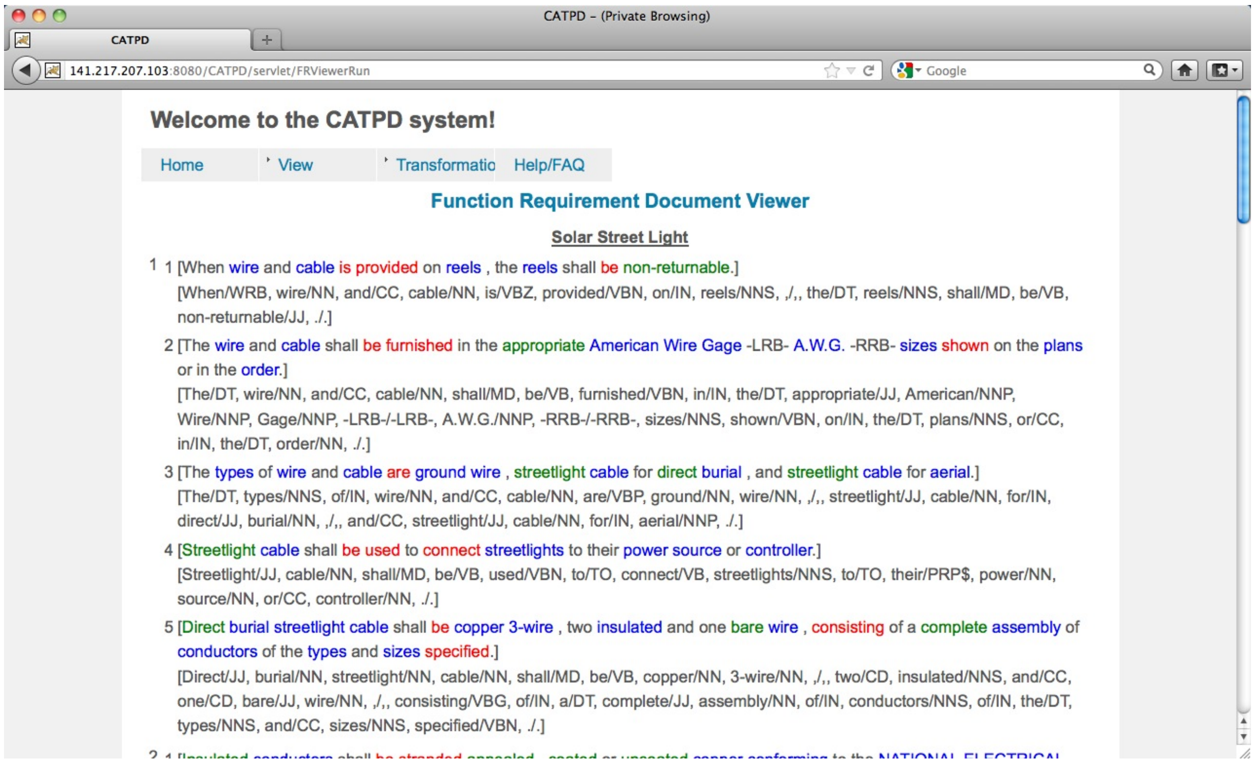
Product Design: Grid-Connected Solar PV System
Select Concept: To transmit electricity to grid
Time Elapsed: 35474 ms

Street Light	0.72917
To transmit power to light	0.50000
To connect streetlights to power source or controller	0.19111
Streetlight cable shall be used to connect streetlights to their power source or controller.	0.19111
To prevent inducing permanent set or injury to the cable	0.14369
The complete assembly shall be packaged on reels having sufficient diameter to prevent inducing permanent set or injury to the cable.	0.14369
To indicate information on reel	0.07815
Each reel shall be adequately labeled to indicate the voltage, the insulation type, the number and size of conductors, the length of cable on the reel, and the trade name of the manufacturer.	0.07815
To transmit power from grid to light	0.72917
To connect poles and other devices to grounding electrodes	0.07500
Ground wire shall be used to connect poles and other devices to grounding electrodes (ground rods).	0.07500
To connect streetlights to power source or controller	0.19111
Streetlight cable shall be used to connect streetlights to their power source or controller.	0.19111
To facilitate installation of the pole with a handling hook or ring	0.03818

Figure 5.3 Example of the transformability check

In Figure 5.3, the top row represents the overall degree of transformability of the product, followed by functional concepts, functions, and functional sentences. The overall degree of a product is the maximum degree of transformability among functional concepts, functions, and functional sentences. In this case, a functional concept of a streetlight, *'To transmit power from grid to light'*, results in the maximum degree. Thus, let's continue with the selected functional concepts from a solar PV system and a streetlight.

Following figures represent the outcomes according to the transformation algorithm. Figure 5.4 represents the refined functional requirement document from the two concept functions: one concept function, *'To transmit electricity to grid'*, comes from the example of a grid-connected solar PV system, and the other concept function, *'To transmit power from grid to light'*, results from the example of a streetlight.


CATPD – (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/FRViewerRun
Google
Home
View
Transformatio
Help/FAQ

Welcome to the CATPD system!

Function Requirement Document Viewer

Solar Street Light

1 1 [When wire and cable is provided on reels , the reels shall be non-returnable.]
[When/WRB, wire/NN, and/CC, cable/NN, is/VBZ, provided/VBN, on/IN, reels/NNS, ,/,, the/DT, reels/NNS, shall/MD, be/VB, non-returnable/JJ, ./.]

2 [The wire and cable shall be furnished in the appropriate American Wire Gage -LRB- A.W.G. -RRB- sizes shown on the plans or in the order.]
[The/DT, wire/NN, and/CC, cable/NN, shall/MD, be/VB, furnished/VBN, in/IN, the/DT, appropriate/JJ, American/NNP, Wire/NNP, Gage/NNP, -LRB-/-LRB-, A.W.G./NNP, -RRB-/-RRB-, sizes/NNS, shown/VBN, on/IN, the/DT, plans/NNS, or/CC, in/IN, the/DT, order/NN, ./.]

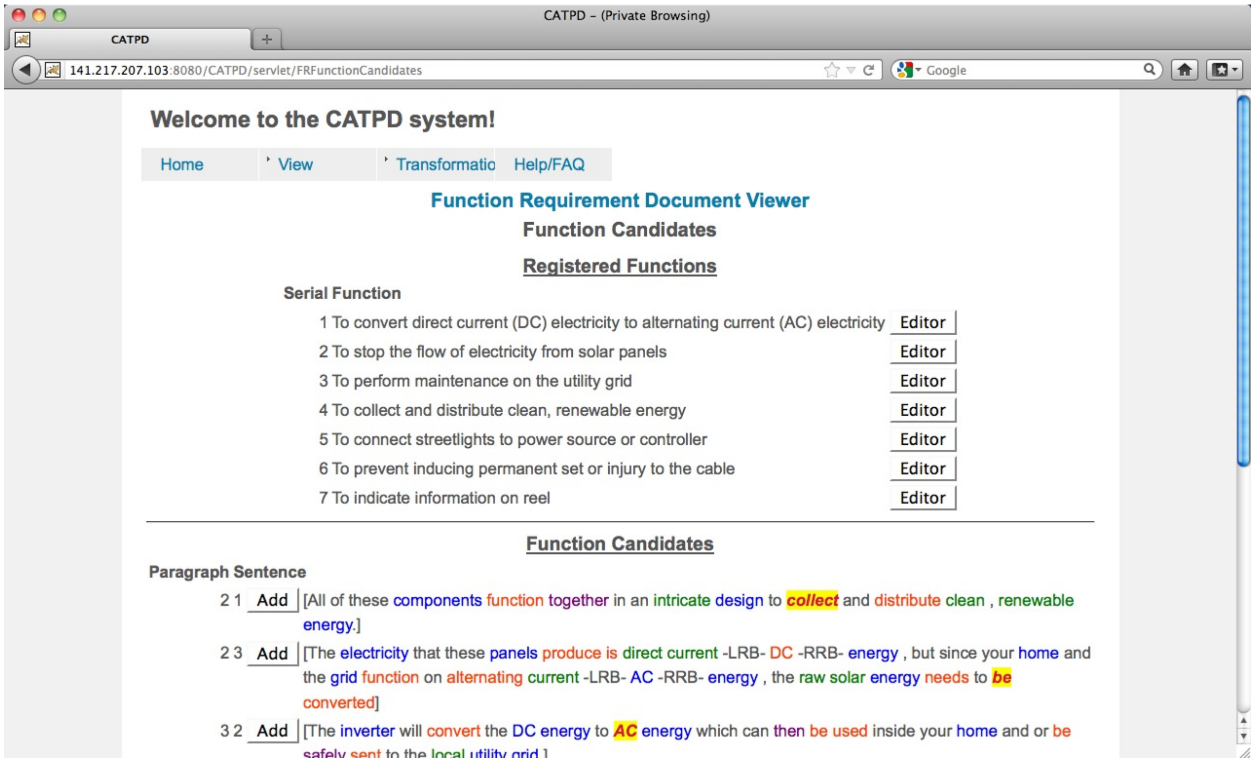
3 [The types of wire and cable are ground wire , streetlight cable for direct burial , and streetlight cable for aerial.]
[The/DT, types/NNS, of/IN, wire/NN, and/CC, cable/NN, are/VBP, ground/NN, wire/NN, ,/,, streetlight/JJ, cable/NN, for/IN, direct/JJ, burial/NN, ,/,, and/CC, streetlight/JJ, cable/NN, for/IN, aerial/NNP, ./.]

4 [Streetlight cable shall be used to connect streetlights to their power source or controller.]
[Streetlight/JJ, cable/NN, shall/MD, be/VB, used/VBN, to/TO, connect/VB, streetlights/NNS, to/TO, their/PRP$, power/NN, source/NN, or/CC, controller/NN, ./.]

5 [Direct burial streetlight cable shall be copper 3-wire , two insulated and one bare wire , consisting of a complete assembly of conductors of the types and sizes specified.]
[Direct/JJ, burial/NN, streetlight/NN, cable/NN, shall/MD, be/VB, copper/NN, 3-wire/NN, ,/,, two/CD, insulated/NNS, and/CC, one/CD, bare/JJ, wire/NN, ,/,, consisting/VBG, of/IN, a/DT, complete/JJ, assembly/NN, of/IN, conductors/NNS, of/IN, the/DT, types/NNS, and/CC, sizes/NNS, specified/VBN, ./.]

2 1 [Insulated conductors shall be stranded annealed , coated or uncoated copper conforming to the NATIONAL ELECTRICAL

Figure 5.4 Refined functional requirement document


CATPD – (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/FRFunctionCandidates
Google
Home
View
Transformatio
Help/FAQ
Editor
Add

Welcome to the CATPD system!

Function Requirement Document Viewer

Function Candidates

Registered Functions

Serial Function

1 To convert direct current (DC) electricity to alternating current (AC) electricity
2 To stop the flow of electricity from solar panels
3 To perform maintenance on the utility grid
4 To collect and distribute clean, renewable energy
5 To connect streetlights to power source or controller
6 To prevent inducing permanent set or injury to the cable
7 To indicate information on reel

Function Candidates

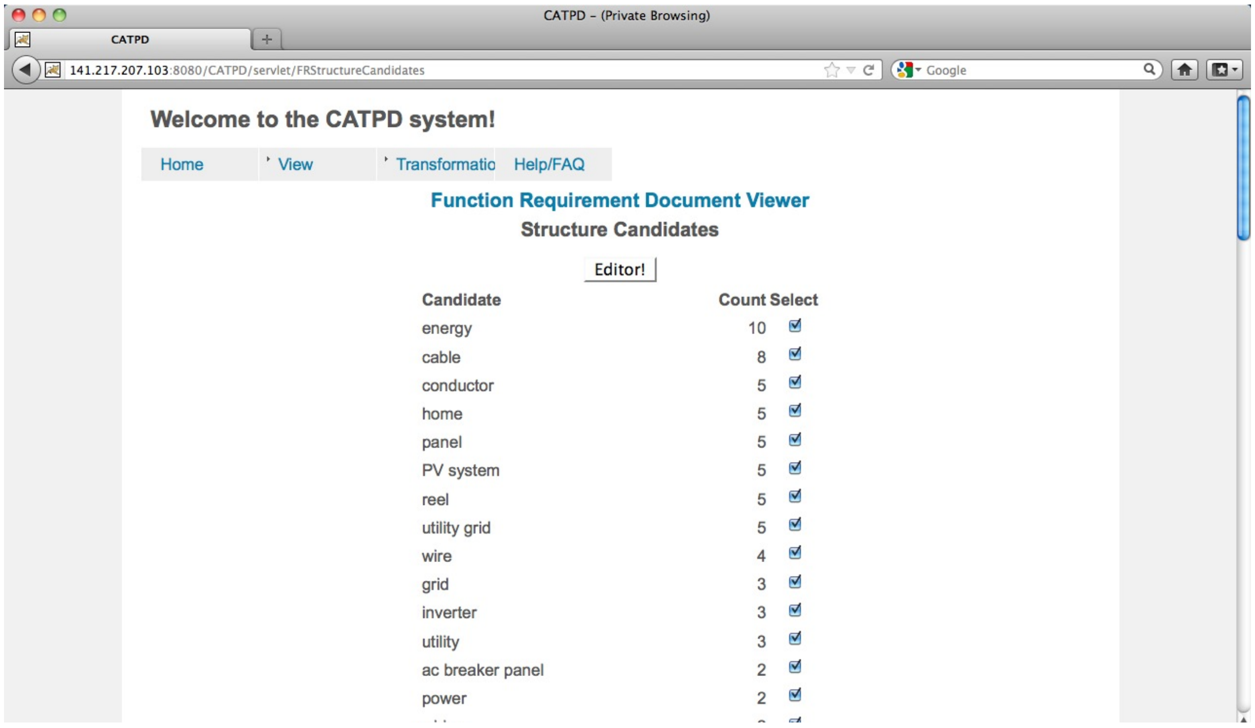
Paragraph Sentence

2 1 [All of these components function together in an intricate design to ***collect*** and distribute clean , renewable energy.]

2 3 [The electricity that these panels produce is direct current -LRB- DC -RRB- energy , but since your home and the grid function on alternating current -LRB- AC -RRB- energy , the raw solar energy needs to ***be*** converted]

3 2 [The inverter will convert the DC energy to ***AC*** energy which can then be used inside your home and or be safely sent to the local utility grid.]

Figure 5.5 Functions after transformation


CATPD - (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/FRStructureCandidates
Google
Home
View
Transformatio
Help/FAQ
Editor!

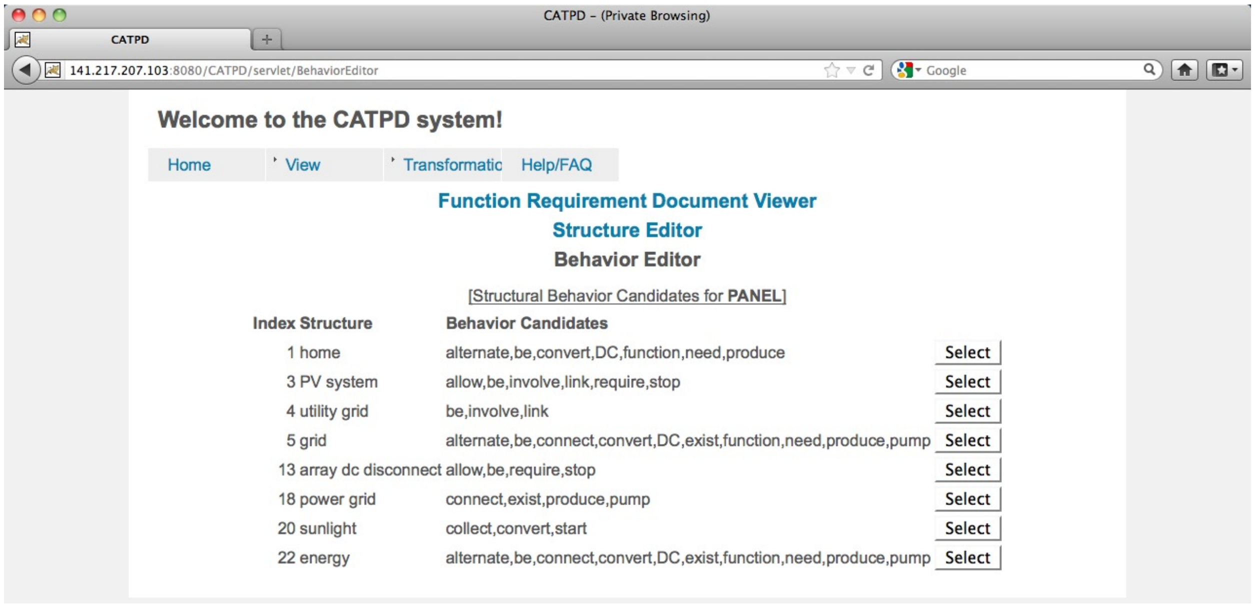
Welcome to the CATPD system!

Function Requirement Document Viewer

Structure Candidates

Candidate	Count	Select
energy	10	✓
cable	8	✓
conductor	5	✓
home	5	✓
panel	5	✓
PV system	5	✓
reel	5	✓
utility grid	5	✓
wire	4	✓
grid	3	✓
inverter	3	✓
utility	3	✓
ac breaker panel	2	✓
power	2	✓

Figure 5.6 Structures after transformation


CATPD - (Private Browsing)
CATPD
141.217.207.103:8080/CATPD/servlet/BehaviorEditor
Google
Home
View
Transformatic
Help/FAQ

Welcome to the CATPD system!

Function Requirement Document Viewer

Structure Editor

Behavior Editor

[Structural Behavior Candidates for **PANEL**]

Index	Structure	Behavior Candidates	
1	home	alternate,be,convert,DC,function,need,produce	Select
3	PV system	allow,be,involve,link,require,stop	Select
4	utility grid	be,involve,link	Select
5	grid	alternate,be,connect,convert,DC,exist,function,need,produce,pump	Select
13	array dc disconnect	allow,be,require,stop	Select
18	power grid	connect,exist,produce,pump	Select
20	sunlight	collect,convert,start	Select
22	energy	alternate,be,connect,convert,DC,exist,function,need,produce,pump	Select

Figure 5.7 Structural behaviors after transformation

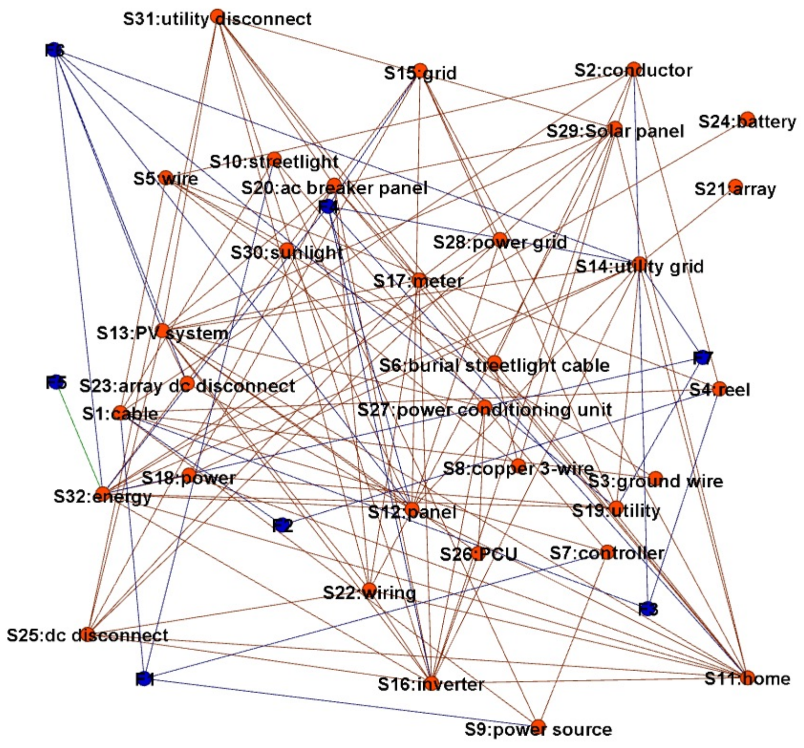


Figure 5.8 Interaction network of the transformative product design

Figure 5.5 lists all the functions that come from the two concept functions. Figure 5.6 arranges all the structures that are associated with the two concept functions. Figure 5.7 displays all the structural behaviors that describe the relationships between the structures rearranged from the transformation process.

Figure 5.8 is the visualization of the generated interaction network that contains two functional concepts. This visualized interaction network model is useful for designers to understand the transformed design knowledge for a new application, starting from a grid-connected solar PV system, especially the functional concept of *'to transmit electricity to grid'*. This kind of solar PV system based streetlight transformative example is a well-known application in industries already. However, without this transformative product design environment, designers would struggle to extract design knowledge, to determine functional concept without missing any relevant functions, and to generate transformative conceptual design alternatives. It is not only a complicated, cumbersome problem, but also a time consuming task, which is undesirable.

5.4. Conclusion

5.4.1. Summary

This chapter presents an algorithm to execute the transformation process with the given concept functions from the previous chapter. This process is to utilize the foundation that has been constructed by the algorithms proposed in Chapter 3 and 4. The concept detection technique and the transformation algorithm could assist the complicated task to interweave the existing design knowledge across domains efficiently.

The results, as shown in the examples of this chapter, represent reasonable design models in the perspective of the traditional functional modeling. However, the generated design model (i.e. an interaction network) is limited to explain the detail relationship between physical artifacts, such as conditional context and control variables. The limitation of this research remains for the future research direction, which is discussed in Chapter 8.

5.4.2. Contribution

A major contribution of this method is to provide conceptual design alternatives systematically and efficiently for the transformative product design. Individual designers have limited design knowledge of their own design expertise. In a sense, even though they are professionals in a specific domain, it could not be a trivial, apparent task to design a new product that embraces various advanced technologies across multiple domains.

CHAPTER 6

TRANSFORMATIVE PRODUCT DESIGN VALIDATION

6.1. Introduction

To fulfill the transformative product design paradigm, previous three chapters proposed methods to systematically extract design knowledge from functional requirements in informal free text in a natural language, to identify semantic-based concept function, and to generate transformative conceptual design alternative with the design transformation process. As this research presents a new design paradigm and the corresponding research methods, there is no previous relevant research works to compare or validate.

There have been research efforts to classify the types of convergence or to identify the necessity of convergence [Kodama, 1992; Iansiti, 1997; Iansiti and West, 1997; Stieglitz, 2003]. These research works have been reported in a managerial point of view to determine the convergence types. Representatively, Stieglitz (2003) summarized the industry convergence types with respect to the demand and supply sides and the substitutability and complementarity of products. He focused on industry convergence rather than product convergence. Nevertheless, since he assumed that industry consists of a group of products that have similar characteristics, this research considers the types of industry convergence as product convergence in a narrower point of view. The following sections consist of two major parts: (1) introduction of industry convergence and (2) validation.

6.2. Industry Convergence

A representative discourse on key characteristics of the new digital economy is a contention of how multiple industries, such as telecommunication, computer, and entertainment, would converge into a huge industry of multimedia [Collis et al., 1997; The Economist, 2000].

As the major talking points in industries are sustainability and renewable energy technologies recently, the attention to convergent technologies or convergent products has risen even higher. However, the concept of industry convergence has not appeared out of sudden.

In 1963, Rosenberg addressed a similar concept of industry convergence for the early evolution in the US machine tool industry [Rosenberg, 1963]. He focused on an industry, which had been emerged one century earlier. That was the machine tool industry, which was a groundbreaking, creative new industry with the mechanical skills, such as drilling, grinding, and polishing. Rosenberg determined this process as technological convergence.

Different industries increasingly depended on the same set of technological techniques in their manufacturing processes. Different products from a market perspective became significantly related to each other from a technological point of view. In the 19th century, examples of technologically convergent industries include firearms, sewing machines and bicycles. They were correlated from a manufacturing process point of view with the same machining tools.

Sahal (1985) and Dosi (1988) argued that certain technological paradigms disseminate their effects across various industries and lead innovative activities even in sluggish markets. Good examples can include digital display technologies like multi-touch screens and their impact on the smartphones, tablet computers, and consumer electronics industries. Many management and business scholars on corporate strategy and industry dynamics prominently studied this idea of industry convergence. According to the claim of the boundary change of traditional industries by technological innovations [Porter, 1985; Hamel and Prahalad, 1994], different existing products may become closer substitutes, since their functions are increasingly incorporated and combined in innovative products [Katz, 1996; Yoffie, 1997]. Hence, either technologies or products can drive industry convergence. Bettis and Hitt (1995) took into account the strategic response to this ‘new competitive landscape’, which is an increased reliance on corporate networks and strategic alliances [Gomes-Casseres, 1996] and the migration and diversification of companies into convergent industries [Wirtz, 2001]. Despite the notion of industry convergence has not been emerged surprisingly to characterize industrial dynamics in the new digital economy, it is not clearly defined [Katz, 1996].

In the next section, in order to analytically clear the concept of industry convergence, four types of industry convergence are defined. The developed taxonomy allows us to identify and disentangle different drivers of industrial convergence, which can be applied for transformative product design paradigm.

6.2.1. Definition

Stieglitz (2003) tried to specify the notion of industry convergence by reflecting upon what is actually meant when talking about industries and markets. This chapter summarizes his research contribution to the identification of the types of industry convergence and the associated research of other researchers. Stieglitz adopted the definition of the substitutability of products [Geroski, 1998; Abbott, 1955; Lancaster, 1966]. Lancaster (1996) emphasizes that consumers do not derive their utility from the product, but from the embodied mixture of product characteristics. In other words, products can be considered substitutes when they share common characteristics, even with the variation of the exact combination or specification of the product characteristics. Consequently, Stieglitz determines the boundary of an industry by grouping products with similar characteristics. Interestingly, there are many examples for consumers to derive or to increase the utility of a product by jointly using this product and other products (e.g., a video recorder and a video cassette, or a CD player, an amplifier and speakers). Thus, products could have both *substitutability* and *complementarity*.

Stieglitz (2003) also summarized a different approach to defining industries. While literature aforementioned in the previous paragraph takes the demand-side point of view, there is also the issue from the supply side. According to the resource-based view of the firm, a firm competes against other firms with similar resources [Bettis, 1998]. Accordingly, industries could be defined as groups of firms with respect to the similar resource basis. Based on two perspectives of the *demand* side (products) and the *supply* side (resources, technologies), Stieglitz defines the concept of industries.

Next, Stieglitz (2003) adopts the definition of *convergence* from the Merriam-Webster (2000) dictionary to define industry convergence. The dictionary defines the word, convergence, as the movement towards the same point or the *'meeting'* of two or more elements. Thus, he concludes that industry convergence involves the 'meeting' of, at least, two industries. In addition, he continues to hold the two different perspectives toward the concept of industry, the demand side and the supply side. A distinction can be made between technology-based convergence from the supply criteria and product-based convergence from the demand criteria.

Another distinction can be further applied for each criterion. According to the product characteristics and the utility of customers, Stieglitz (2003) determines that products can have substitutability and complementarity. Based on these, the previous two convergence types are classified into two sub-types each: technology-based convergence includes *technology substitution* and *technology integration* while product-based convergence does *product substitution* and *product complementarity*.

Technology substitution involves a new technology replacing different technologies in established industries. Technology integration represents the phenomena that various technologies previously fallen into different industries are fused or integrated [Kodama, 1992; Iansiti, 1997]. Technology integration may give rise to entirely new industries. Greenstein and Khanna (1997) described that product-based convergence driven industries can be characterized by substitutable products or complementary product characteristics. For the former case, Stieglitz (2003) described that an established product from one industry evolves to integrate product features that are similar to those of another product in a different industry. It is called product complementarity as two formerly unrelated products are turned into complements.

The taxonomy is summarized in Table 6.1.

Table 6.1 Types of industry convergence [Stieglitz, 2003]

	Substitutes	**Complements**
Technology-based convergence	Technology substitution	Technology integration
Product-based convergence	Product substitution	Product complementarity

Stieglitz (2003) employed a framework, advanced by Saviotti and Metcalfe (1984) and Saviotti (1996), which integrates the supply and demand characteristics of products and industries, in order to illustrate each of the four types of industry convergence. Figure 6.1 shows how to represent a particular product by its technology and product characteristics respectively.

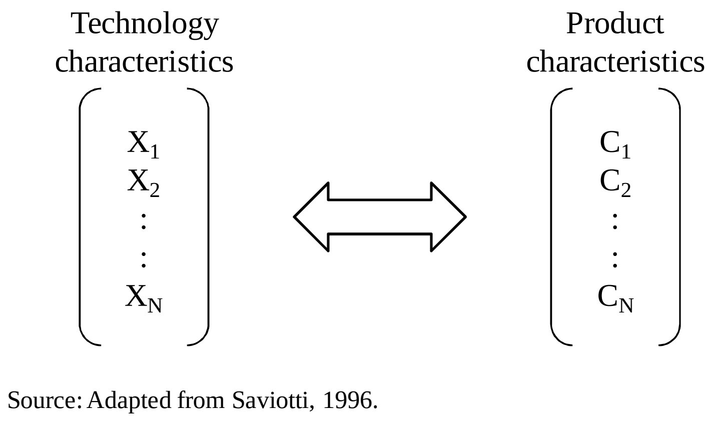

Technology characteristics
X_1
X_2
:
:
X_N
Product characteristics
C_1
C_2
:
:
C_N
Source: Adapted from Saviotti, 1996.

Figure 6.1 The technology and product characteristics of a product [Stieglitz, 2003]

The technological characteristics X_1, X_2, …, X_n consist of the knowledge base, which is necessary to transform inputs and labor into a particular product. The product displays a combination of characteristics C_1, C_2, …, C_n to meet the needs of customers or buyers. For the transformative product design paradigm, this research assumes that technologies also are represented in design knowledge consisting of functions, structures, and the associated behaviors since either technologies or products need their specifications at the conceptual level. In the following two sections, the different types of convergence will be further discussed and specified by using this framework for the transformative product design paradigm.

6.2.2. Technology Based Convergence Types

Stieglitz (2003) summarized the technology based convergence types with respect to how two industries that use different technologies to produce the products *A* and products *B* eventually become technologically convergent through either technology substitution or integration (Figure 6.2).

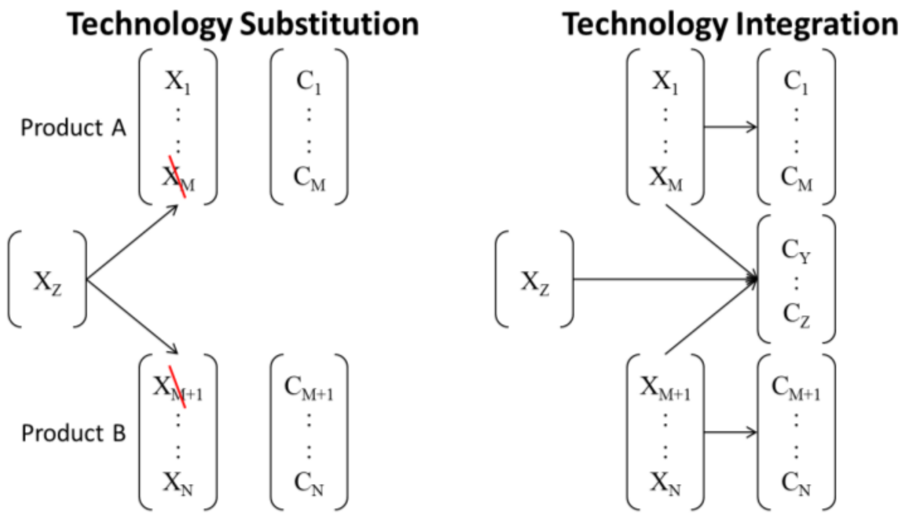


Figure 6.2 Types of technology-based convergence [Stieglitz, 2003]

Technology substitution

Stieglitz (2003) defines that technology substitution is triggered by the introduction of a new process technology *X_z*. (see the left-hand side of Figure 6.2). Products *A* and *B* in different industries represent previously existing products and they are not related before the introduction of a new *process technology X_Z*. The adoption of this technology in the production of two products is a technological substitution. Stieglitz highlights that, unlike all other forms of industry convergence, *process innovation* drives technology substitution. In addition, as shown in the figure, the product characteristics of two products are not affected.

Tushman and Anderson (1986) pointed that technology substitution can imply a technological discontinuity, which may lead incumbent firms to face challenges since a part of their capabilities could be obsolete. Another challenging issue addressed by Stieglitz (2003) is that the new technology does not build on the existing technological capabilities of incumbent firms. Rather entrants or firms from different industries are often a primary source of innovation.

Since a new technology can be applied in more industries, new firms emerge in a specialized industry upstream. Correspondingly, there is an increasing tendency among incumbent firms to outsource their technologies [Rosenberg, 1963; Pavitt, 1984; Rao, 1999]. These researchers obtained their market insights on industrial structures caused from the technology substitution. This research focuses on opportunities for new technologies. Thus the proposed methods in this research aim to answer the question of how to apply new technologies (or even new products in consideration of the fact that a technology is part of a product) to more products across any industry, for technology substitution. Consequently, the product development time would be decreased to introduce new applications of a new technology (or product) by identifying the right opportunities across products and industries.

An example of this type of convergence is found in Rosenberg's (1963) classic notion of technological convergence. Firearms (product A) and sewing machines (product B) became convergent technologically. As precision grinding mechanical technology (new technology X_Z) was introduced in industries, it replaced older technologies (X_M of product A, X_{M+1} of product B) of firearms and sewing machines. Even if this example illustrates the process technology, technology substitution can happen in general with new technologies, for example, new digital technologies.

Since the 1970s, many established industries, such as machine tools, telecommunications, consumer electronics, and even financial services, have broadly adopted digital technologies [Chandler, 2001; Gambardella and Torrisi, 1998; Helpman, 1999; Mowery, 1996]. New digital technologies have still replacing older, analogue technologies. In Section 6.3, a validation example for this type of convergence will be shown.

Technology integration

The introduction of a new technology X_Z may trigger technology integration like technology substitution. The difference between technology integration and technology substitution is the role of a new technology. Technology integration combines the new technology with existing technologies X_M and X_{M+1} to produce a new product C (see the right-hand side of Figure 6.2), while a new technology replaces existing technologies, for technology substitution. Stieglitz (2003) gives an example of this type of convergence, i.e., the handheld computer in the early 1990s. The new technology of handwriting recognition technology (X_Z) stimulated to develop the handheld computer, by combining various existing technologies (X_M, X_{M+1}) of telecommunications, computers, and consumer electronics.

Unlike technology substitution, technology integration may allow established firms to utilize, at least in part, their existing technological capabilities in combination with new technologies [Teece et al., 1994; Bower, 2002]. Established firms could take a competitive advantage over new entrants from their pre-entry experience with established technologies [Klepper and Simons, 2000; Helfat and Lieberman, 2002]. However, even with their pre-entry experience, it is not a noticeably obvious process to launch a new product into market. Manufacturers and users have to discover the basic characteristics establishing the product and how value can be determined from them [Suarez and Utterback, 1995; Adner and Levinthal, 2000]. Moreover, as Iansiti and West (1997, p.69) point out, ‘if a company selects technologies that don’t work well together, it can end up with a product that is hard to manufacture, is late getting to market, and does not fulfill its envisioned purpose.’ This is one of motivations of this research for the transformative product design paradigm by assisting the technology selection process in consideration of functional purposes of new products.

6.2.3. Product Based Convergence Types

Product-based convergence types are illustrated in the demand side with a market point of view [Stieglitz, 2003]. For these types of convergence, products begin to share similar or complementary product characteristics with the introduction of a new technology (see Figure 6.3).

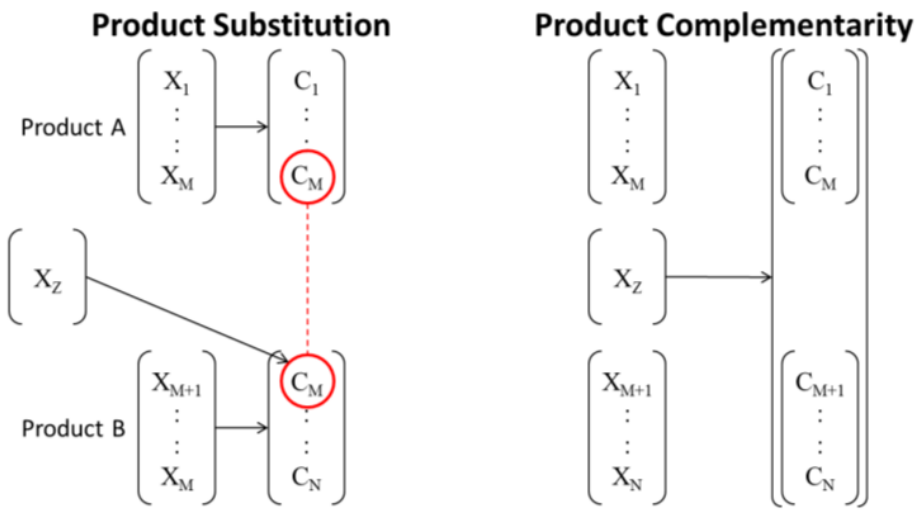


Figure 6.3 Types of product-based convergence [Steiglitz, 2003]

Product substitution

Product substitution leads to a (greater) substitutability between formerly unrelated products as they (increasingly) come to share the same product characteristics [Stieglitz, 2003]. The left-hand side of Figure 6.3 illustrates product *B* includes a new characteristic C_M, replacing an existing characteristic C_{M+1}, according to the introduction of a new technology X_Z that enables the substitution in product characteristics.

When one product becomes a closer substitute for the other product, two products will share more product characteristics. The dynamics of product substitution leads the increase of cross-disciplinary technology advances in which firms from two or more products expand their existing products by incorporating technologies from the other products. Accordingly, a product not only has to assimilate the new technology X_Z, but also adopts existing technologies of the other product. Stieglitz (2003) addressed the possibility that product substitution may be accompanied by technology-based convergence as the technological features of both products converge. He also expected that the extreme case prospects that products merge into one larger product with similar technologies and product characteristics. Nevertheless, this research differentiates the two types of convergence as is since technology integration integrates and combines existing product characteristics from different products, while product substitution replaces existing product characteristic(s) of one product with other product characteristics of the other products. In the transformative product design paradigm, the design transformation process detects the highly possible direction toward new applications from existing technologies or products with respect to the different convergent types.

The learning process of product substitution is slightly different from the one of technology integration. It departs from existing dominant products and firms would like to build on and extend their existing products to develop convergent products incrementally. This process may cause uncertainty and ambiguity for established products [Greenstein and Khanna, 1997]. Therefore, the design transformation process has to be able to reduce the uncertainty and ambiguity from the process by discovering what bundles of product characteristics that customers want and which new capabilities to build.

A traditional example of convergence through product substitution was the dynamic relationship between mainframes and minicomputers in the 1970s and early 1980s [Bresnahan and Greenstein, 1999]. A recent example can be the relationship between personal computers (product *A*) and smartphones (product *B*) such as iPhone and Android phones. Personal computers were used for general computing tasks, while the first generation smartphones before iPhone and Android phones were specialized for business tasks such as digital organizer, email access, and calculator. Accordingly, they differed considerably in terms of product characteristics and were considered distinct products. Technological innovations in mobile app (X_Z), which is a software application running on smartphones, augmented user interfaces for smartphones, which allowed users to access more Internet-based services and to enjoy more complex, general computing tasks (C_M) that were used to be available on personal computers. The result affects the tablet computers also and leads increased competition between personal computers and smartphones/tablet computers.

Product complementarity

Stieglitz (2003) defines industry convergence through product complementarity as the following explanation: existing products, which were once used independently, become complementary from a user perspective. This type of product-associated convergence may be caused by technological development (X_Z), which provides the opportunity to use the two existing products together (see the right-hand side of Figure 6.3). To achieve successful product complementarity, a situation is necessary, in which two (or more) products deliver a higher value if jointly used.

In the perspective of manufacturing products *A* or *B*, the new technology (X_Z) is not actually needed. Instead, the new technology plays a role of the ‘glue’ to connect the two products. Some kind of interface or standard is required to facilitate the complementary use of the products [Katz, 1996; Yoffie, 1997].

Unlike product substitution, product complementarity does not lead to technology-based convergence; however, this is hardly to say that product complementarity impacts nothing on the existing products. Product complementarity triggers often product innovations. Nevertheless, Stieglitz (2003) insisted that product complementarity has a major impact on industry dynamics and corporate strategies in the established products. Therefore, this type of convergence is not appropriated with the transformative product design paradigm, which pursues a new way of identification of new applications with cross-disciplinary technology advance. Consequently, this research ignores this type for the design transformation process.

6.2.4. Summary

This section studies the definition of convergence. Stieglitz (2003) determines the boundary of an industry by grouping products with similar characteristics, which is related to *substitutability*. In addition, products can have characteristic of complementarity when customers increase their utilities of a product by jointly using products. He also determines types of converges with respect to the demand side and the supply side. The demand side is related to product-based convergence while another convergence lies on the basis of technology, which is the supply side. According to his taxonomy of convergence, there are four types of convergence: (1) technology substitution, (2) technology integration, (3) product substitution, and (4) product complementarity. When introducing a new technology (i.e., a new cross-disciplinary technology advance in the transformative product design paradigm), each convergence type has unique characteristics, depending on how product characteristics are affected by the new technology. However, the last convergence type, product complementarity is established on the marketing perspective, not on the product perspective. Therefore, this research excludes product complementarity for the validation process.

6.3. Validation

6.3.1. Digital Revolution

Today, market demands faster, smaller, cheaper, and more reliable products, especially in the hardware, telecommunication, and semi-conductor industries. Technical challenges in developing such products are not a single problem, but multi-disciplinary problems. Consequently, consumers have been gradually seeking more convergent products that combine additional features from one product to the other device so that manufacturers and service providers have tried to take the convergence opportunities in order to increase the market competitiveness. Digital revolution fulfills such market demands and lies at the heart of convergence.

Mobile devices are one of the most actively converging products by digital revolution. The daily carrying characteristic of mobile devices makes them a test bed for the convergence by integrating various features with advanced technologies. Key technologies of digital revolution come from the telecommunication industry, which transforms the analog to the digital via Internet protocols. Consumers are now playing music and games or using diverse online services via their mobile phones. Furthermore, smartphones have not only integrated PDA features into cellular phones, but also employed more complex, general computing tasks used to be offered by personal computers.

According to the analysis by Gartner in Apr 2011, 296 million mobile phones have been sold worldwide and the growth still keeps increasing and probably reaches to 1.1 billion units being sold in 2015 (see Table 6.2). Recently, as smartphones have been rapidly increased, it is expected to see the great escalation from 58.6% in 2010 to 96.6% in 2015 in terms of the market share by operating systems of Google’s Android, Apple’s iOS, Microsoft’s Windows Phone 7, and RIM's BalckBerry OS. As represented in the following table, mobile devices are good examples of the increase of demands of convergence in market as new digital technologies have been introduced. Thus, this section employs digital technologies and mobile devices as examples to validate the proposed methods and the prototype system according to the types of convergence presented in the previous section.

Table 6.2 Worldwide mobile phone sales and shares by operating systems (Unit: Thousand)

OS	**2007**	**2008**	**2009**	**2010**	**2011**	**2012**	**2015**
Symbian	77,684	72,934	80,878	111,577	89,930	32,666	661
Share (%)	63.5	52.4	46.9	37.6	19.2	5.2	0.1
Android			6,798	67,225	179,873	310,088	539,318
Share (%)			3.9	22.7	38.5	49.2	48.8
RIM	11,768	23,149	34,347	47,452	62,600	79,335	122,864
Share (%)	9.6	16.6	19.9	16.0	13.4	12.6	11.1
iOS	3,303	11,418	24,890	46,598	90,560	118,848	189,924
Share (%)	2.7	8.2	14.4	15.7	19.4	18.9	17.2
Microsoft	14,698	16,498	15,031	12,378	26,346	68,156	215,998
Share (%)	12.0	11.8	8.7	4.2	5.6	10.8	19.5
Other OSs	14,863	15,290	10,432	11,417.40	18,392.30	21,383.70	36,133.90
Share (%)	12.1	11.0	6.1	3.8	3.9	3.4	3.3
Total	**122,316**	**139,288**	**172,376**	**296,647**	**467,701**	**630,476**	**1,104,898**

Source: Gartner (Apr 2011)

6.3.2. Technology Substitution

Stieglitz (2003) provides examples for technology substitution. A video recorder (product *A*) and a film camera (product *B*) had not shared any relationship between the two. As digital technologies (a new technology X_Z) emerged rapidly in various industries, the two products have adopted a new technology trend to digitize the analog products. Video recorders convert the analog sound and image into the digital sound and image in a sequence of discrete values. Cameras transmit the image of subjects on the diaphragm into an image sensor to interpret. After introducing digital technologies to two products, there has been no change of two products in terms of their product characteristics.

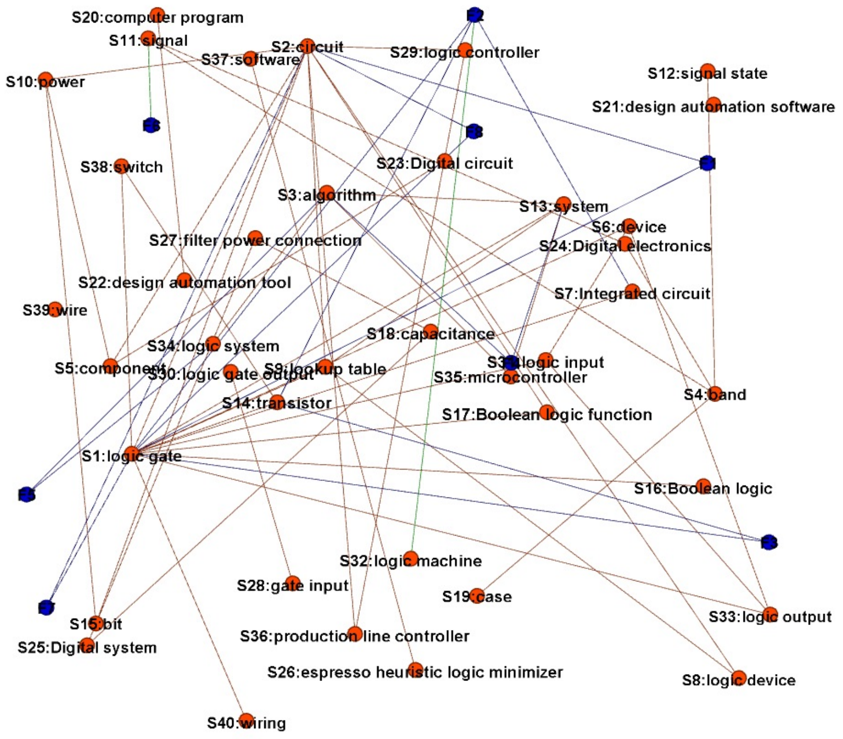


Figure 6.4 Interaction network of digital technologies

First of all, according to the proposed methods, the prototype system is able to translate the functional requirements of digital technologies, even if they are hardly to say products. This is possible if the functional requirement contains teleological knowledge and physical knowledge. Thus, the identification method for design knowledge extraction can be also extended to knowledge extraction of technologies. This is a significant contribution of the research method to extract existing knowledge of either technologies or products. In this research, technologies can be also interpreted in the *interaction network* world of design knowledge. Figure 6.4 is an interaction network of digital technologies, which is generated after converting and tokenizing the functional requirement analysis of digital technologies.

Figure 6.5 shows an interaction network of a video recorder and Figure 6.6 illustrates an interaction network after the design transformation process with the digital technologies. For this type of conversion, the design transformation process simply detects the similar characteristics between design technologies and a video recorder, which is analog signal that can be substituted by digital technologies. Analog signals of a video recorder and a film camera are results of two products, while they become input sources for digital technologies to digitize. As a result, the transformed interaction network presents the combination of two interaction networks.

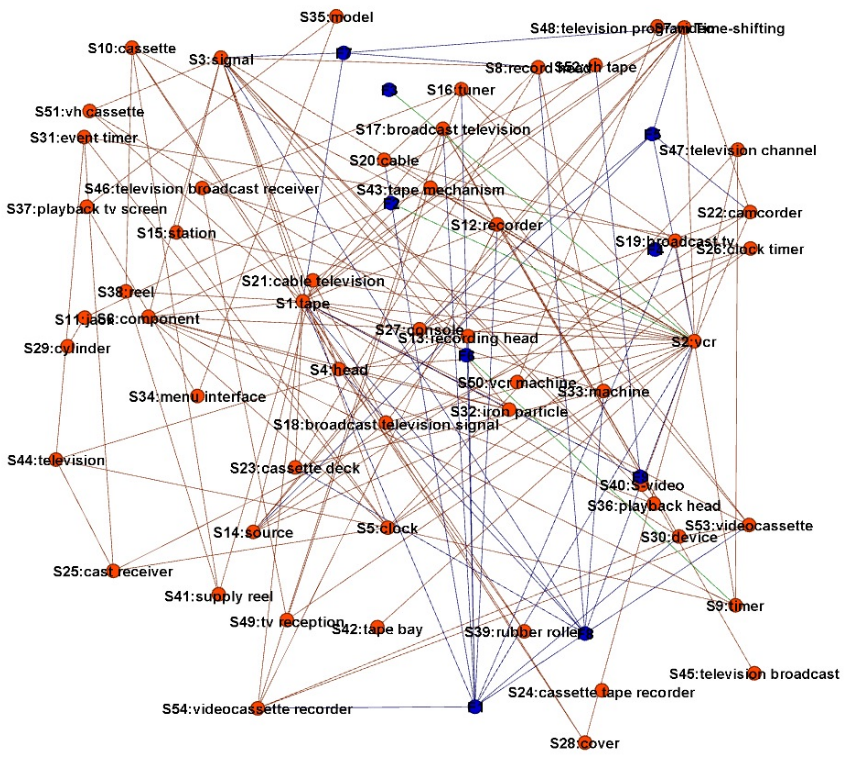


Figure 6.5 Interaction network of a video recorder

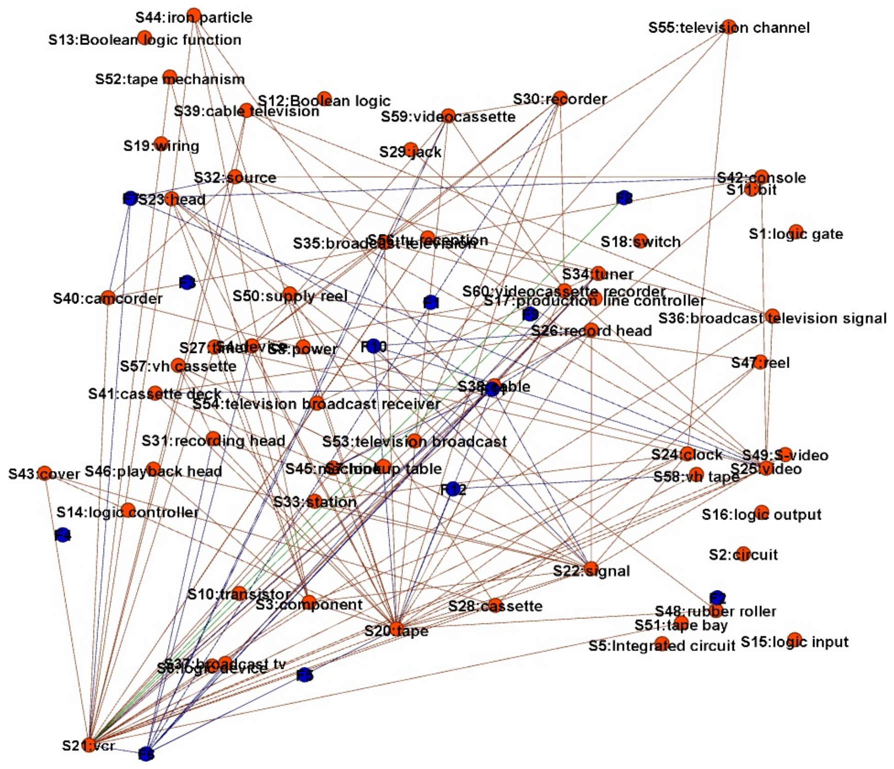
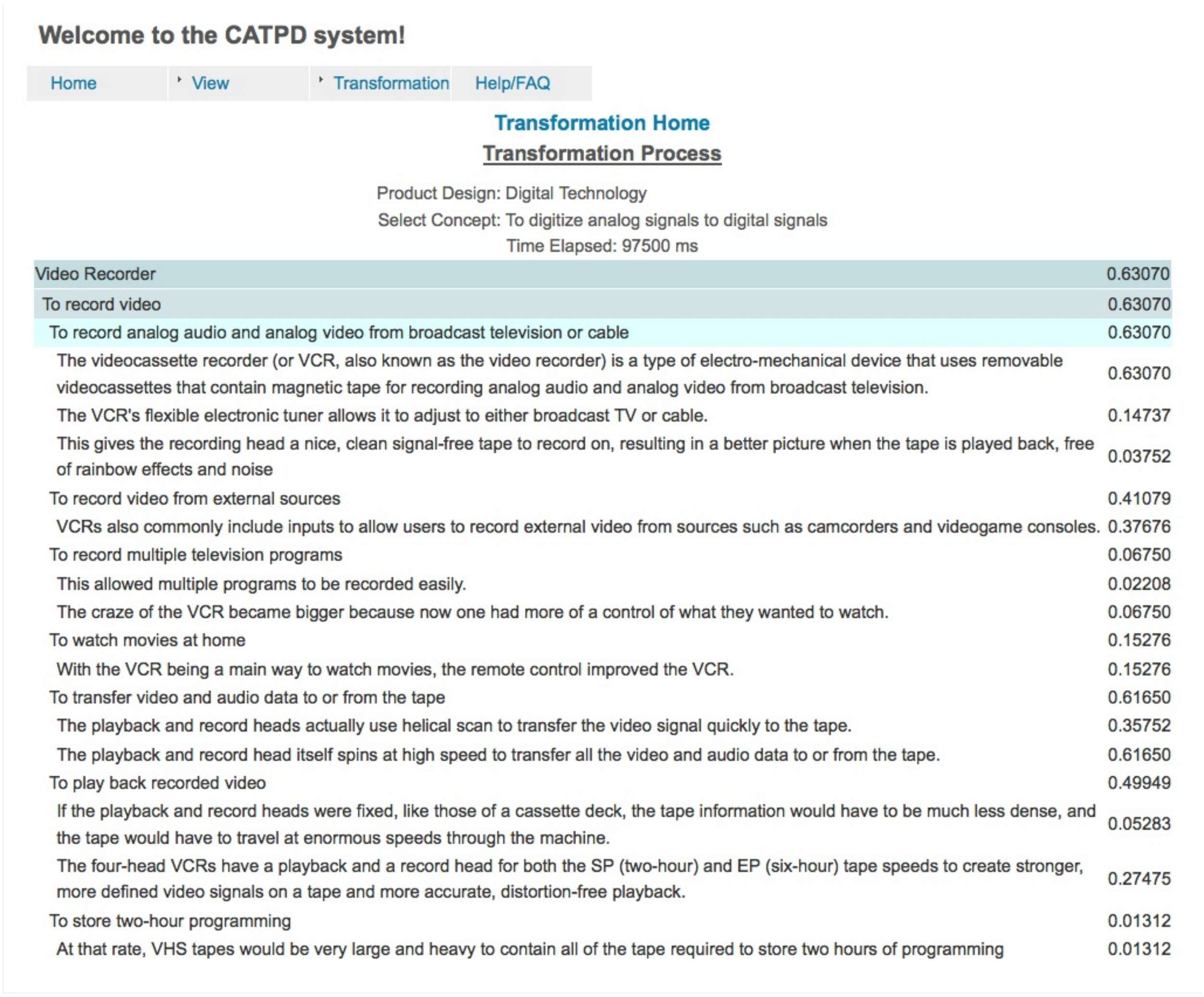


Figure 6.6 Interaction network of a video recorder with digital technologies

A new technology substitutes a current technology of an existing product and may impact on products to substitute current technologies. Based on the generated transformative interaction network (i.e., design alternative), designers could configure which structures (components) are required to substitute with digital technologies for the type of technology substitution conversion, since existing and new structures remain connected in the transformed interaction network in Figure 6.6. The prototype system has exhibited the ability to guide the direction of new applications with a new technology (i.e., digital technologies in this example) by identifying candidate products (i.e., a video recorder and a film camera) for technology substitution (see Figure 6.7).



Transformation Process

Product Design: Digital Technology
Select Concept: To digitize analog signals to digital signals
Time Elapsed: 97500 ms

Video Recorder	0.63070
To record video	0.63070
To record analog audio and analog video from broadcast television or cable	0.63070
The videocassette recorder (or VCR, also known as the video recorder) is a type of electro-mechanical device that uses removable videocassettes that contain magnetic tape for recording analog audio and analog video from broadcast television.	0.63070
The VCR's flexible electronic tuner allows it to adjust to either broadcast TV or cable.	0.14737
This gives the recording head a nice, clean signal-free tape to record on, resulting in a better picture when the tape is played back, free of rainbow effects and noise	0.03752
To record video from external sources	0.41079
VCRs also commonly include inputs to allow users to record external video from sources such as camcorders and videogame consoles.	0.37676
To record multiple television programs	0.06750
This allowed multiple programs to be recorded easily.	0.02208
The craze of the VCR became bigger because now one had more of a control of what they wanted to watch.	0.06750
To watch movies at home	0.15276
With the VCR being a main way to watch movies, the remote control improved the VCR.	0.15276
To transfer video and audio data to or from the tape	0.61650
The playback and record heads actually use helical scan to transfer the video signal quickly to the tape.	0.35752
The playback and record head itself spins at high speed to transfer all the video and audio data to or from the tape.	0.61650
To play back recorded video	0.49949
If the playback and record heads were fixed, like those of a cassette deck, the tape information would have to be much less dense, and the tape would have to travel at enormous speeds through the machine.	0.05283
The four-head VCRs have a playback and a record head for both the SP (two-hour) and EP (six-hour) tape speeds to create stronger, more defined video signals on a tape and more accurate, distortion-free playback.	0.27475
To store two-hour programming	0.01312
At that rate, VHS tapes would be very large and heavy to contain all of the tape required to store two hours of programming	0.01312

Figure 6.7 The degree of transformability for digital technologies

6.3.3. Technology Integration

Technology integration illustrates the convergence of generating new product characteristics by integrating a new technology. The handwriting recognition technology is one of examples of technology integration. According to the introduction of the technology, product characteristics of mobile phones, computers, and/or consumer electronics like a calculator converged into a new product. To show this convergence type, the design knowledge acquisition of the technology is a fundamental issue for the later design transformation process since all the research methods in this research are strictly dependent on the interaction network. As illustrated in the previous section, the handwriting recognition technology is successfully de-documented and visualized in the interaction network model (see Figure 6.8).

In this validation section, a handwriting recognition technology is used as a new technology (X_Z) to identify existing products to transform. The design transformation process, with this given technology X_Z, determines the degrees of transformability of products in the design repository. In this example, a personal computer and a touchscreen are selected for the transformation, which have been developed in different products (see Figure 6.9 and Figure 6.10).

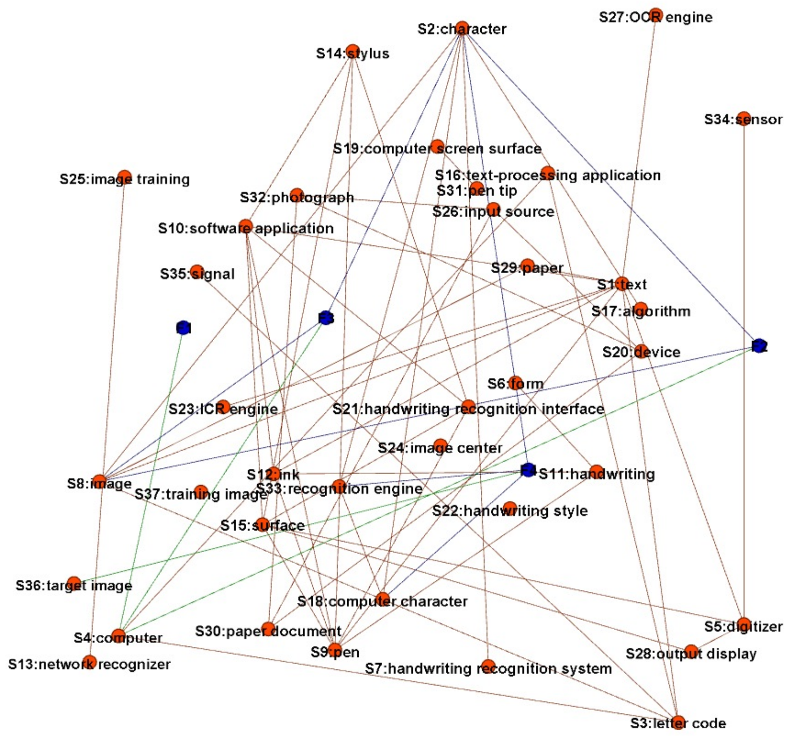


Figure 6.8 Interaction network of handwriting recognition technology

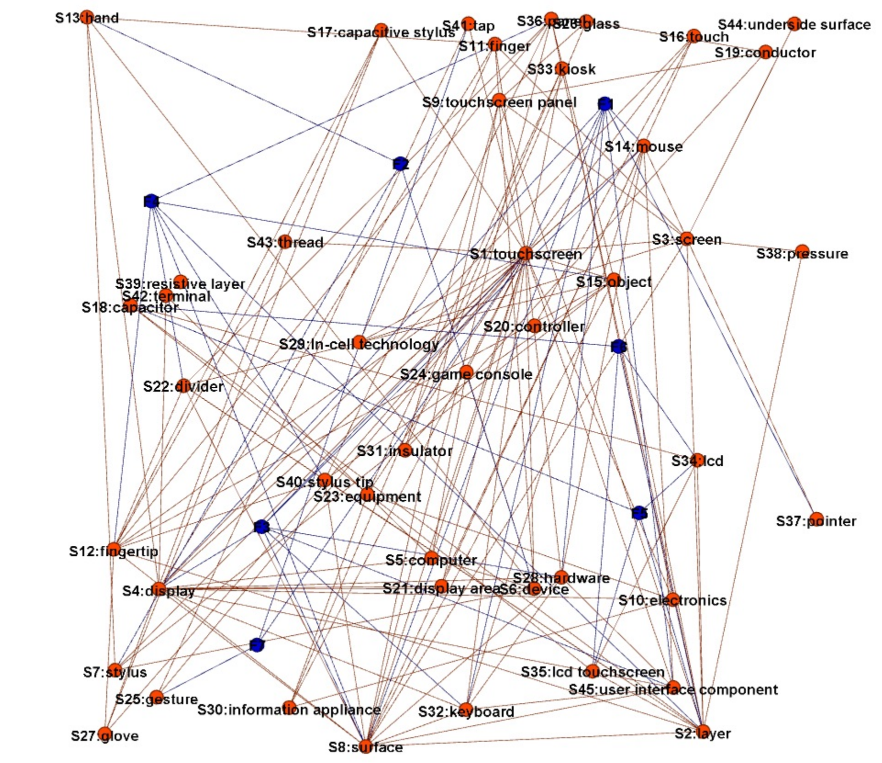


Figure 6.9 Interaction network of touchscreens

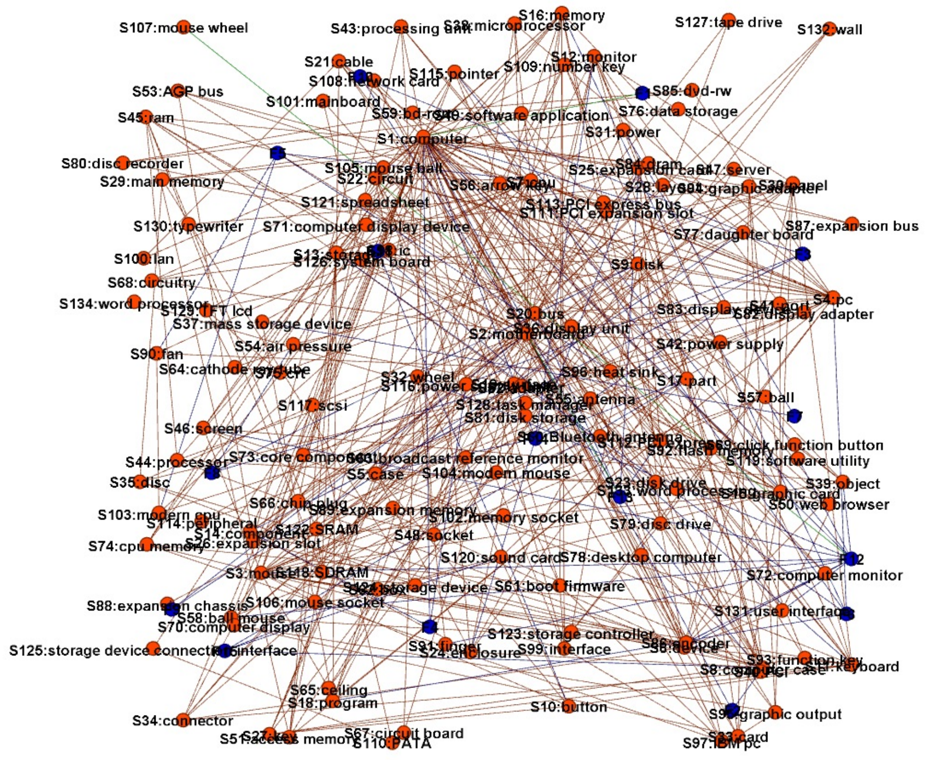
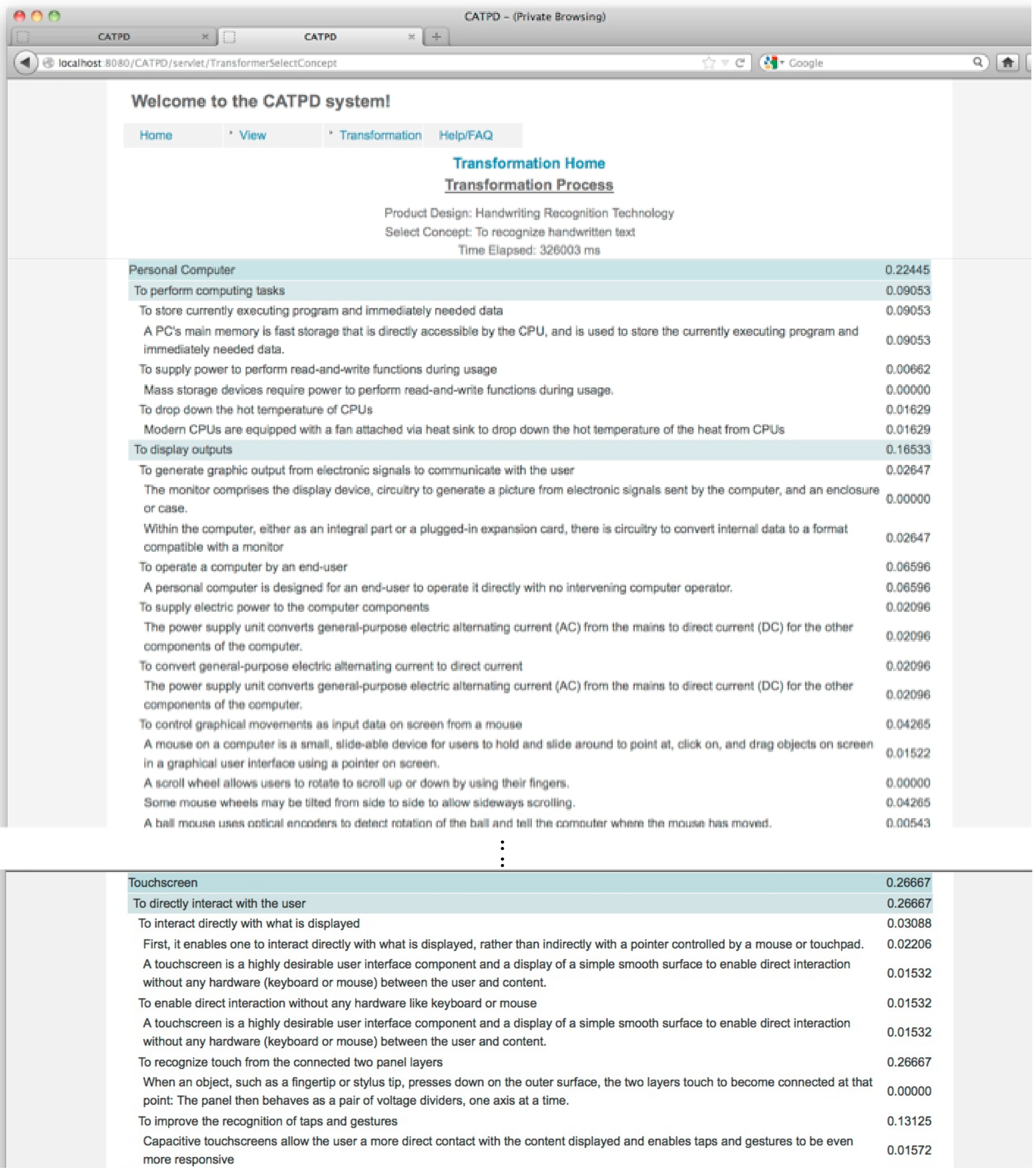


Figure 6.10 Interaction network of a personal computer

Before introducing the handwriting recognition technology to markets, two different products have been targeting different users. Personal computers have had individual end-users while touchscreens have been applied to the automated machines in business such as ATMs and self-checkout machines.


CATPD – (Private Browsing)
CATPD
CATPD
localhost:8080/CATPD/servlet/TransformerSelectConcept
Google

Welcome to the CATPD system!

Home | View | Transformation | Help/FAQ

Transformation Home

Transformation Process

Product Design: Handwriting Recognition Technology
Select Concept: To recognize handwritten text
Time Elapsed: 326003 ms

Personal Computer	0.22445
To perform computing tasks	0.09053
To store currently executing program and immediately needed data	0.09053
A PC's main memory is fast storage that is directly accessible by the CPU, and is used to store the currently executing program and immediately needed data.	0.09053
To supply power to perform read-and-write functions during usage	0.00662
Mass storage devices require power to perform read-and-write functions during usage.	0.00000
To drop down the hot temperature of CPUs	0.01629
Modern CPUs are equipped with a fan attached via heat sink to drop down the hot temperature of the heat from CPUs	0.01629
To display outputs	0.16533
To generate graphic output from electronic signals to communicate with the user	0.02647
The monitor comprises the display device, circuitry to generate a picture from electronic signals sent by the computer, and an enclosure or case.	0.00000
Within the computer, either as an integral part or a plugged-in expansion card, there is circuitry to convert internal data to a format compatible with a monitor	0.02647
To operate a computer by an end-user	0.06596
A personal computer is designed for an end-user to operate it directly with no intervening computer operator.	0.06596
To supply electric power to the computer components	0.02096
The power supply unit converts general-purpose electric alternating current (AC) from the mains to direct current (DC) for the other components of the computer.	0.02096
To convert general-purpose electric alternating current to direct current	0.02096
The power supply unit converts general-purpose electric alternating current (AC) from the mains to direct current (DC) for the other components of the computer.	0.02096
To control graphical movements as input data on screen from a mouse	0.04265
A mouse on a computer is a small, slide-able device for users to hold and slide around to point at, click on, and drag objects on screen in a graphical user interface using a pointer on screen.	0.01522
A scroll wheel allows users to rotate to scroll up or down by using their fingers.	0.00000
Some mouse wheels may be tilted from side to side to allow sideways scrolling.	0.04265
A ball mouse uses optical encoders to detect rotation of the ball and tell the computer where the mouse has moved.	0.00543

⋮

Touchscreen	0.26667
To directly interact with the user	0.26667
To interact directly with what is displayed	0.03088
First, it enables one to interact directly with what is displayed, rather than indirectly with a pointer controlled by a mouse or touchpad.	0.02206
A touchscreen is a highly desirable user interface component and a display of a simple smooth surface to enable direct interaction without any hardware (keyboard or mouse) between the user and content.	0.01532
To enable direct interaction without any hardware like keyboard or mouse	0.01532
A touchscreen is a highly desirable user interface component and a display of a simple smooth surface to enable direct interaction without any hardware (keyboard or mouse) between the user and content.	0.01532
To recognize touch from the connected two panel layers	0.26667
When an object, such as a fingertip or stylus tip, presses down on the outer surface, the two layers touch to become connected at that point: The panel then behaves as a pair of voltage dividers, one axis at a time.	0.00000
To improve the recognition of taps and gestures	0.13125
Capacitive touchscreens allow the user a more direct contact with the content displayed and enables taps and gestures to be even more responsive	0.01572

Figure 6.11 The degrees of transformability for the handwriting recognition technology

According to the degrees of transformability with respect to the handwriting recognition technology in Figure 6.11, there might be two existing products: personal computers and touchscreens. When a designer selects these two products for the design transformation process, the remaining transformation process with the handwriting recognition technology continues to combine functions from the two products altogether. The result of this process is presented in Figure 6.12.

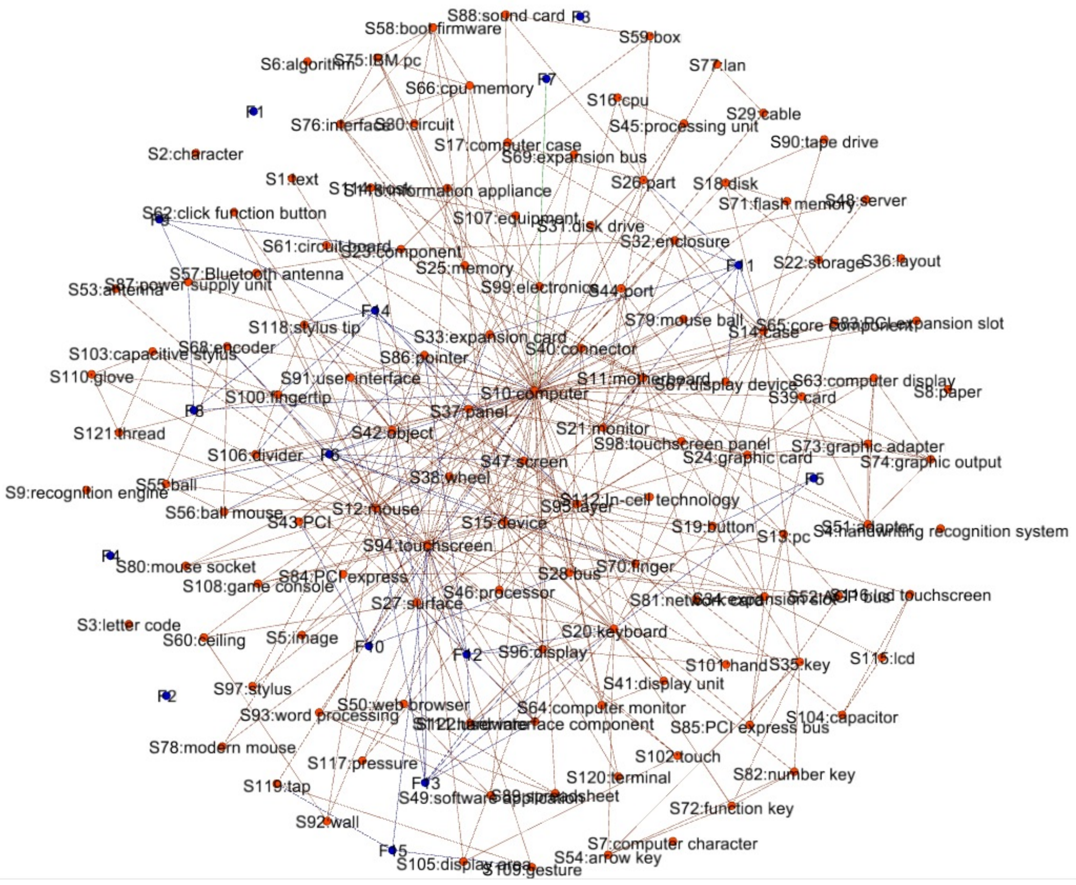
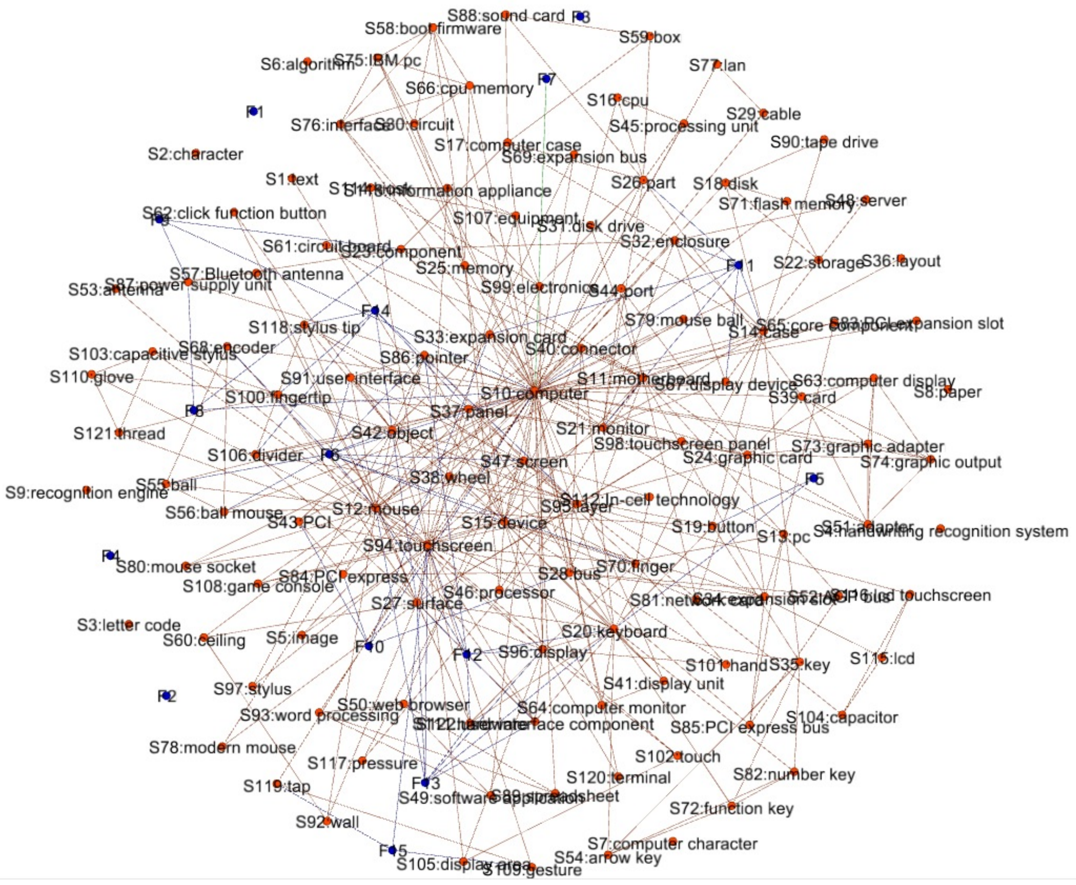


Figure 6.12 Interaction network of a handwriting recognition technology enabled computer



According to the type of technology integration convergence, the difference from the previous convergence type is to apply a new technology to multiple products in order to combine altogether or to develop a new product in consideration of a new technology. When relationships between structures and functions in a newly generated interaction network are disconnected, it means that a new transformative design require a new product by combining those different products and technologies. The previous convergence simply applies a new technology into different products independently so that there is no apparent integration between existing products. The prototype system shows a possible direction that a new technology could be a catalyst to integrate different products. Thus, the convergence type of technology integration could be also supported by the prototype system with the proposed methods.

6.3.4. Product Substitution

To validate the last type of convergence, product substitution, this research explores technological innovations in mobile apps as the new technology that facilitates a new, innovative product design by interweaving multiple product characteristics from different products. Mobile apps are computer software application running on smartphones. Introducing mobile apps that allow third-party developers has made smartphones to perform better integration with their hardware than they used to do.

If the prototype system is able to provide such an innovative product design environment that can reduce human efforts to generate innovative product design alternatives, this research successfully achieves the goal to differentiate the three types of convergence and to generate the corresponding design transformation process. Before continuing the process, another challenging issue is of how to interpret such software design into the interaction network. Like previous two examples for the technology-based convergence types, the identification method also presents a fruitful result for the de-documentation process of software products (see Figure 6.13).

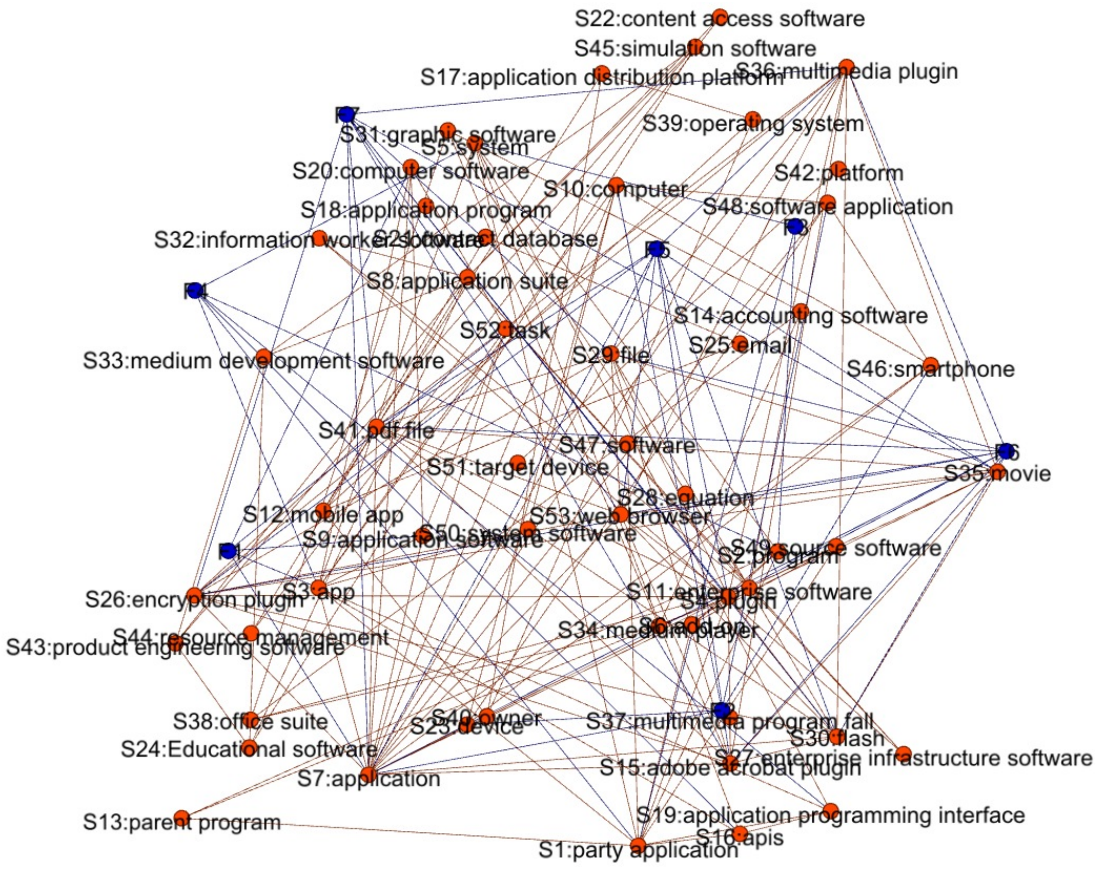


Figure 6.13 Interaction network of mobile apps

The product substitution convergence type results in the integration of existing product characteristics due to the introduction of a new technology. In this example, this research considers a barcode scanner, which has never been considered as an appropriate feature for smartphones. An interaction network of smartphones is exemplified in Figure 6.14.

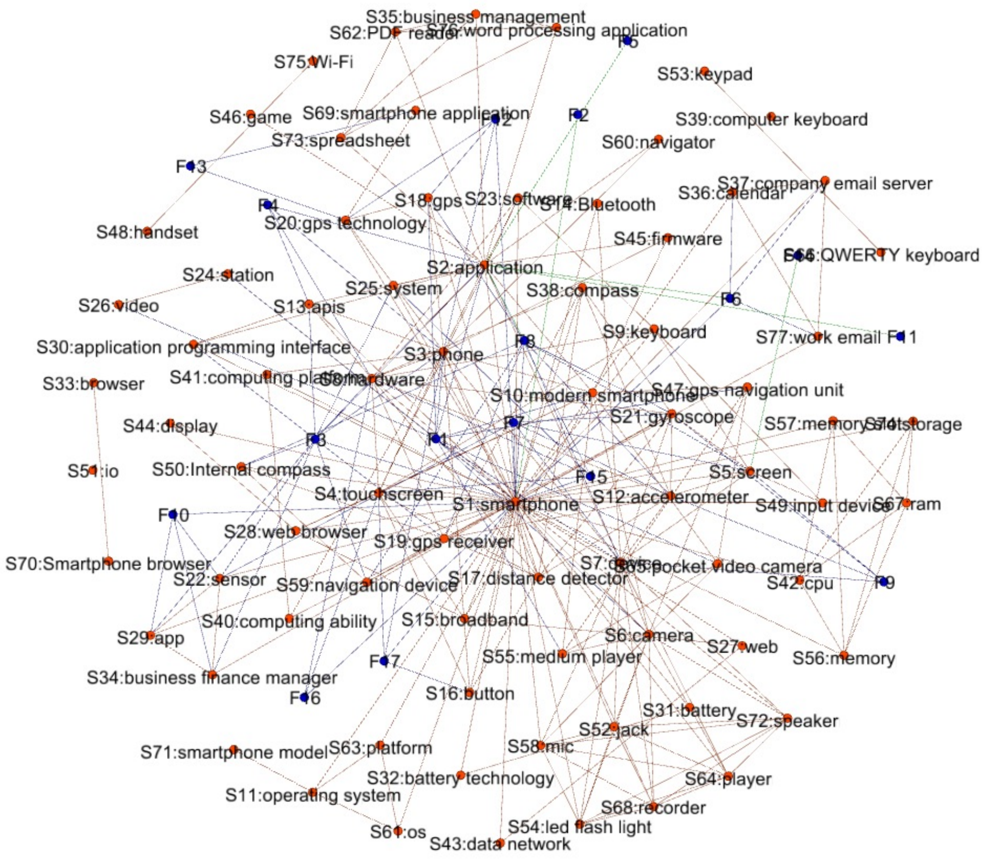
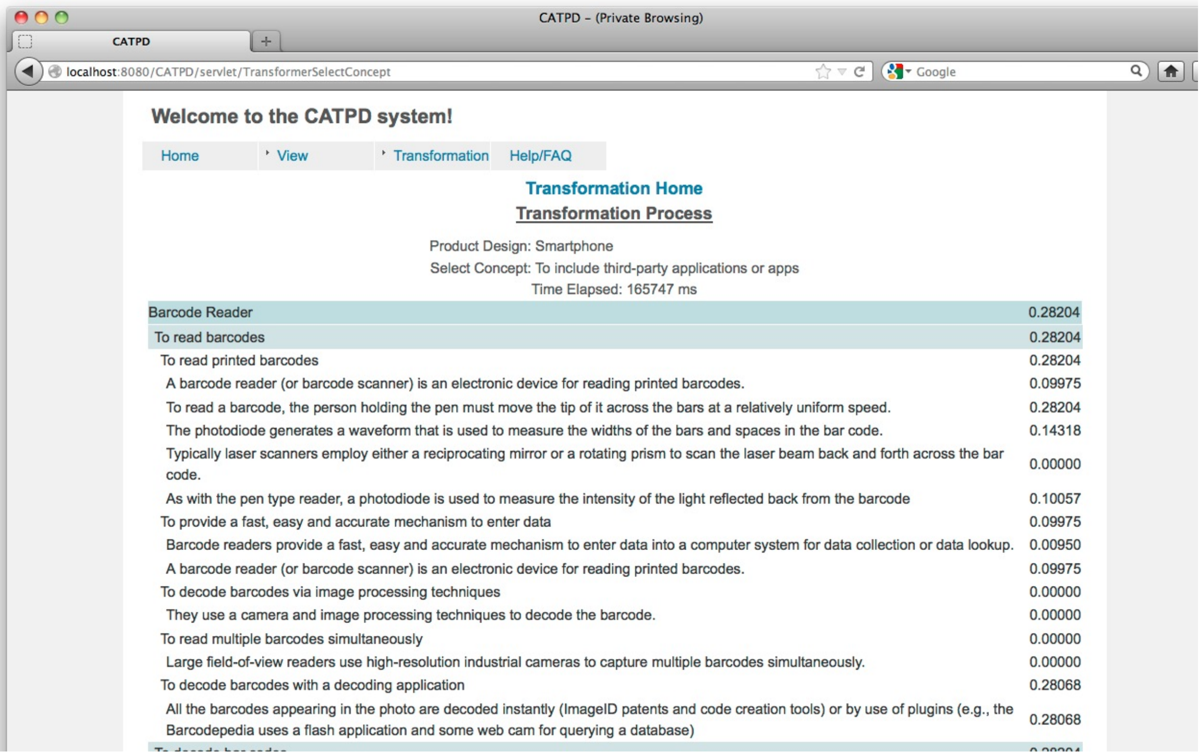


Figure 6.14 Interaction network of a smartphone

The meaningful difference of this type of convergence is that the design transformation process combines two existing products with the new technology. Hence, in a functional design point of view, this convergence type is similar to technology integration. However, this convergence type may reduce the number of structures since the physical parts for one product (in this example, barcode reader) present in the other product (in this case, smartphones). Therefore, while the technology integration may result in the disconnection between structures in the network for a newly transformed product design, the product substitution pays attention to the duplicated structures from the transformed product design.


CATPD – (Private Browsing)
CATPD
localhost:8080/CATPD/servlet/TransformerSelectConcept
Google
Home
View
Transformation
Help/FAQ

Welcome to the CATPD system!

Transformation Home

Transformation Process

Product Design: Smartphone
Select Concept: To include third-party applications or apps
Time Elapsed: 165747 ms

Barcode Reader	0.28204
To read barcodes	0.28204
To read printed barcodes	0.28204
A barcode reader (or barcode scanner) is an electronic device for reading printed barcodes.	0.09975
To read a barcode, the person holding the pen must move the tip of it across the bars at a relatively uniform speed.	0.28204
The photodiode generates a waveform that is used to measure the widths of the bars and spaces in the bar code.	0.14318
Typically laser scanners employ either a reciprocating mirror or a rotating prism to scan the laser beam back and forth across the bar code.	0.00000
As with the pen type reader, a photodiode is used to measure the intensity of the light reflected back from the barcode	0.10057
To provide a fast, easy and accurate mechanism to enter data	0.09975
Barcode readers provide a fast, easy and accurate mechanism to enter data into a computer system for data collection or data lookup.	0.00950
A barcode reader (or barcode scanner) is an electronic device for reading printed barcodes.	0.09975
To decode barcodes via image processing techniques	0.00000
They use a camera and image processing techniques to decode the barcode.	0.00000
To read multiple barcodes simultaneously	0.00000
Large field-of-view readers use high-resolution industrial cameras to capture multiple barcodes simultaneously.	0.00000
To decode barcodes with a decoding application	0.28068
All the barcodes appearing in the photo are decoded instantly (ImageID patents and code creation tools) or by use of plugins (e.g., the Barcodepedia uses a flash application and some web cam for querying a database)	0.28068

Figure 6.15 The degree of transformability for smartphones with mobile apps

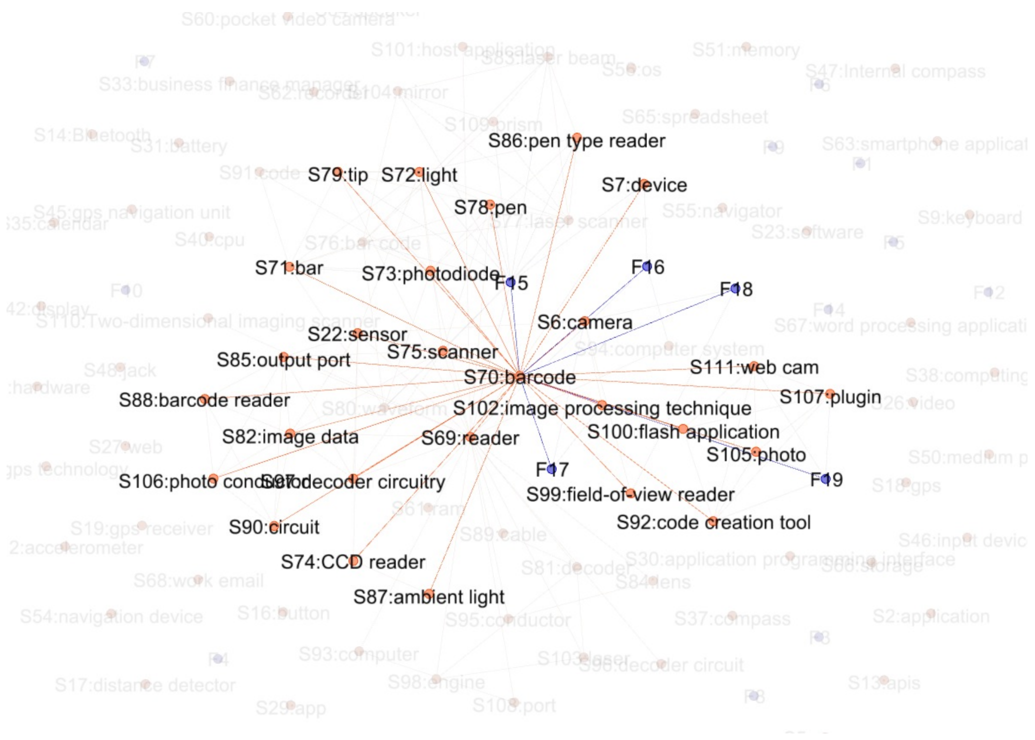


Figure 6.16 Interaction network of a transformed smartphone as a barcode reader

6.4. Conclusion

6.4.1. Summary

This chapter summarizes the types of industry convergence studied by Stieglitz (2003) in a managerial view. He classified four types of convergence: two from the technology-based approach and others from the product-based approach. In addition to the categorization, he pays attention to the characteristics of products, which are substitutability and complementarity. The former can be determined by sharable similar product characteristics of different products, while the following is related to the utility of customers, who use products. Due to this customers' point of view, out of the four categorization of convergence, the last type is excluded from the validation process. The product complementarity convergence type is defined by the perspective of customers in a marketing view. Hence, the first three types of convergence are adopted to validate the research methods, which are *technology substitution*, *technology integration*, *product substitution* and the proposed methods and the prototype system supports a transformative product design environment that covers these three types of convergence.

6.4.2. Contribution

Throughout the validation process, a significant finding is that, to fully support the convergence types, the transformative product design paradigm has to reflect technologies and software that might be slightly different in terms of knowledge representation. However, this challenging issue can be resolved throughout the validation process if functional requirements for technologies or software products contain teleological knowledge and physical knowledge. Since the proposed interaction network is capable to represent two perspectives of knowledge, appropriate functional descriptions of products, software products, and even technologies can be translated into the interaction network. By using this translated interaction network based knowledge, the remaining transformative product design processes could continue.

Another contribution is to determine the differences among the convergence types in the transformative product design's perspective. The first technology substitution convergence type is related to the design transformation that transforms products with a new technology and not affects other products accordingly. The second type of technology integration requires considering the design transformation to combine existing products with a new technology. Hence, there is a possibility that a transformed design may have disconnected relationships between structures. The last convergence type of product substitution requires combining functional characteristics of product design between products due to a new technology. So this type of convergence may have duplicated structures from the combination of existing products. Therefore, in the future research, it is necessary to support these differences in the design transformation process to facilitate the prototypes system for the proposed design paradigm.

CHAPTER 7
SYSTEM DEVELOPMENT OF A COMPUTER-AIDED DESIGN SUPPORTING SYSTEM

7.1. System Architecture

This research proposes several methods to realize the transformative product design by analyzing the existing design knowledge for a new direction of products. Figure 7.1 illustrates the overall system architecture to implement the proposed algorithms. A designer is able to upload any functional requirement document written in an informal free text document in English. It is possible for a designer to determine functions and structures with a minimum intervention. In order to implement this process, there are three agents: (1) FR reader, which helps reading a document, (2) Text Splitter, which decomposes document into word-level linguistic components, and (3) POS Helper, which assists to analyze the *part-of-speech* of any word in a sentence. Once a designer is done with the interaction network generation from the functional description, he/she is ready to continue the transformation process.

The transformation process starts from the request of concept detection to identify concept functions as candidates for the transformation process. The Concept Finder agent store detected concept functions into the design knowledge base. According to the foundation of the concept functions, the Transformer agent will generate a design alternative, which is an interaction network, for a new product by interweaving multiple concepts across domains in order to obtain an insight for a new product direction.

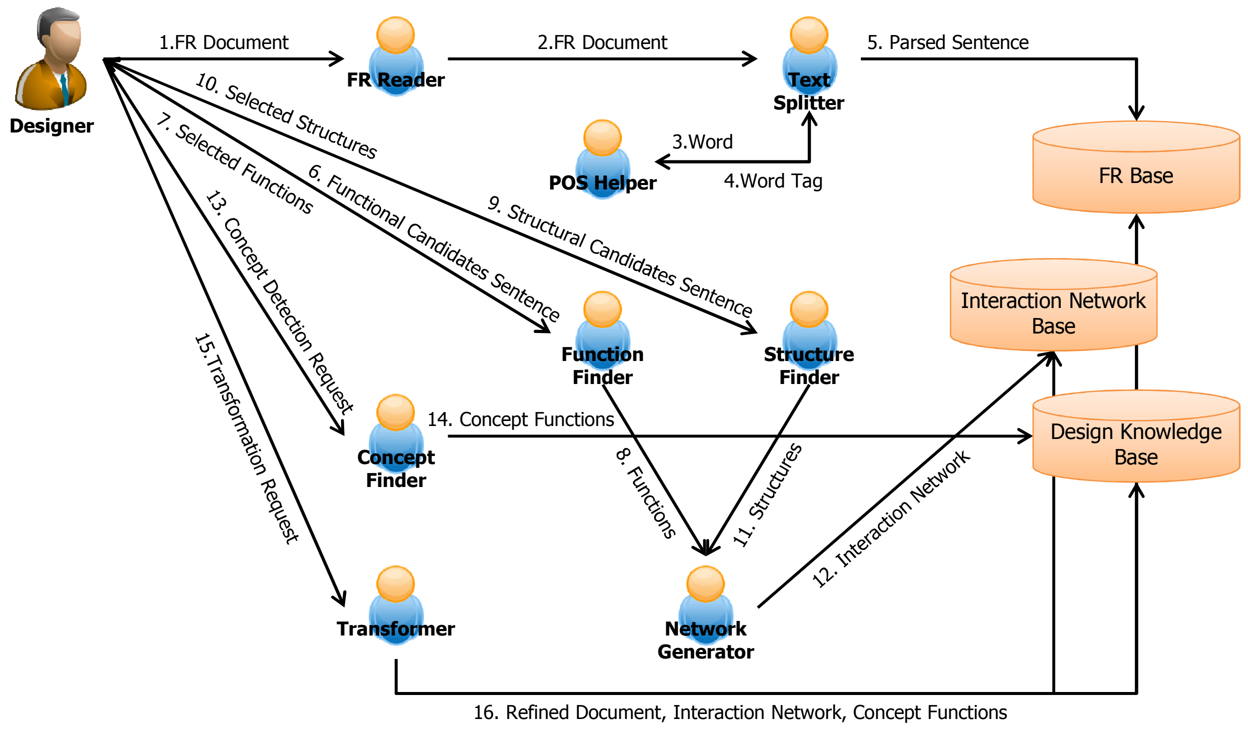
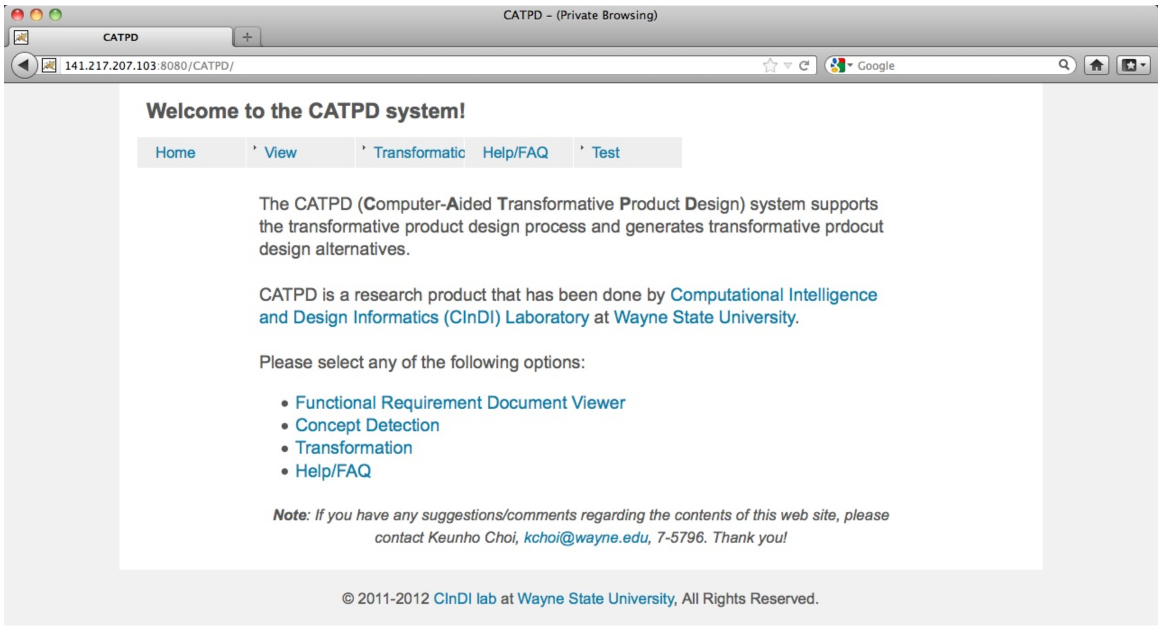


Figure 7.1 System architecture of computer-aided design supporting system



Welcome to the CATPD system!

The CATPD (**C**omputer-**A**ided **T**ransformative **P**roduct **D**esign) system supports the transformative product design process and generates transformative prdocut design alternatives.

CATPD is a research product that has been done by Computational Intelligence and Design Informatics (CInDI) Laboratory at Wayne State University.

Please select any of the following options:

- Functional Requirement Document Viewer
- Concept Detection
- Transformation
- Help/FAQ

Note*: If you have any suggestions/comments regarding the contents of this web site, please contact Keunho Choi, kchoi@wayne.edu, 7-5796. Thank you!*

© 2011-2012 CInDI lab at Wayne State University, All Rights Reserved.

Figure 7.2 Welcome page of the prototype system for computer-aided TPD

7.1.1. Class Diagram

The proposed system follows the object-oriented programming concept to utilize Java-based natural language processing Application Programming Interfaces (APIs) available with General Public Licenses (GPLs) and to implement a web-based user interface. To realize the system architecture in the previous section, the class diagram in the following figure is developed. The class diagram illustrates the major classes implemented in the proposed system from the functional requirement analysis to the design transformation via the interaction network and the semantic similarity score. The class *TextFileHandler* plays a main role to extract the source of design knowledge from informal free text based functional requirements written in English with a lexical dictionary, *Word-Net* and to generate and store a parsed functional requirement information into *Functional Requirement Base* (FR Base), which is explained in the next section.

To translate the extracted source into design knowledge, there are three different classes: (1) FunctionCandidateFinder, (2) StructureCandidateFinder, and (3) BehaviorFinder. The three classes help to identify functions, structures, behaviors from the extracted design knowledge source, respectively. The class *INGenerator* visualizes the translated design knowledge in the proposed design representation model, which is *interaction network*.

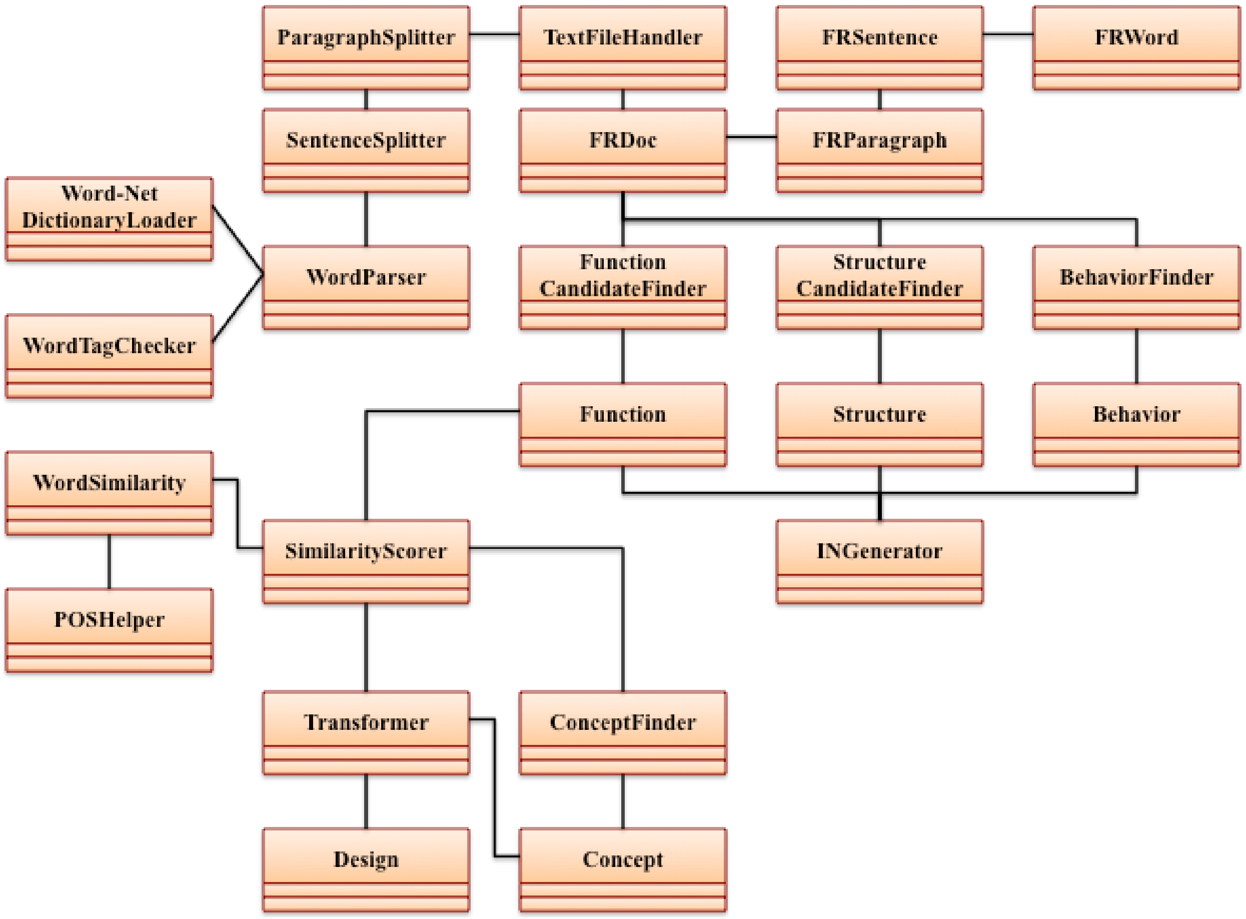


Figure 7.3 Class diagram of the prototype system

The class *SimilarityScorer* plays a role of the proposed semantics similarity score for both the concept detection and transformation processes. The class *Transformer* calls *SimilarityScorer* to check the degree of transformability of other products for the select concept.

7.1.2. Enhanced Entity Relationship Diagram Model

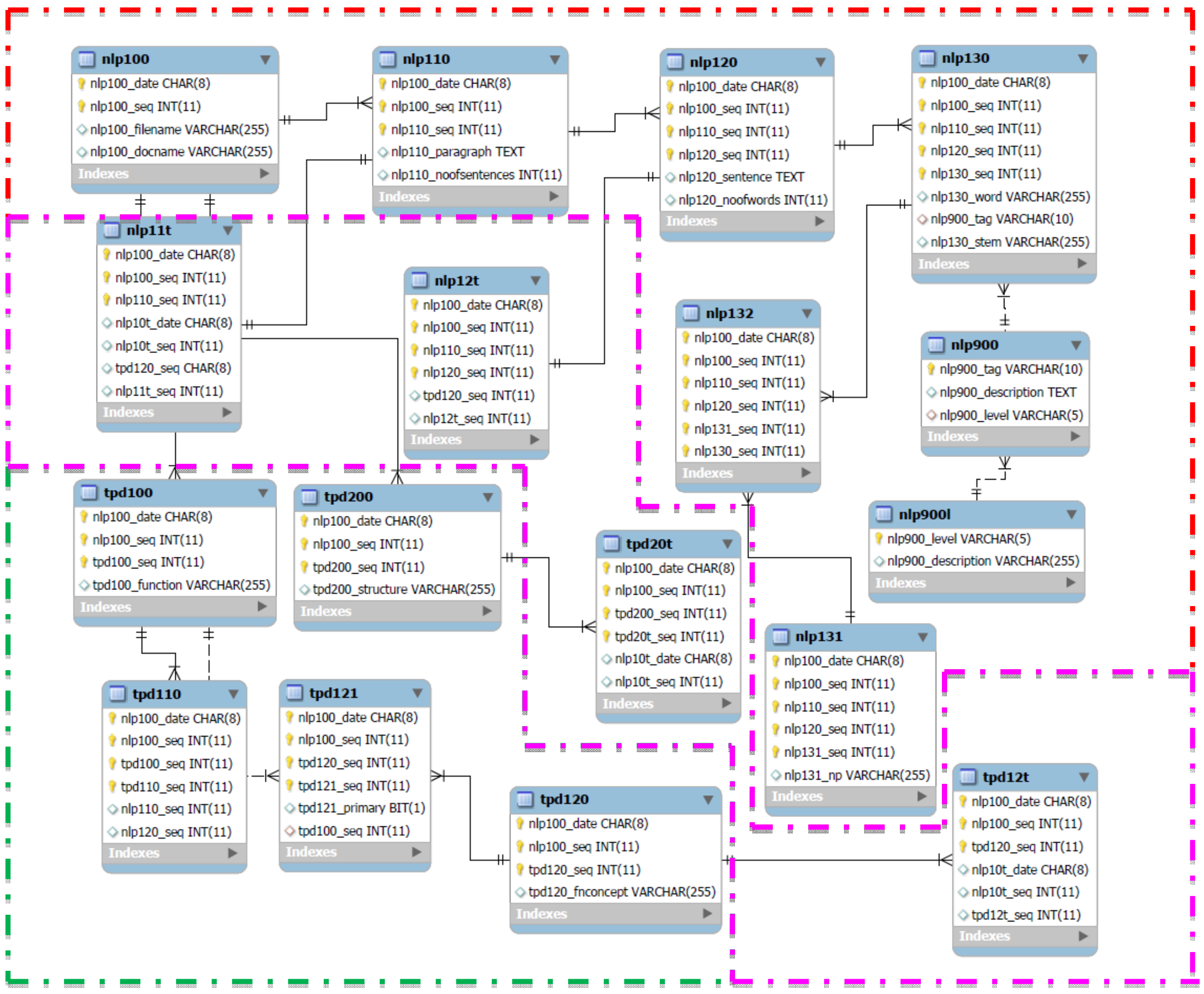


Figure 7.4 Enhanced ER diagram of the prototype system

Figure 7.4 illustrates the enhanced entity-relationship diagram for the prototype system. The tables that have names starting with '*nlp*' in the red dotted lines belong to the functional requirement base (FR Base), while the tables that have names starting with '*tpd*' in the green dotted lines belong to the interaction network base (IN Base). The tables that have names ending with*'t'* (i.e., *nlp11t*, *nlp12t*, *tpd12t*, and *tpd20t*), in purple dotted lines, are excluded from the two repositories and are associated with the design knowledge base (DK Base). The following lists the description of each table:

- nlp100: functional requirement document written in a natural language
- nlp110: functional requirement paragraph
- nlp120: functional requirement sentence
- nlp130: functional requirement word
- nlp131: structure candidates
- nlp132: association between structure candidates and functional requirement word
- nlp900: *part-of-speech* of English
- nlp900l: word level of each part-of-speech
- tpd100: function schema
- tpd110: functional requirement sentence associated with function schema
- tpd120: concept function
- tpd121: primitive and extended functions belonging to concept function
- tpd200: structure schema
- nlp11t: transformed functional requirement paragraph
- nlp12t: transformed functional requirement sentence
- tpd12t: transformed function schema
- tpd20t: transformed structure schema

7.2. System Implementation

A prototype system is to demonstrate the feasibility of the proposed algorithms in this research with realistic examples in a web-based environment. To enable the collaborative design environment for transformative product design, the system is designed to support a web-based environment. To fulfill this requirement, Java servlet with Java dynamic application and JDK 1.6, HTML, cascading style sheets, and Apache Tomcat Server 6.0 are considered for the web technologies. For the knowledge bases of the system, MySQL Community Server 5.1.63 is selected for the database management system. To enable the communication between the database and the system, the JDBC of MySQL Connector for Java that is provided by the database management system is adopted.

One of key functionalities of the system is to employ the natural language processing technologies. To support the technologies, two Java APIs are chosen: Stanford Natural Language Parser, which is a probabilistic parser and an interface that works out the grammatical structure of sentences for multiple natural languages, such as English, German, Arabic and Chinese [Klein and Manning, 2003a; Klein and Manning, 2003b; de Marneffe et al., 2006], and MIT Java WordNet Interface (JWI 2.2.3), which is a Java library for interfacing with WordNet. WordNet is a freely and publicly available semantic dictionary of English, developed at Princeton University and the version used in this system is 2.1 [Miller, 1995; Fellbaum, 1998]. For the network visualization, Gephi 0.8.1, the open graph visualization and exploration platform, is adopted and gexf4j, the Java GEXF (Graph Exchange XML Format) API [Ficarola, 2012], is employed for the interface between the system and the input file for the Gephi [Bastian *et al.*, 2009].

7.2.1. System Flow

The following figures describe the system flow of each process presented in Chapters 3, 4, and 5. Each figure contains user interfaces of each step to give insights to readers. The first figure illustrates the interaction network generation process. By employing the NLP tools, the functional requirement description is analyzed and de-documented into parsed sentences as the source for the design knowledge extraction. The following three steps are the identification process to detect functions, structures, and their interactions. These steps require human intervention to select appropriate functions and structures from the de-documented functional description. The final step of this process is to visualize the generated interaction network according to the functions, structures, and behaviors.

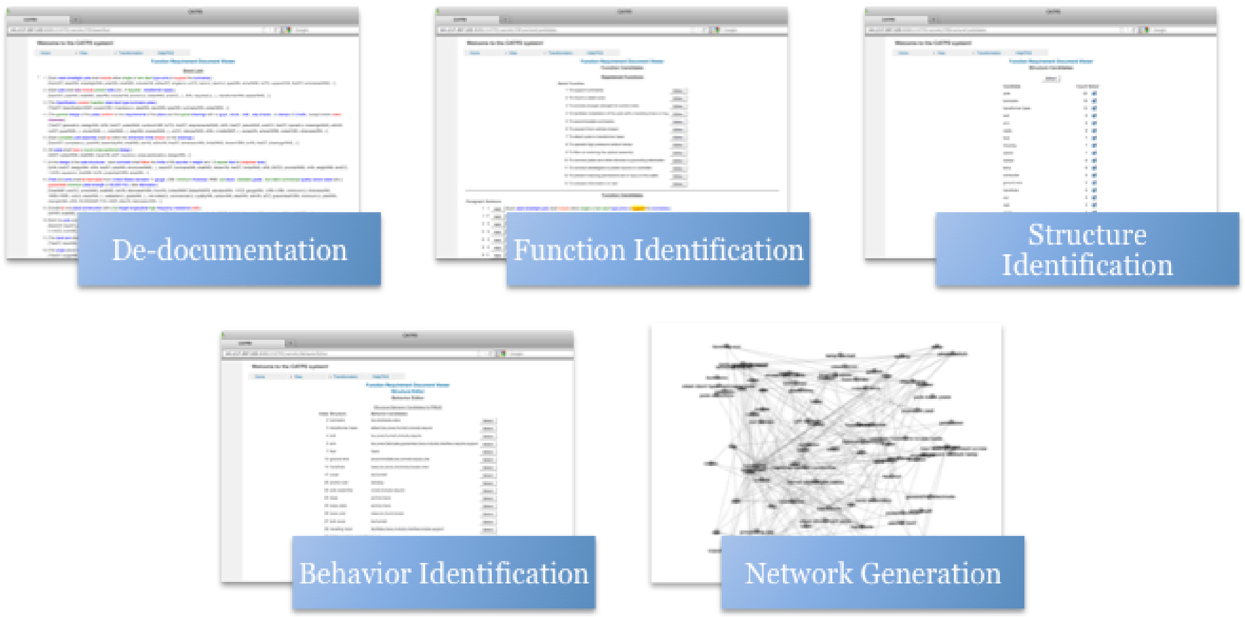


Figure 7.5 System flow of the interaction network generation process

To generate a transformative product design, the very first step is for designers to identify an advanced technology (or features/functions) from the base product. The next figure describes this process. A designer provides a functional keyword (phrase or sentence) written in a natural language to look up a functional concept from the base product. Then the system measures the semantic similarity scores of each function of the base product. If a designer selects appropriate primitive function(s), then the system automatically provides the extended concept by exploring the interaction network of the base product.

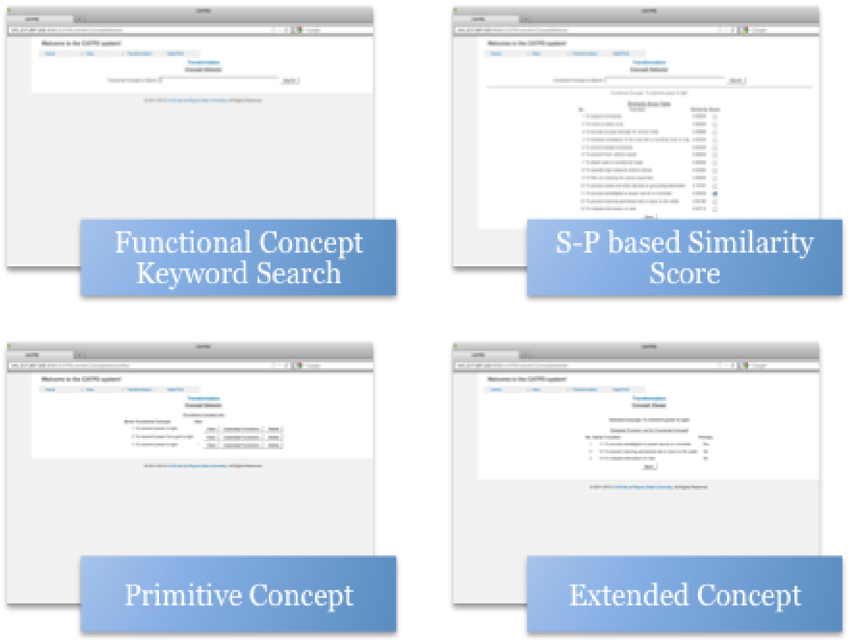


Figure 7.6 System flow of the concept detection process

The beauty of the implemented system is the transformation process in the following figure. This process reclaims the proposed similarity score to check the degree of transformability of other products. Given the select concept function of the base product, including its primitive function(s) and extended functions, the system explores the interaction network repository and returns the degree of transformability of other products in descending order, if the degree is greater than the pre-determined threshold. When a designer chooses appropriate reference products, the system continues to regenerate a design document by examining both interaction networks of the base and reference products. Finally, the system systematically identifies a new interaction network with an associated GEXF file for visualization.

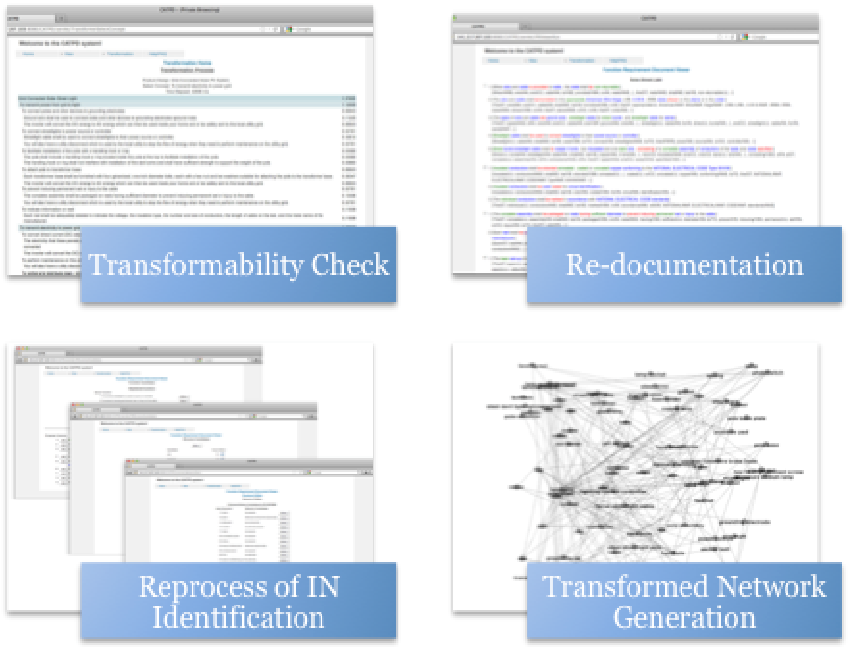

Transformability Check
Re-documentation
Reprocess of IN Identification
Transformed Network Generation

Figure 7.7 System flow of the transformation process

The next three subsections list the three representative source codes associated with the proposed research methods in Chapters 3, 4, and 5. The source codes of classes related to natural language processing tools, for readers' reference, are listed in Appendices C, D, E, and F.

7.2.2. Similarity Score

The following listing shows the source code to implement the Stem-POS based Similarity score in order to identify the primitive concept functions in Chapter 4 as well as the validation process for the transformative product design alternatives in Chapter 5. This listing is part of the similarity score to compare a pair of words in two phrases or sentences, respectively.

```
public class WordSimilarity {
    private IDictionary dict;
    private String word1;
    private String word2;
    private POS pos1;
    private POS pos2;
    private POSHelper helper1;
    private POSHelper helper2;

    private double simscore = 0.0;

    public WordSimilarity (String word1, String word2, POS pos) {
        this.word1 = word1;
        this.word2 = word2;
        this.pos1 = pos;
        this.pos2 = pos;

        this.init();
    }

    public WordSimilarity (String word1, String word2, POS pos1, POS pos2) {
        this.word1 = word1;
        this.word2 = word2;
        this.pos1 = pos1;
        this.pos2 = pos2;

        this.init();
    }

    private void init() {
        this.dict = new WNDic().getDictionary();

        this.helper1 = new POSHelper(this.dict, this.word1, this.pos1);
        this.helper2 = new POSHelper(this.dict, this.word2, this.pos2);
    }
```

```
public double getSimilarity() {
    double synonym_sim = this.getSimilarityInSynonyms();
    double hypernym_sim = this.getSimilarityInSemantics(Pointer.HYPERNYM);
    double hyponym_sim = this.getSimilarityInSemantics(Pointer.HYPONYM);
    double holonym_sim = this.getSimilarityInSemantics(Pointer.HOLONYM_PART);
    holonym_sim = Math.max(holonym_sim,
                            this.getSimilarityInSemantics(Pointer.HOLONYM_MEMBER));
    double meronym_sim = this.getSimilarityInSemantics(Pointer.MERONYM_PART);
    meronym_sim = Math.max(meronym_sim,
                            this.getSimilarityInSemantics(Pointer.MERONYM_MEMBER));

    this.simscore = Math.max(meronym_sim, holonym_sim);
    this.simscore = Math.max(hyponym_sim, this.simscore);
    this.simscore = Math.max(hypernym_sim, this.simscore);
    this.simscore = Math.max(synonym_sim, this.simscore);

    return this.simscore;
}

public double getSimilarityInSynonyms() {
    double similarity = 0.0;

    int card1 = 0;
    int card2 = 0;
    int commoncard = 0;

    List<String> stems1 = this.helper1.getStems();
    List<String> stems2 = this.helper2.getStems();

    try {
        for (String stem1 : stems1) {
            IIndexWord idxWord1 = this.helper1.getIndexWord(stem1, this.pos1);
            IWordID wordID1 = this.helper1.getWordID(idxWord1);
            IWord word1 = this.helper1.getWord(wordID1);
            ISynset synset1 = this.helper1.getSynset(word1);
            List<IWord> synonyms1 = this.helper1.getSynonyms(synset1);

            card1 += synonyms1.size();

            stem1 = stem1.replaceAll("_", " ");

            for (String stem2 : stems2) {
                IIndexWord idxWord2 = this.helper2.getIndexWord(stem2, this.pos2);
                IWordID wordID2 = this.helper2.getWordID(idxWord2);
                IWord word2 = this.helper2.getWord(wordID2);
                ISynset synset2 = this.helper2.getSynset(word2);
```

```
                    List<IWord> synonyms2 = this.helper2.getSynonyms(synset2);

                    card2 += synonyms2.size();

                    for (IWord synonym1 : synonyms1) {
                        String synword1 = synonym1.getLemma().toString();
                        synword1.replaceAll("_", " ");

                        boolean isMatched = false;

                        for (IWord synonym2 : synonyms2) {
                            String synword2 = synonym2.getLemma().toString();
                            synword2.replaceAll("_", " ");

                            if (!isMatched && synword1.equalsIgnoreCase(synword2)) {
                                commoncard++;
                                isMatched = true;
                            } else {
                                if (!isMatched && synword1.contains(synword2)) {
                                    commoncard++;
                                    isMatched = true;
                                } else {
                                    if (!isMatched && synword2.contains(synword1)) {
                                        commoncard++;
                                        isMatched = true;
                                    }
                                }
                            }
                        }
                    }

                    stem2 = stem2.replaceAll("_", " ");
                }
            }
        } catch (NullPointerException e) {
//          System.out.println("Error: WordSimilarity.getSimilarityInSynonyms():
NullPointerException...");
        }

        double sim1 = (card1 == 0) ? 0.0 : (double)commoncard / (double)card1;
        double sim2 = (card2 == 0) ? 0.0 : (double)commoncard / (double)card2;

        similarity = Math.max(sim1, sim2);
        similarity *= (this.pos1 != this.pos2) ? 0.5 : 1.0;

        return similarity;
```

```
    }

    public double getSimilarityInSemantics(Pointer pointer) {
        double similarity = 0.0;

        int card1 = 0;
        int card2 = 0;
        int commoncard = 0;

        List<String> stems1 = this.helper1.getStems();
        List<String> stems2 = this.helper2.getStems();

        try {
            for (String stem1 : stems1) {
                IIndexWord idxWord1 = this.helper1.getIndexWord(stem1, this.pos1);
                IWordID wordID1 = this.helper1.getWordID(idxWord1);
                IWord word1 = this.helper1.getWord(wordID1);
                ISynset synset1 = this.helper1.getSynset(word1);
                List<String> semanticwords1 = this.helper1.getSemanticWords(synset1, pointer);

                card1 += semanticwords1.size();

                stem1 = stem1.replaceAll("_", " ");

                for (String stem2 : stems2) {
                    IIndexWord idxWord2 = this.helper2.getIndexWord(stem2, this.pos2);
                    IWordID wordID2 = this.helper2.getWordID(idxWord2);
                    IWord word2 = this.helper2.getWord(wordID2);
                    ISynset synset2 = this.helper2.getSynset(word2);
                    List<String> semanticwords2 = this.helper2.getSemanticWords(synset2,
pointer);

                    card2 += semanticwords2.size();

                    for (String semanticword1 : semanticwords1) {
                        semanticword1.replaceAll("_", " ");

                        boolean isMatched = false;

                        for (String semanticword2 : semanticwords2) {
                            semanticword2.replaceAll("_", " ");

                            if (!isMatched && semanticword1.equalsIgnoreCase(semanticword2))
{
                                commoncard++;
                                isMatched = true;
```

```
                                    } else {
                                        if (!isMatched && semanticword1.contains(semanticword2)) {
                                            commoncard++;
                                            isMatched = true;
                                        } else {
                                            if (!isMatched && semanticword2.contains(semanticword1)) {
                                                commoncard++;
                                                isMatched = true;
                                            }
                                        }
                                    }
                                }
                            }

                            stem2 = stem2.replaceAll("_", " ");
                        }
                    }
                } catch (NullPointerException e) {
//              System.out.println("Error: WordSimilarity.getSimilarityInSemantics():
NullPointerException...");
                }

                double sim1 = (card1 == 0) ? 0.0 : (double)commoncard / (double)card1;
                double sim2 = (card2 == 0) ? 0.0 : (double)commoncard / (double)card2;

                similarity = Math.max(sim1, sim2);
                similarity *= (this.pos1 != this.pos2) ? 0.5 : 1.0;

                if (pointer == Pointer.HYPERNYM || pointer == Pointer.HYPONYM) {
                    similarity *= 0.9;
                }
                if (pointer == Pointer.HOLONYM_PART || pointer == Pointer.HOLONYM_MEMBER)
{
                    similarity *= 0.7;
                }
                if (pointer == Pointer.MERONYM_PART || pointer ==
Pointer.MERONYM_MEMBER) {
                    similarity *= 0.7;
                }

                return similarity;
    }
}
```

Figure 7.8 Listing of the class WordSimilarity being used for the SPS score

7.2.3. Concept Detection Algorithm

The following listing denotes the source code to implement the extended concept detection algorithm in Chapter 4.

```
public class ConceptFinder {
    private String date;
    private int serial;
    private int cserial;

    private LinkedList<Integer> fserials = new LinkedList<Integer>();
    private LinkedList<String> functions = new LinkedList<String>();

    private String ssql, sqlS, sqlF, sqlJ, sqlW, sqlG, sqlH, sqlO, sqlI, sqlV, sqlU, sqlD, sqlC;
    private SQLHandler sqlh;
    private ResultSet rs = null;

    public ConceptFinder(String date, int serial) {
        this.date = date;
        this.serial = serial;
    }

    public ConceptFinder(String date, String serial) {
        this.date = date;
        this.serial = Integer.parseInt(serial);
    }

    public ConceptFinder(String date, int serial, int cserial) {
        this.date = date;
        this.serial = serial;
        this.cserial = cserial;
    }

    public ConceptFinder(String date, String serial, String cserial) {
        this.date = date;
        this.serial = Integer.parseInt(serial);
        this.cserial = Integer.parseInt(cserial);
    }

    public void run() {
        // Investigate additional relevant functions and Save them
        // 1. Detect direct behaviors related to primary functions
        // 2. Detect direct structures related to the behaviors
        // 3. Detect structural behaviors related to the structures
        // 4. Detect functions related to the structural behaviors
        // To Retrieve Additional Relevant Functions
        this.sqlS = "SELECT DISTINCT sb.nlp100_date, sb.nlp100_seq, f.tpd100_seq,
f.tpd100_function ";

        // SUB-QUERY 1: Structure Sentences (Structural Behavior)
        this.sqlJ = "SELECT DISTINCT s.nlp100_date, s.nlp100_seq, ";
        this.sqlJ += "s.tpd200_seq, s.tpd200_structure, ";
        this.sqlJ += "n.nlp110_seq, n.nlp120_seq ";
        this.sqlJ += "FROM catpd.tpd200 s INNER JOIN catpd.nlp131 n ";
        this.sqlJ += "ON s.nlp100_date = n.nlp100_date ";
        this.sqlJ += "AND s.nlp100_seq = n.nlp100_seq ";
        this.sqlJ += "AND s.tpd200_structure = n.nlp131_np ";
        this.sqlJ += "WHERE s.nlp100_date = '"+ this.date + "' ";
        this.sqlJ += "AND s.nlp100_seq = "+ String.valueOf(this.serial);

        this.sqlF = "FROM (" + this.sqlJ + ") sb ";

        // SUB-QUERY 2: Structures
        this.sqlC = "SELECT DISTINCT ts1.nlp100_date, ts1.nlp100_seq, ";
        this.sqlC += "ts1.tpd200_seq AS tpd200_seq1, ts1.tpd200_structure AS
tpd200_structure1, ";
        this.sqlC += "ts2.tpd200_seq AS tpd200_seq2, ts2.tpd200_structure AS
tpd200_structure2 ";

        // SUB-QUERY 2-1 & 2-2:
        this.sqlJ = "SELECT s.nlp100_date, s.nlp100_seq, s.tpd200_seq, s.tpd200_structure, ";
        this.sqlJ += "n.nlp110_seq, n.nlp120_seq, n.nlp131_seq ";
        this.sqlJ += "FROM (";
        this.sqlJ += "SELECT * FROM catpd.tpd200 ";
        this.sqlJ += "WHERE nlp100_date = '" + this.date + "' ";
        this.sqlJ += "AND nlp100_seq = " + String.valueOf(this.serial) + " ) s ";
        this.sqlJ += "INNER JOIN catpd.nlp131 n ";
        this.sqlJ += "ON s.nlp100_date = n.nlp100_date ";
        this.sqlJ += "AND s.nlp100_seq = n.nlp100_seq ";
        this.sqlJ += "AND s.tpd200_structure = n.nlp131_np";

        this.sqlC += "FROM (" + this.sqlJ + ") ts1 INNER JOIN (" + this.sqlJ + ") ts2 ";
        this.sqlC += "ON ts1.nlp100_date = ts2.nlp100_date ";
        this.sqlC += "AND ts1.nlp100_seq = ts2.nlp100_seq ";
        this.sqlC += "AND ts1.nlp110_seq = ts2.nlp110_seq ";
        this.sqlC += "AND ts1.nlp120_seq = ts2.nlp120_seq ";
        this.sqlC += "AND ts1.tpd200_seq <> ts2.tpd200_seq ";

        // SUB-QUERY 2-3: Primary Functions for Functional Concept
        this.sqlJ = "SELECT c.nlp100_date, c.nlp100_seq, c.tpd120_seq, c.tpd120_fnconcept, ";
        this.sqlJ += "pc.tpd121_seq, pc.tpd121_primary, pc.tpd100_seq, ";
        this.sqlJ += "f.tpd100_function, fs.nlp110_seq, fs.nlp120_seq ";
        this.sqlJ += "FROM catpd.tpd120 c INNER JOIN catpd.tpd121 pc ";
        this.sqlJ += "ON c.nlp100_date = pc.nlp100_date ";
        this.sqlJ += "AND c.nlp100_seq = pc.nlp100_seq ";
        this.sqlJ += "AND c.tpd120_seq = pc.tpd120_seq ";
        this.sqlJ += "INNER JOIN catpd.tpd100 f ";
        this.sqlJ += "ON c.nlp100_date = f.nlp100_date ";
        this.sqlJ += "AND c.nlp100_seq = f.nlp100_seq ";
        this.sqlJ += "AND pc.tpd100_seq = f.tpd100_seq ";
        this.sqlJ += "INNER JOIN catpd.tpd110 fs ";
        this.sqlJ += "ON c.nlp100_date = fs.nlp100_date ";
        this.sqlJ += "AND c.nlp100_seq = fs.nlp100_seq ";
        this.sqlJ += "AND pc.tpd100_seq = fs.tpd100_seq ";
        this.sqlJ += "WHERE c.nlp100_date = '" + this.date + "' ";
        this.sqlJ += "AND c.nlp100_seq = " + String.valueOf(this.serial) + " ";
        this.sqlJ += "AND c.tpd120_seq = " + String.valueOf(this.cserial) + " ";
        this.sqlJ += "AND pc.tpd121_primary = TRUE";

        // Relational Condition to Obtain Structures that Associate to Primary Functions
        this.sqlC += "INNER JOIN (" + this.sqlJ + ") p ";
        this.sqlC += "ON ts1.nlp100_date = p.nlp100_date ";
        this.sqlC += "AND ts1.nlp100_seq = p.nlp100_seq ";
        this.sqlC += "AND ts1.nlp110_seq = p.nlp110_seq ";
        this.sqlC += "AND ts1.nlp120_seq = p.nlp120_seq";

        // Relational Condition for Matching Structural Behavior and Structures
        this.sqlF += "INNER JOIN (" + this.sqlC + ") s ";
        this.sqlF += "ON sb.nlp100_date = s.nlp100_date ";
        this.sqlF += "AND sb.nlp100_seq = s.nlp100_seq ";
        this.sqlF += "AND (sb.tpd200_seq = s.tpd200_seq1 OR sb.tpd200_seq = s.tpd200_seq2)
";

        // SUB-QUERY 3: Functional Sentences
        this.sqlF += "INNER JOIN catpd.tpd110 fs ";
        this.sqlF += "ON sb.nlp100_date = fs.nlp100_date ";
        this.sqlF += "AND sb.nlp100_seq = fs.nlp100_seq ";
        this.sqlF += "AND sb.nlp110_seq = fs.nlp110_seq ";
        this.sqlF += "AND sb.nlp120_seq = fs.nlp120_seq ";

        // SUB-QUERY 4: Functions
        this.sqlF += "INNER JOIN catpd.tpd100 f ";
        this.sqlF += "ON sb.nlp100_date = f.nlp100_date ";
        this.sqlF += "AND sb.nlp100_seq = f.nlp100_seq ";
        this.sqlF += "AND fs.tpd100_seq = f.tpd100_seq ";
        // SUB-QUERY 5: Primary Functions for Functional Concept (Reuse SUB-QUERY 2-1
& 2-2)
        // Relational Condition to Exclude Primary Functions
        this.sqlF += "LEFT JOIN (" + this.sqlJ + ") p ";
        this.sqlF += "ON sb.nlp100_date = p.nlp100_date ";
        this.sqlF += "AND sb.nlp100_seq = p.nlp100_seq ";
        this.sqlF += "AND f.tpd100_seq = p.tpd100_seq ";

        // Condition to Exclude Primary Functions
        this.sqlW = "WHERE p.tpd100_seq IS NULL ";
        this.sqlO = "ORDER BY sb.nlp100_date, sb.nlp100_seq, f.tpd100_seq";

        this.ssql = this.sqlS + this.sqlF + this.sqlW + this.sqlO;
        this.sqlh = new SQLHandler(this.ssql);
        this.rs = this.sqlh.getRSet();

        try {
            this.fserials = new LinkedList<Integer>();
            this.functions = new LinkedList<String>();

            while (this.rs.next()) {
                this.fserials.add(this.rs.getInt("tpd100_seq"));
                this.functions.add(this.rs.getString("tpd100_function"));
            }
        } catch (SQLException e) {
            System.out.println("ConceptExtender.run() SQL Error..." + e.getMessage());
        } catch (Exception e) {
            System.out.println("ConceptExtender.run() Error..." + e.getMessage());
        } finally {
            this.sqlh.destroy();
        }

        this.setExtendedFunctions();
    }

    private void setExtendedFunctions() {
        // Delete Existing Extended Functions
        this.sqlD = "DELETE FROM catpd.tpd121 ";
        this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
        this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
        this.sqlW += "AND tpd120_seq = " + String.valueOf(this.cserial) + " ";
        this.sqlW += "AND CASE WHEN tpd121_primary THEN true ELSE false END <> true
";

        this.ssql = this.sqlD + this.sqlW;
        this.sqlh = new SQLHandler(this.ssql);
        this.sqlh.destroy();

        // Save New Extended Functions
        for (int index = 0; index < this.fserials.size(); index++) {
            int efserial = this.getExtendedFnSerial(index);

            if (efserial > 0) {
                DataHelper helper = new DataHelper();

                String table = "catpd.tpd121";
                String [] fields = {"nlp100_date", "nlp100_seq", "tpd120_seq", "tpd100_seq"};
                String [] values = {this.date, String.valueOf(this.serial),
String.valueOf(this.cserial),
                                String.valueOf(efserial)};
                boolean [] bnumeric = {false, true, true, true};
                boolean isExist = helper.isExist(table, fields.length, fields, values, bnumeric);

                if (!isExist) {
                    // Simply DOUBLE-CHECK
                    // Save the function when it doesn't exist in the database
                    String field = "tpd121_seq";
                    int iNoOfKeys = 3;
                    String [] keyfields = {"nlp100_date", "nlp100_seq", "tpd120_seq"};
                    String [] keyvalues = {this.date, String.valueOf(this.serial),
String.valueOf(this.cserial)};
                    boolean [] isNumber = {false, true, true};

                    // New Serial Number for an Extended Function
                    int eserial = helper.getMaxSerial(field, table, iNoOfKeys, keyfields,
keyvalues, isNumber);

                    // Update the extended function
                    this.sqlU = "UPDATE catpd.tpd121 ";
                    this.sqlS = "SET tpd121_primary = false, ";
                    this.sqlS += "tpd100_seq = " + String.valueOf(efserial) + " ";
                    this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
                    this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
                    this.sqlW += "AND tpd120_seq = " + String.valueOf(this.cserial) + " ";
                    this.sqlW += "AND tpd121_seq = " + String.valueOf(eserial) + " ";

                    this.ssql = this.sqlU + this.sqlS + this.sqlW;
                    this.sqlh = new SQLHandler(this.ssql);
                    this.sqlh.destroy();
                }
            }
        }
```

```
}

public int getExtendedFnSerial(int index) {
    int serial = 0;

    if (index >= 0 || index < this.fserials.size()) {
        serial = this.fserials.get(index);
    }

    return serial;
}

public LinkedList<Integer> getExtendedFnSerials() {
    return this.fserials;
}

public String getExtendedFunctions(int index) {
    String function = "";

    if (index >= 0 || index < this.functions.size()) {
        function = this.functions.get(index);
    }

    return function;
}

public LinkedList<String> getExtendedFunctions() {
    return this.functions;
}

public void runAll() {
    this.sqlS = "SELECT c.nlp100_date, c.nlp100_seq, c.tpd120_seq, c.tpd120_fnconcept ";
    this.sqlF = "FROM catpd.tpd120 c ";
    this.sqlW = "WHERE c.nlp100_date = '" + this.date + "' ";
    this.sqlW += "AND c.nlp100_seq = " + String.valueOf(this.serial) + " ";

    this.ssql = this.sqlS + this.sqlF + this.sqlW;
    SQLHandler handler = new SQLHandler(this.ssql);
    ResultSet rset = handler.getRSet();

    try {
        while (rset.next()) {
            this.cserial = rset.getInt("tpd120_seq");
            this.run();
        }
    } catch (SQLException e) {
            System.out.println("ConceptExtender.runAll() SQL Error..." + e.getMessage());
        } catch (Exception e) {
            System.out.println("ConceptExtender.runAll() Error..." + e.getMessage());
        } finally {
            handler.destroy();
        }

        this.fserials.clear();
        this.functions.clear();
    }

}
```

Figure 7.9 Listing of the class ConceptFinder being used for Concept Detection Algorithm

7.2.4. Transformation Algorithm

The following listing presents the source code to implement the transformation algorithm in Chapter 5.

```
public class Transformer {
private String date;
private int serial;
private String docname;

private String conceptfunctions = "";

private String ssql, sqlS, sqlF, sqlJ, sqlW, sqlG, sqlO, sqlI, sqlV, sqlU, sqlD, sqlC;
private SQLHandler sqlh;
private ResultSet rs = null;

/**
* Constructor
* When there are no given primary keys for a new transformed document,
* new primary keys will be given automatically.
*/
public Transformer() {
this.init();
this.docname = "";
}

/**
* Constructor
* @param docname: Transformation Document Name to Update
*/
public Transformer(String docname) {
this.init();
this.docname = docname;
this.updateDocName(docname);
}

/**
* Constructor
* @param date: Transformation Document Primary Key of Date
* @param serial: Transformation Document Primary Key of Serial
*/
public Transformer(String date, int serial) {
this.date = date;
this.serial = serial;
this.setDocName();
}

/**
* Constructor
* @param date: Transformation Document Primary Key of Date
* @param serial: Transformation Document Primary Key of Date
* @param docname: Transformation Document Name
*/
public Transformer(String date, int serial, String docname) {
this.date = date;
this.serial = serial;
this.docname = docname;
this.updateDocName(docname);
}

/**
* init()
* To create a set of primary keys for a new transformation document
*/
private void init() {
DataHelper helper = new DataHelper();
String date = helper.getToday();

String table = "catpd.nlp100";
String field = "nlp100_seq";

String [] keyfields = {"nlp100_date"};
String [] keyvalues = {date};
boolean [] isNumber = {false};
int iNoOfKeys = keyfields.length;

int serial = helper.getMaxSerial(field, table, iNoOfKeys, keyfields, keyvalues, isNumber);

this.date = date;
this.serial = serial;
}

/**
* setDocName()
* To set the transformation document name variable internally
*/
private void setDocName() {
DataHelper helper = new DataHelper();

String table = "catpd.nlp100";
String field = "nlp100_docname";

String [] keyfields = {"nlp100_date", "nlp100_seq"};
String [] keyvalues = {this.date, String.valueOf(this.serial)};
boolean [] isNumber = {false, true};

this.docname = helper.getString(table, field, keyfields, keyvalues, isNumber);
}

/**
* updateDocName()
* To update the transformation document name
*/
public void updateDocName(String docname) {
DataHelper helper = new DataHelper();

String table = "catpd.nlp100";
String field = "nlp100_docname";
boolean isNumeric = false;

String [] keyfields = {"nlp100_date", "nlp100_seq"};
String [] keyvalues = {this.date, String.valueOf(this.serial)};
boolean [] isNumber = {false, true};

helper.updateField(table, field, docname, isNumeric, keyfields, keyvalues, isNumber);
}

/**
* getDate()
* @return Transformation Date
```

```
*/
public String getDate() {
return this.date;
}

/**
* getSerial()
* @return Transformation Serial
*/
public int getSerial() {
return this.serial;
}

/**
* getDocName()
* @return Transformation Document Name
*/
public String getDocName() {
return this.docname;
}

/**
* transform()
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param cindex: Functional Concept Primary Key of Concept Serial
*/
public void transform(String date, int serial, int cindex) {
this.clearConcept(date, serial, cindex);
this.getConceptFunctions(date, serial, cindex);
this.importDesign(date, serial, cindex);
this.setWords();
this.setFunctions(date, serial, cindex);
this.setStructures(date, serial);

String pk = this.date + "-" + String.valueOf(this.serial) + " with ";
pk += date + "-" + String.valueOf(serial) + "-" + String.valueOf(cindex);
System.out.println("Successfully completed... " + pk);
}

/**
* clearConcept
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param cindex: Functional Concept Primary Key of Concept Serial
*/
```

```
private void clearConcept(String date, int serial, int cindex) {
//Delete Existing Sentences
this.sqlS = "SELECT DISTINCT s.nlp110_seq, s.nlp120_seq ";
this.sqlF = "FROM catpd.nlp120 s INNER JOIN catpd.nlp12t c ";
this.sqlF += "ON s.nlp100_date = c.nlp100_date ";
this.sqlF += "AND s.nlp100_seq = c.nlp100_seq ";
this.sqlF += "AND s.nlp110_seq = c.nlp110_seq ";
this.sqlW = "WHERE s.nlp100_date = '" + this.date + "' ";
this.sqlW += "AND s.nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND c.tpd120_seq = " + String.valueOf(cindex) + " ";
this.sqlO = "ORDER BY s.nlp110_seq";

this.ssql = this.sqlS + this.sqlF + this.sqlW + this.sqlO;
this.sqlh = new SQLHandler(this.ssql);
this.rs = this.sqlh.getRSet();

try {
SQLHandler handler;

while (this.rs.next()) {
int pindex = this.rs.getInt("nlp110_seq");
int sindex = this.rs.getInt("nlp120_seq");

this.sqlD = "DELETE FROM catpd.nlp12t ";
this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND nlp110_seq = " + String.valueOf(pindex) + " ";
this.sqlW += "AND nlp120_seq = " + String.valueOf(sindex) + " ";

this.ssql = this.sqlD + this.sqlW;
handler = new SQLHandler(this.ssql);
handler.destroy();

this.sqlD = "DELETE FROM catpd.nlp120 ";
this.ssql = this.sqlD + this.sqlW;
handler = new SQLHandler(this.ssql);
handler.destroy();
}
} catch (SQLException e) {
System.out.println("Transformer.clearConcept() SQL Error..." + e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.out.println("Transformer.clearConcept() Error..." + e.getMessage());
e.printStackTrace();
} finally {
this.sqlh.destroy();
```

```
}

//Delete Existing Paragraphs
this.sqlS = "SELECT DISTINCT p.nlp110_seq ";
this.sqlF = "FROM catpd.nlp110 p INNER JOIN catpd.nlp11t c ";
this.sqlF += "ON p.nlp100_date = c.nlp100_date ";
this.sqlF += "AND p.nlp100_seq = c.nlp100_seq ";
this.sqlF += "AND p.nlp110_seq = c.nlp110_seq ";
this.sqlW = "WHERE p.nlp100_date = '" + this.date + "' ";
this.sqlW += "AND p.nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND c.nlp10t_date = '" + date + "' ";
this.sqlW += "AND c.nlp10t_seq = " + String.valueOf(serial) + " ";
this.sqlW += "AND c.tpd120_seq = " + String.valueOf(cindex) + " ";
this.sqlO = "ORDER BY p.nlp110_seq";

this.ssql = this.sqlS + this.sqlF + this.sqlW + this.sqlO;
this.sqlh = new SQLHandler(this.ssql);
this.rs = this.sqlh.getRSet();

try {
SQLHandler handler;

while (this.rs.next()) {
int pindex = this.rs.getInt("nlp110_seq");

this.sqlD = "DELETE FROM catpd.nlp11t ";
this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND nlp110_seq = " + String.valueOf(pindex) + " ";

this.ssql = this.sqlD + this.sqlW;
handler = new SQLHandler(this.ssql);
handler.destroy();

this.sqlD = "DELETE FROM catpd.nlp110 ";
this.ssql = this.sqlD + this.sqlW;
handler = new SQLHandler(this.ssql);
handler.destroy();
}
} catch (SQLException e) {
System.out.println("Transformer.clearConcept() SQL Error..." + e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.out.println("Transformer.clearConcept() Error..." + e.getMessage());
e.printStackTrace();
} finally {
```

```
this.sqlh.destroy();
}

}

/**
* getConceptFunctions()
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param cindex: Functional Concept Primary Key of Concept Serial
*/
private void getConceptFunctions(String date, int serial, int cindex) {
this.conceptfunctions = "";

this.sqlS = "SELECT c.tpd100_seq ";
this.sqlF = "FROM catpd.tpd121 c ";
this.sqlW = "WHERE c.nlp100_date = '" + date + "' ";
this.sqlW += "AND c.nlp100_seq = " + String.valueOf(serial) + " ";
this.sqlW += "AND c.tpd120_seq = " + String.valueOf(cindex) + " ";
this.sqlO = "ORDER BY c.tpd100_seq";

this.ssql = this.sqlS + this.sqlF + this.sqlW + this.sqlO;
this.sqlh = new SQLHandler(this.ssql);
this.rs = this.sqlh.getRSet();

try {
while (this.rs.next()) {
String field = "tpd100_seq";
Object obj = this.rs.getObject(field);

if (obj != null) {
this.conceptfunctions += this.conceptfunctions.equals("") ? "" : ", ";
this.conceptfunctions += this.rs.getString(field);
}
}
} catch (SQLException e) {
System.out.println("Transformer.getConceptFunctions() SQL Error..." + e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.out.println("Transformer.getConceptFunctions() Error..." + e.getMessage());
e.printStackTrace();
} finally {
this.sqlh.destroy();
}
}
```

```
/**
* importDesign()
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param cindex: Functional Concept Primary Key of Concept Serial
*/
private void importDesign(String date, int serial, int cindex) {
//Structural Behaviors with respect to Concept Functions
this.sqlS = "SELECT DISTINCT ts1.nlp100_date, ts1.nlp100_seq, ";
this.sqlS += "ts1.tpd200_seq AS tpd200_seq1, ts1.tpd200_structure AS structure1, ";
this.sqlS += "ts2.tpd200_seq AS tpd200_seq2, ts2.tpd200_structure AS structure2 ";

//SUB-QUERY: Structure
this.sqlJ = "SELECT s.nlp100_date, s.nlp100_seq, s.tpd200_seq, s.tpd200_structure, ";
this.sqlJ += "n.nlp110_seq, n.nlp120_seq, n.nlp131_seq ";
this.sqlJ += "FROM (";
this.sqlJ += "SELECT * FROM catpd.tpd200 ";
this.sqlJ += "WHERE nlp100_date = '" + date + "' ";
this.sqlJ += "AND nlp100_seq = " + String.valueOf(serial) + ") s ";
this.sqlJ += "INNER JOIN catpd.nlp131 n ";
this.sqlJ += "ON s.nlp100_date = n.nlp100_date ";
this.sqlJ += "AND s.nlp100_seq = n.nlp100_seq ";
this.sqlJ += "AND s.tpd200_structure = n.nlp131_np";

//Relational Condition for Structural Behaviors
this.sqlF = "FROM (" + this.sqlJ + ") ts1 INNER JOIN (" + this.sqlJ + ") ts2 ";
this.sqlF += "ON ts1.nlp100_date = ts2.nlp100_date ";
this.sqlF += "AND ts1.nlp100_seq = ts2.nlp100_seq ";
this.sqlF += "AND ts1.nlp110_seq = ts2.nlp110_seq ";
this.sqlF += "AND ts1.nlp120_seq = ts2.nlp120_seq ";
this.sqlF += "AND ts1.tpd200_seq <> ts2.tpd200_seq ";

//SUB-QUERY: Concept Functions
this.sqlJ = "SELECT DISTINCT fs.nlp100_date, fs.nlp100_seq, fs.nlp110_seq, fs.nlp120_seq ";
this.sqlJ += "FROM catpd.tpd110 fs INNER JOIN catpd.tpd100 f ";
this.sqlJ += "ON fs.nlp100_date = f.nlp100_date ";
this.sqlJ += "AND fs.nlp100_seq = f.nlp100_seq ";
this.sqlJ += "AND fs.tpd100_seq = f.tpd100_seq ";
this.sqlJ += "WHERE f.tpd100_seq IN (" + this.conceptfunctions + ")";

//Relational Condition for Structural Behaviors and Concept Functions
this.sqlF += "INNER JOIN (" + this.sqlJ + ") tf ";
this.sqlF += "ON ts1.nlp100_date = tf.nlp100_date ";
this.sqlF += "AND ts1.nlp100_seq = tf.nlp100_seq ";
this.sqlF += "AND ts1.nlp110_seq = tf.nlp110_seq ";
this.sqlF += "AND ts1.nlp120_seq = tf.nlp120_seq";
this.sqlJ = this.sqlS + this.sqlF;

//All the sentences related to Structural Behaviors with respect to Concept Functions
this.sqlS = "SELECT DISTINCT n.nlp100_date, n.nlp100_seq, n.nlp110_seq, n.nlp120_seq,
s.nlp120_sentence ";

this.sqlF = "FROM catpd.nlp131 n INNER JOIN (" + this.sqlJ + ") ts ";
this.sqlF += "ON n.nlp100_date = ts.nlp100_date ";
this.sqlF += "AND n.nlp100_seq = ts.nlp100_seq ";
this.sqlF += "AND (n.nlp131_np = ts.structure1 OR n.nlp131_np = ts.structure2) ";
this.sqlF += "INNER JOIN catpd.nlp120 s ";
this.sqlF += "ON n.nlp100_date = s.nlp100_date ";
this.sqlF += "AND n.nlp100_seq = s.nlp100_seq ";
this.sqlF += "AND n.nlp110_seq = s.nlp110_seq ";
this.sqlF += "AND n.nlp120_seq = s.nlp120_seq ";

this.sqlO = "ORDER BY n.nlp100_date, n.nlp100_seq, n.nlp110_seq, n.nlp120_seq";

this.ssql = this.sqlS + this.sqlF + this.sqlO;
this.sqlh = new SQLHandler(this.ssql);
this.rs = this.sqlh.getRSet();

try {
int pindex = 0; // Primary Key of Paragraph of Transformation Document
int sindex = 0; // Primary Key of Sentence of Transformation Document

//Primary Keys
int cpindex = 0;     // Primary Key of Paragraph of Concept Functions
int csindex = 0;     // Primary Key of Sentence of Concept Functions
String sentence = "";
String field = "";

while (this.rs.next()) {
//Get the paragraph index of concept function
field = "nlp110_seq";
cpindex = this.rs.getInt(field);
//Get each paragraph index
pindex = this.getPIndex(date, serial, cindex, cpindex);

//Get the sentence index of concept function
field = "nlp120_seq";
csindex = this.rs.getInt(field);
//Get each sentence index
sindex = this.getSIndex(pindex, date, serial, csindex);
//Set Each Sentence
field = "nlp120_sentence";
sentence = this.rs.getString(field);
this.setSentence(pindex, sindex, cindex, csindex, sentence);

//Update the previous paragraph
this.updateParagraph(pindex);
}
} catch (SQLException e) {
System.out.println("Transformer.importDesign() SQL Error..." + e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.out.println("Transformer.importDesign() Error..." + e.getMessage());
e.printStackTrace();
} finally {
this.sqlh.destroy();
}
}

/**
* isParagraphExist()
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param cpindex: Functional Concept Primary Key of Paragraph Serial
* @return
*/
private boolean isParagraphExist(String date, int serial, int cpindex) {
boolean isExist = false;

//Paragraph Transformation History Table
String table = "catpd.nlp11t";
String [] fields = {"nlp100_date", "nlp100_seq", "nlp10t_date", "nlp10t_seq", "nlp11t_seq"};
String [] values = {this.date, String.valueOf(this.serial),
date, String.valueOf(serial), String.valueOf(cpindex)};
boolean [] isNumber = {false, true, false, true, true};
int iNoOfKeys = fields.length;

//Check if each paragraph exists in the database
DataHelper helper = new DataHelper();
isExist = helper.isExist(table, iNoOfKeys, fields, values, isNumber);

return isExist;
}

/**
* getPIndex()
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param cindex: Functional Concept Primary Key of Concept Serial
* @param cpindex: Functional Concept Primary Key of Paragraph Serial
* @return: Paragraph Index
*/
private int getPIndex(String date, int serial, int cindex, int cpindex) {
int pindex = 0;

DataHelper helper = new DataHelper();

//Check if each paragraph exists in the database
boolean isExist = this.isParagraphExist(date, serial, cpindex);

if (isExist) {
//Get the existing paragraph index
String table = "catpd.nlp11t";
String field = "nlp110_seq";
String [] fields = {"nlp100_date", "nlp100_seq", "nlp10t_date", "nlp10t_seq", "nlp11t_seq"};
String [] values = {this.date, String.valueOf(this.serial),
date, String.valueOf(serial), String.valueOf(cpindex)};
boolean [] isNumber = {false, true, false, true, true};

pindex = Integer.parseInt(helper.getString(table, field, fields, values, isNumber));
} else {
//Get a new paragraph index
String table = "catpd.nlp110";
String field = "nlp110_seq";
String [] fields = {"nlp100_date", "nlp100_seq"};
String [] values = {this.date, String.valueOf(this.serial)};
boolean [] isNumber = {false, true};
int iNoOfKeys = fields.length;

//Create a new paragraph record by obtaining the paragraph serial number
pindex = helper.getMaxSerial(field, table, iNoOfKeys, fields, values, isNumber);
//Create a transformation history record of the paragraph
this.setParagraph(pindex, date, serial, cindex, cpindex);
}

return pindex;
}

/**
* setParagraph()
* @param pindex: a new paragraph index
* @param cdate: Functional Concept Primary Key of Document Date
* @param cserial: Functional Concept Primary Key of Document Serial
* @param cindex: Functional Concept Primary Key of Concept Serial
* @param cpindex: Functional Concept Primary Key of Paragraph Index
*/
private void setParagraph(int pindex, String cdate, int cserial, int cindex, int cpindex) {
//Create a transformation history record of Paragraph, pindex, only
this.sqlI = "INSERT INTO catpd.nlp11t ";
this.sqlI += "(nlp100_date, nlp100_seq, nlp110_seq, nlp10t_date, nlp10t_seq, tpd120_seq,
nlp11t_seq) ";

this.sqlV = "VALUES (";
this.sqlV += "'" + this.date + "',";
this.sqlV += String.valueOf(this.serial) + ",";
this.sqlV += String.valueOf(pindex) + ",";
this.sqlV += "'" + cdate + "',";
this.sqlV += String.valueOf(cserial) + ",";
this.sqlV += String.valueOf(cindex) + ",";
this.sqlV += String.valueOf(cpindex);
this.sqlV += ")";

this.ssql = this.sqlI + this.sqlV;
SQLHandler handler = new SQLHandler(this.ssql);
handler.destroy();
}

/**
* isSentenceExist()
* @param pindex: Paragraph Index
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param csindex: Functional Concept Primary Key of Sentence Serial
* @return
*/
private boolean isSentenceExist(int pindex, String date, int serial, int csindex) {
boolean isExist = false;

//Sentence Transformation History Table
String table = "catpd.nlp12t";
String [] fields = {"nlp100_date", "nlp100_seq", "nlp110_seq", "nlp12t_seq"};
String [] values = {this.date, String.valueOf(this.serial),
String.valueOf(pindex), String.valueOf(csindex)};
boolean [] isNumber = {false, true, true, true};
int iNoOfKeys = fields.length;

//Check if each sentence exists in the database
DataHelper helper = new DataHelper();
isExist = helper.isExist(table, iNoOfKeys, fields, values, isNumber);

return isExist;
}

/**
* getSIndex()
* @param pindex: Paragraph Index
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param csindex: Functional Concept Primary Key of Sentence Serial
* @return: Sentence Index
*/
private int getSIndex(int pindex, String date, int serial, int csindex) {
int sindex = 0;

DataHelper helper = new DataHelper();

//Check if each sentence exists in the database
boolean isExist = this.isSentenceExist(pindex, date, serial, csindex);

if (isExist) {
//Get the existing sentence index
String table = "catpd.nlp12t";
String field = "nlp120_seq";
String [] fields = {"nlp100_date", "nlp100_seq", "nlp110_seq", "nlp12t_seq"};
String [] values = {this.date, String.valueOf(this.serial),
String.valueOf(pindex), String.valueOf(csindex)};
boolean [] isNumber = {false, true, true, true};

sindex = Integer.parseInt(helper.getString(table, field, fields, values, isNumber));
} else {
//Get a new sentence index
String table = "catpd.nlp120";
String field = "nlp120_seq";
String [] fields = {"nlp100_date", "nlp100_seq", "nlp110_seq"};
String [] values = {this.date, String.valueOf(this.serial), String.valueOf(pindex)};
boolean [] isNumber = {false, true, true};
int iNoOfKeys = fields.length;

//Create a new paragraph record by obtaining the paragraph serial number
sindex = helper.getMaxSerial(field, table, iNoOfKeys, fields, values, isNumber);
}

return sindex;
}
```

```
/**
* setSentence()
* @param pindex: a new paragraph index
* @param sindex: a new sentence index
* @param cindex: Functional Concept Primary Key of Concept Serial
* @param csindex: Functional Concept Primary Key of Sentence Index
* @param sentence: Sentence of Functional Concept
*/
private void setSentence(int pindex, int sindex, int cindex, int csindex, String sentence) {
DataHelper helper = new DataHelper();

this.sqlU = "UPDATE catpd.nlp120 ";
this.sqlS = "SET nlp120_sentence = '" + sentence + "' ";
this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND nlp110_seq = " + String.valueOf(pindex) + " ";
this.sqlW += "AND nlp120_seq = " + String.valueOf(sindex) + " ";

this.ssql = this.sqlU + this.sqlS + this.sqlW;
SQLHandler handler = new SQLHandler(this.ssql);
handler.destroy();

boolean isExist = false;

//Sentence Transformation History Table
String table = "catpd.nlp12t";
String [] fields = {"nlp100_date", "nlp100_seq", "nlp110_seq", "nlp120_seq"};
String [] values = {this.date, String.valueOf(this.serial), String.valueOf(pindex),
String.valueOf(sindex)};
boolean [] isNumber = {false, true, true, true};
int iNoOfKeys = fields.length;

//Check if the sentence history exists in the database
isExist = helper.isExist(table, iNoOfKeys, fields, values, isNumber);

if (!isExist) {
this.sqlI = "INSERT INTO catpd.nlp12t ";
this.sqlI += "(nlp100_date, nlp100_seq, nlp110_seq, nlp120_seq, tpd120_seq, nlp12t_seq) ";

this.sqlV = "VALUES (";
this.sqlV += "'" + this.date + "',";
this.sqlV += String.valueOf(this.serial) + ",";
this.sqlV += String.valueOf(pindex) + ",";
this.sqlV += String.valueOf(sindex) + ",";
this.sqlV += String.valueOf(cindex) + ",";
this.sqlV += String.valueOf(csindex);
this.sqlV += ")";

this.ssql = this.sqlI + this.sqlV;
handler = new SQLHandler(this.ssql);
handler.destroy();
}
}

/**
* updateParagraph()
* @param pindex: a new paragraph index
*/
private void updateParagraph(int pindex) {
//Obtain Sentence Index of Duplicated Sentences
this.sqlJ = "SELECT t1.sindex ";
this.sqlJ += "FROM (";

this.sqlJ += "SELECT nlp120_sentence, ";
this.sqlJ += "CONCAT(CAST(nlp110_seq AS CHAR), '-', CAST(nlp120_seq AS CHAR)) AS
sindex ";
this.sqlJ += "FROM catpd.nlp120 ";
this.sqlJ += "WHERE nlp100_date = '" + this.date + "' ";
this.sqlJ += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlJ += "AND (nlp120_sentence <> '' AND nlp120_sentence IS NOT NULL)";

this.sqlJ += ") t1 LEFT JOIN (";

this.sqlJ += "SELECT nlp120_sentence, ";
this.sqlJ += "MIN(CONCAT(CAST(nlp110_seq AS CHAR), '-', CAST(nlp120_seq AS CHAR))) AS
minindex ";
this.sqlJ += "FROM catpd.nlp120 ";
this.sqlJ += "WHERE nlp100_date = '" + this.date + "' ";
this.sqlJ += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlJ += "AND (nlp120_sentence <> '' AND nlp120_sentence IS NOT NULL) ";
this.sqlJ += "GROUP BY nlp100_date, nlp100_seq, nlp120_sentence";

this.sqlJ += ") t2 ";
this.sqlJ += "ON t1.nlp120_sentence = t2.nlp120_sentence ";
this.sqlJ += "AND t1.sindex = t2.minindex ";

this.sqlJ += "WHERE t2.minindex IS NULL";    // Excluding the Minimum Index

//Update the duplicated sentence with null
this.sqlU = "UPDATE catpd.nlp120 ";
this.sqlS = "SET nlp120_sentence = NULL ";
this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND CONCAT(CAST(nlp110_seq AS CHAR), '-', CAST(nlp120_seq AS
CHAR)) IN (" + this.sqlJ + ") ";

this.ssql = this.sqlU + this.sqlS + this.sqlW;
SQLHandler handler = new SQLHandler(this.ssql);
handler.destroy();

//Update Paragraph
this.sqlS = "SELECT nlp100_date, nlp100_seq, nlp110_seq, ";
this.sqlS += "GROUP_CONCAT(nlp120_sentence SEPARATOR ' ') AS nlp110_paragraph, ";
this.sqlS += "COUNT(*) AS nlp110_noofsentences ";
this.sqlF = "FROM catpd.nlp120 ";
this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND nlp110_seq = " + String.valueOf(pindex) + " ";
this.sqlW += "AND (nlp120_sentence <> '' OR nlp120_sentence IS NOT NULL) ";
this.sqlG = "GROUP BY nlp100_date, nlp100_seq, nlp110_seq";

this.ssql = this.sqlS + this.sqlF + this.sqlW + this.sqlG;
handler = new SQLHandler(this.ssql);
ResultSet rset = handler.getRSet();

try {
if (rset.next()) {
DataHelper helper = new DataHelper();

String table = "catpd.nlp110";
String [] keyfields = {"nlp100_date", "nlp100_seq", "nlp110_seq"};
String [] keyvalues = {this.date, String.valueOf(this.serial), String.valueOf(pindex)};
boolean [] isNumber = {false, true, true};

String field = "nlp110_paragraph";
String value = rset.getString(field);
boolean isNumeric = false;
helper.updateField(table, field, value, isNumeric, keyfields, keyvalues, isNumber);

field = "nlp110_noofsentences";
value = rset.getString(field);
isNumeric = true;
helper.updateField(table, field, value, isNumeric, keyfields, keyvalues, isNumber);
}
} catch (SQLException e) {
System.out.println("Transformer.updateParagraph() SQL Error..." + e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.out.println("Transformer.updateParagraph() Error..." + e.getMessage());
e.printStackTrace();
} finally {
handler.destroy();
}
}

/**
* clearWords()
*/
private void clearWords() {
String [] sTable = {"nlp130", "nlp131", "nlp132"};

this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";

for (int i = sTable.length; i > 0; i--) {
this.sqlD = "DELETE FROM catpd." + sTable[i-1] + " ";
this.ssql = this.sqlD + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();
}
}

/**
* setWords()
*/
private void setWords() {
this.clearWords();

String subquery = "";

//Update the number of words in sentences
this.sqlU = "UPDATE catpd.nlp120 sw ";

//SUB-QUERY: Retrieve all the primary keys of sentences before/after transformation
//Retrieve the number of words in each sentence with the primary keys
this.sqlS = "SELECT s.nlp100_date, s.nlp100_seq, s.nlp110_seq, s.nlp120_seq, ";
this.sqlS += "s.nlp10t_date, s.nlp10t_seq, s.nlp11t_seq, s.nlp12t_seq, ";
this.sqlS += "ns.nlp120_sentence, ns.nlp120_noofwords, s.tpd120_seq ";

//SUB-SUB-QUERY: To retrieve primary keys of each sentence before transformation
//Sub-sub-query will be reused later in this method.
subquery = "SELECT tp.nlp100_date, tp.nlp100_seq, tp.nlp110_seq, ts.nlp120_seq, ";
subquery += "tp.nlp10t_date, tp.nlp10t_seq, tp.nlp11t_seq, ts.nlp12t_seq, tp.tpd120_seq ";
subquery += "FROM (";
subquery += "SELECT * FROM catpd.nlp11t tp ";
subquery += "WHERE nlp100_date = '" + this.date + "' ";
subquery += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
subquery += ") tp ";
subquery += "INNER JOIN catpd.nlp12t ts ";
subquery += "ON tp.nlp100_date = ts.nlp100_date ";
subquery += "AND tp.nlp100_seq = ts.nlp100_seq ";
subquery += "AND tp.nlp110_seq = ts.nlp110_seq ";

this.sqlF = "FROM (" + subquery + ") s INNER JOIN catpd.nlp120 ns ";
this.sqlF += "ON s.nlp10t_date = ns.nlp100_date ";
this.sqlF += "AND s.nlp10t_seq = ns.nlp100_seq ";
this.sqlF += "AND s.nlp11t_seq = ns.nlp110_seq ";
this.sqlF += "AND s.nlp12t_seq = ns.nlp120_seq ";

this.sqlC = this.sqlS + this.sqlF;

//Relational condition for UPDATE AND SUB-QUERY
this.sqlJ = "INNER JOIN (" + this.sqlC + ") tw ";
this.sqlJ += "ON sw.nlp100_date = tw.nlp100_date ";
this.sqlJ += "AND sw.nlp100_seq = tw.nlp100_seq ";
this.sqlJ += "AND sw.nlp110_seq = tw.nlp110_seq ";
this.sqlJ += "AND sw.nlp120_seq = tw.nlp120_seq ";

this.sqlS = "SET sw.nlp120_noofwords = tw.nlp120_noofwords ";

this.sqlW = "WHERE sw.nlp100_date = '" + this.date + "' ";
this.sqlW += "AND sw.nlp100_seq = " + String.valueOf(this.serial);

this.ssql = this.sqlU + this.sqlJ + this.sqlS + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();

//Import the corresponding words from the concept document
this.sqlI = "INSERT INTO catpd.nlp130 (nlp100_date, nlp100_seq, nlp110_seq, nlp120_seq, ";
this.sqlI += "nlp130_seq, nlp130_word, nlp900_tag, nlp130_stem) ";

this.sqlS = "SELECT s.nlp100_date, s.nlp100_seq, s.nlp110_seq, s.nlp120_seq, ";
this.sqlS += "nw.nlp130_seq, nw.nlp130_word, nw.nlp900_tag, nw.nlp130_stem ";
this.sqlF = "FROM (" + subquery + ") s INNER JOIN catpd.nlp130 nw ";
this.sqlF += "ON s.nlp10t_date = nw.nlp100_date ";
this.sqlF += "AND s.nlp10t_seq = nw.nlp100_seq ";
this.sqlF += "AND s.nlp11t_seq = nw.nlp110_seq ";
this.sqlF += "AND s.nlp12t_seq = nw.nlp120_seq ";
this.sqlO = "ORDER BY s.nlp100_date, s.nlp100_seq, s.nlp110_seq, s.nlp120_seq,
nw.nlp130_seq";

this.ssql = this.sqlI + this.sqlS + this.sqlF + this.sqlO;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();

//Import noun phrases related to the words
this.sqlI = "INSERT INTO catpd.nlp131 ";
this.sqlI += "(nlp100_date, nlp100_seq, nlp110_seq, nlp120_seq, nlp131_seq, nlp131_np) ";

this.sqlS = "SELECT DISTINCT s.nlp100_date, s.nlp100_seq, s.nlp110_seq, s.nlp120_seq, ";
this.sqlS += "np.nlp131_seq, np.nlp131_np ";
this.sqlF = "FROM (" + subquery + ") s INNER JOIN catpd.nlp130 nw ";
this.sqlF += "ON s.nlp10t_date = nw.nlp100_date ";
this.sqlF += "AND s.nlp10t_seq = nw.nlp100_seq ";
this.sqlF += "AND s.nlp11t_seq = nw.nlp110_seq ";
this.sqlF += "AND s.nlp12t_seq = nw.nlp120_seq ";
this.sqlF += "INNER JOIN catpd.nlp132 n ";
this.sqlF += "ON s.nlp10t_date = n.nlp100_date ";
this.sqlF += "AND s.nlp10t_seq = n.nlp100_seq ";
this.sqlF += "AND s.nlp11t_seq = n.nlp110_seq ";
this.sqlF += "AND s.nlp12t_seq = n.nlp120_seq ";
this.sqlF += "AND nw.nlp130_seq = n.nlp130_seq ";
this.sqlF += "INNER JOIN catpd.nlp131 np ";
this.sqlF += "ON s.nlp10t_date = np.nlp100_date ";
this.sqlF += "AND s.nlp10t_seq = np.nlp100_seq ";
this.sqlF += "AND s.nlp11t_seq = np.nlp110_seq ";
this.sqlF += "AND s.nlp12t_seq = np.nlp120_seq ";
this.sqlF += "AND n.nlp131_seq = np.nlp131_seq ";
this.sqlO = "ORDER BY s.nlp100_date, s.nlp100_seq, s.nlp110_seq, s.nlp120_seq,
n.nlp131_seq";

this.ssql = this.sqlI + this.sqlS + this.sqlF + this.sqlO;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();

//Import nouns related to the words
this.sqlI = "INSERT INTO catpd.nlp132 ";
this.sqlI += "(nlp100_date, nlp100_seq, nlp110_seq, nlp120_seq, nlp131_seq, nlp130_seq) ";

this.sqlS = "SELECT s.nlp100_date, s.nlp100_seq, s.nlp110_seq, s.nlp120_seq, ";
this.sqlS += "n.nlp131_seq, n.nlp130_seq ";
this.sqlF = "FROM (" + subquery + ") s INNER JOIN catpd.nlp130 nw ";
this.sqlF += "ON s.nlp10t_date = nw.nlp100_date ";
this.sqlF += "AND s.nlp10t_seq = nw.nlp100_seq ";
this.sqlF += "AND s.nlp11t_seq = nw.nlp110_seq ";
this.sqlF += "AND s.nlp12t_seq = nw.nlp120_seq ";
this.sqlF += "INNER JOIN catpd.nlp132 n ";
this.sqlF += "ON s.nlp10t_date = n.nlp100_date ";
this.sqlF += "AND s.nlp10t_seq = n.nlp100_seq ";
this.sqlF += "AND s.nlp11t_seq = n.nlp110_seq ";
this.sqlF += "AND s.nlp12t_seq = n.nlp120_seq ";
this.sqlF += "AND nw.nlp130_seq = n.nlp130_seq ";
this.sqlO = "ORDER BY s.nlp100_date, s.nlp100_seq, s.nlp110_seq, s.nlp120_seq, ";
this.sqlO += "n.nlp131_seq, n.nlp130_seq";

this.ssql = this.sqlI + this.sqlS + this.sqlF + this.sqlO;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();
}

/**
* setFunctions
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param cindex: Functional Concept Primary Key of Concept Serial
*/
private void setFunctions(String date, int serial, int cindex) {
this.clearFunctions(date, serial, cindex);

//Import Functional Concept
DataHelper helper = new DataHelper();

String table = "catpd.tpd120";
String field = "tpd120_seq";

String [] fields1 = {"nlp100_date", "nlp100_seq"};
String [] values1 = {this.date, String.valueOf(this.serial)};
boolean [] isNumber1 = {false, true};
int iNoOfKeys1 = fields1.length;

//Import functional concept after transformation
int new_cindex = helper.getMaxSerial(field, table, iNoOfKeys1, fields1, values1, isNumber1);

this.sqlU = "UPDATE catpd.tpd120 tc ";

this.sqlS = "SELECT '" + this.date + "' AS nlp100_date, ";
this.sqlS += String.valueOf(this.serial) + " AS nlp100_seq, ";
this.sqlS += String.valueOf(new_cindex) + " AS tpd120_seq, ";
this.sqlS += "fc.tpd120_fnconcept ";
this.sqlF = "FROM catpd.tpd120 fc ";
this.sqlW = "WHERE fc.nlp100_date = '" + date + "' ";
this.sqlW += "AND fc.nlp100_seq = " + String.valueOf(serial) + " ";
this.sqlW += "AND fc.tpd120_seq = " + String.valueOf(cindex) + " ";

this.sqlJ = "INNER JOIN (" + this.sqlS + this.sqlF + this.sqlW + ") fc ";
this.sqlJ += "ON tc.nlp100_date = fc.nlp100_date ";
this.sqlJ += "AND tc.nlp100_seq = fc.nlp100_seq ";
this.sqlJ += "AND tc.tpd120_seq = fc.tpd120_seq ";

this.sqlS = "SET tc.tpd120_fnconcept = fc.tpd120_fnconcept ";
this.sqlW = "WHERE tc.nlp100_date = '" + this.date + "' ";
this.sqlW += "AND tc.nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND tc.tpd120_seq = " + String.valueOf(new_cindex) + " ";

this.ssql = this.sqlU + this.sqlJ + this.sqlS + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();

//Create a transformation record
this.sqlI = "INSERT INTO catpd.tpd12t ";
this.sqlI += "(nlp100_date, nlp100_seq, tpd120_seq, nlp10t_date, nlp10t_seq, tpd12t_seq) ";

this.sqlS = "SELECT '" + this.date + "' AS nlp100_date, ";
this.sqlS += String.valueOf(this.serial) + " AS nlp100_seq, ";
this.sqlS += String.valueOf(new_cindex) + " AS tpd120_seq, ";
this.sqlS += "fc.nlp100_date AS nlp10t_date, fc.nlp100_seq AS nlp10t_seq, fc.tpd120_seq AS
tpd12t_seq ";
this.sqlF = "FROM catpd.tpd120 fc ";
this.sqlW = "WHERE fc.nlp100_date = '" + date + "' ";
this.sqlW += "AND fc.nlp100_seq = " + String.valueOf(serial) + " ";
this.sqlW += "AND fc.tpd120_seq = " + String.valueOf(cindex) + " ";

this.ssql = this.sqlI + this.sqlS + this.sqlF + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();

//Import Functions
this.sqlS = "SELECT cf.nlp100_date, cf.nlp100_seq, cf.tpd121_seq, cf.tpd121_primary, ";
this.sqlS += "cf.tpd100_seq, f.tpd100_function ";

this.sqlJ = "SELECT * ";
this.sqlJ += "FROM catpd.tpd120 ";
this.sqlJ += "WHERE nlp100_date = '" + date + "' ";
this.sqlJ += "AND nlp100_seq = " + String.valueOf(serial) + " ";
this.sqlJ += "AND tpd120_seq = " + String.valueOf(cindex);

this.sqlF = "FROM (" + this.sqlJ + ") fc INNER JOIN catpd.tpd121 cf ";
this.sqlF += "ON fc.nlp100_date = cf.nlp100_date ";
this.sqlF += "AND fc.nlp100_seq = cf.nlp100_seq ";
this.sqlF += "AND fc.tpd120_seq = cf.tpd120_seq ";
this.sqlF += "INNER JOIN catpd.tpd100 f ";
this.sqlF += "ON fc.nlp100_date = f.nlp100_date ";
this.sqlF += "AND fc.nlp100_seq = f.nlp100_seq ";
this.sqlF += "AND cf.tpd100_seq = f.tpd100_seq ";

this.ssql = this.sqlS + this.sqlF;
this.sqlh = new SQLHandler(this.ssql);
this.rs = this.sqlh.getRSet();

try {
while (this.rs.next()) {
int old_findex = this.rs.getInt("tpd100_seq");
String function = this.rs.getString("tpd100_function");
int cfindex = this.rs.getInt("tpd121_seq");
boolean primary = this.rs.getBoolean("tpd121_primary");

//Check if each function exists already
table = "catpd.tpd100";

String [] fields2 = {"nlp100_date", "nlp100_seq", "tpd100_function"};
String [] values2 = {this.date, String.valueOf(this.serial), function};
boolean [] isNumber2 = {false, true, false};
int iNoOfKeys2 = fields2.length;

boolean isExist = helper.isExist(table, iNoOfKeys2, fields2, values2, isNumber2);

table = "catpd.tpd100";
field = "tpd100_seq";
int findex = 0;

if (isExist) {
//If a function exists already, obtain the serial number
String idx = helper.getString(table, field, fields2, values2, isNumber2);
findex = Integer.parseInt(idx);
} else {
String [] fields3 = {"nlp100_date", "nlp100_seq"};
String [] values3 = {this.date, String.valueOf(this.serial)};
boolean [] isNumber3 = {false, true};
int iNoOfKeys3 = fields3.length;

//Import each function
findex = helper.getMaxSerial(field, table, iNoOfKeys3, fields3, values3, isNumber3);
this.sqlU = "UPDATE catpd.tpd100 ";
this.sqlS = "SET tpd100_function = '" + function + "' ";
this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND tpd100_seq = " + String.valueOf(findex);

this.ssql = this.sqlU + this.sqlS + this.sqlW;
SQLHandler handler = new SQLHandler(this.ssql);
handler.destroy();
}

//Import each function related to functional concept
this.sqlI = "INSERT INTO catpd.tpd121 ";
this.sqlI += "(nlp100_date, nlp100_seq, tpd120_seq, tpd121_seq, tpd121_primary, tpd100_seq)
";
this.sqlV = "VALUES (";
this.sqlV += "'" + this.date + "',";
this.sqlV += String.valueOf(this.serial) + ",";
this.sqlV += String.valueOf(new_cindex) + ",";
this.sqlV += String.valueOf(cfindex) + ",";
this.sqlV += String.valueOf(primary) + ",";
this.sqlV += String.valueOf(findex);
this.sqlV += ")";

this.ssql = this.sqlI + this.sqlV;
SQLHandler handler = new SQLHandler(this.ssql);
handler.destroy();

//Import sentences that belong to each function
this.sqlI = "INSERT INTO catpd.tpd110 ";
this.sqlI += "(nlp100_date, nlp100_seq, tpd100_seq, tpd110_seq, nlp110_seq, nlp120_seq) ";

this.sqlS = "SELECT pt.nlp100_date, pt.nlp100_seq, ";
this.sqlS += String.valueOf(findex) + " AS tpd100_seq, fs.tpd110_seq, ";
this.sqlS += "pt.nlp110_seq, st.nlp120_seq ";

this.sqlJ = "SELECT * FROM catpd.tpd120 ";
this.sqlJ += "WHERE nlp100_date = '" + date + "' ";
this.sqlJ += "AND nlp100_seq = " + String.valueOf(serial) + " ";
this.sqlJ += "AND tpd120_seq = " + String.valueOf(cindex);

this.sqlF = "FROM (" + this.sqlJ + ") fc ";

this.sqlJ = "SELECT * FROM catpd.tpd121 ";
this.sqlJ += "WHERE nlp100_date = '" + date + "' ";
this.sqlJ += "AND nlp100_seq = " + String.valueOf(serial) + " ";
this.sqlJ += "AND tpd120_seq = " + String.valueOf(cindex) + " ";
this.sqlJ += "AND tpd100_seq = " + String.valueOf(old_findex);

this.sqlF += "INNER JOIN (" + this.sqlJ + ") cf ";
this.sqlF += "ON fc.nlp100_date = cf.nlp100_date ";
this.sqlF += "AND fc.nlp100_seq = cf.nlp100_seq ";
this.sqlF += "AND fc.tpd120_seq = cf.tpd120_seq ";
this.sqlF += "INNER JOIN catpd.tpd100 f ";
this.sqlF += "ON fc.nlp100_date = f.nlp100_date ";
this.sqlF += "AND fc.nlp100_seq = f.nlp100_seq ";
this.sqlF += "AND cf.tpd100_seq = f.tpd100_seq ";
this.sqlF += "INNER JOIN catpd.tpd110 fs ";
this.sqlF += "ON fc.nlp100_date = fs.nlp100_date ";
this.sqlF += "AND fc.nlp100_seq = fs.nlp100_seq ";
this.sqlF += "AND cf.tpd100_seq = fs.tpd100_seq ";
this.sqlF += "INNER JOIN catpd.nlp11t pt ";
this.sqlF += "ON fc.nlp100_date = pt.nlp10t_date ";
this.sqlF += "AND fc.nlp100_seq = pt.nlp10t_seq ";
this.sqlF += "AND fs.nlp110_seq = pt.nlp11t_seq ";
this.sqlF += "INNER JOIN catpd.nlp12t st ";
this.sqlF += "ON pt.nlp100_date = st.nlp100_date ";
this.sqlF += "AND pt.nlp100_seq = st.nlp100_seq ";
this.sqlF += "AND pt.nlp110_seq = st.nlp110_seq ";
this.sqlF += "AND fs.nlp120_seq = st.nlp12t_seq ";
this.sqlF += "LEFT JOIN catpd.tpd110 z ";
this.sqlF += "ON pt.nlp100_date = z.nlp100_date ";
this.sqlF += "AND pt.nlp100_seq = z.nlp100_seq ";
this.sqlF += "AND z.tpd100_seq = " + String.valueOf(findex) + " ";
this.sqlF += "AND fs.tpd110_seq = z.tpd110_seq ";

this.sqlW = "WHERE z.tpd110_seq IS NULL";

this.ssql = this.sqlI + this.sqlS + this.sqlF + this.sqlW;
handler = new SQLHandler(this.ssql);
handler.destroy();
}
} catch (SQLException e) {
System.out.println("Transformer.setFunctions() SQL Error..." + e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.out.println("Transformer.setFunctions() Error..." + e.getMessage());
e.printStackTrace();
} finally {
this.sqlh.destroy();
}
}
```

```
/**
* clearFunctions()
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param cindex: Functional Concept Primary Key of Concept Serial
*/
private void clearFunctions(String date, int serial, int cindex) {
//Obtain primary keys of functional concept after transformation
this.sqlS = "SELECT nlp100_date, nlp100_seq, tpd120_seq ";
this.sqlF = "FROM catpd.tpd12t ";
this.sqlW = "WHERE nlp10t_date = '" + date + "' ";
this.sqlW += "AND nlp10t_seq = " + String.valueOf(serial) + " ";
this.sqlW += "AND tpd12t_seq = " + String.valueOf(cindex) + " ";

this.ssql = this.sqlS + this.sqlF + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.rs = this.sqlh.getRSet();

String [] spk = new String[3];
for (int index = 0; index < spk.length; index++) {
spk[index] = "";
}

try {
if (this.rs.next()) {
for (int index = 0; index < spk.length; index++) {
spk[index] = this.rs.getString(index+1);
}
}
} catch (SQLException e) {
System.out.println("Transformer.clearFunctions() SQL Error (1) ..." + e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.out.println("Transformer.clearFunctions() Error (1) ..." + e.getMessage());
e.printStackTrace();
} finally {
this.sqlh.destroy();
}

//If any function related to the functional concept exists in the database, delete it.
if (!spk[0].equals("")) {
String functions = "";

this.sqlS = "SELECT GROUP_CONCAT(CAST(f.tpd100_seq AS CHAR) SEPARATOR ',') AS
functions ";
```

```
this.sqlJ = "SELECT DISTINCT nlp100_date, nlp100_seq, tpd100_seq ";
this.sqlJ += "FROM catpd.tpd121 ";
this.sqlJ += "WHERE nlp100_date = '" + spk[0] + "' ";
this.sqlJ += "AND nlp100_seq = " + spk[1] + " ";
this.sqlJ += "AND tpd120_seq = " + spk[2] + " ";

this.sqlF = "FROM (" + this.sqlJ + ") f ";
this.sqlJ = this.sqlJ.replace("AND tpd120_seq = ", "AND tpd120_seq <> ");
this.sqlF += "LEFT JOIN (" + this.sqlJ + ") o ";
this.sqlF += "ON f.nlp100_date = o.nlp100_date ";
this.sqlF += "AND f.nlp100_seq = o.nlp100_seq ";
this.sqlF += "AND f.tpd100_seq = o.tpd100_seq ";

this.sqlW = "WHERE o.tpd100_seq IS NULL ";
this.sqlG = "GROUP BY f.nlp100_date, f.nlp100_seq ";

this.ssql = this.sqlS + this.sqlF + this.sqlW + this.sqlG;
this.sqlh = new SQLHandler(this.ssql);
this.rs = this.sqlh.getRSet();

try {
if (this.rs.next()) {
functions = this.rs.getString("functions");
}
} catch (SQLException e) {
System.out.println("Transformer.clearFunctions() SQL Error (2) ..." + e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.out.println("Transformer.clearFunctions() Error (2) ..." + e.getMessage());
e.printStackTrace();
} finally {
this.sqlh.destroy();
}

if (!functions.equals("")) {
//Delete Functions
this.sqlW = "WHERE nlp100_date = '" + spk[0] + "' ";
this.sqlW += "AND nlp100_seq = " + spk[1] + " ";
this.sqlW += "AND tpd100_seq IN (" + functions + ") ";

String [] sTable = {"tpd100", "tpd110"};

for (int i = sTable.length; i > 0; i--) {
this.sqlD = "DELETE FROM catpd." + sTable[i-1] + " ";
this.ssql = this.sqlD + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();
}
}

//Delete Functional Concept
this.sqlW = "WHERE nlp100_date = '" + spk[0] + "' ";
this.sqlW += "AND nlp100_seq = " + spk[1] + " ";
this.sqlW += "AND tpd120_seq = " + spk[2] + " ";

String [] sTable = {"tpd120", "tpd121", "tpd12t"};

for (int i = sTable.length; i > 0; i--) {
this.sqlD = "DELETE FROM catpd." + sTable[i-1] + " ";
this.ssql = this.sqlD + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();
}
}
}

/**
* setStructures
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param cindex: Functional Concept Primary Key of Concept Serial
*/
private void setStructures(String date, int serial) {
this.clearStructures(date, serial);

//Retrieve relevant structures before transformation
this.sqlS = "SELECT DISTINCT s.nlp100_date, s.nlp100_seq, ts.tpd200_seq,
ts.tpd200_structure ";

this.sqlJ = "SELECT * FROM catpd.nlp120 ";
this.sqlJ += "WHERE nlp100_date = '" + this.date + "' ";
this.sqlJ += "AND nlp100_seq = " + String.valueOf(this.serial);

this.sqlF = "FROM (" + this.sqlJ + ") t INNER JOIN catpd.nlp11t pt ";
this.sqlF += "ON t.nlp100_date = pt.nlp100_date ";
this.sqlF += "AND t.nlp100_seq = pt.nlp100_seq ";
this.sqlF += "AND t.nlp110_seq = pt.nlp110_seq ";
this.sqlF += "INNER JOIN catpd.nlp12t st ";
this.sqlF += "ON t.nlp100_date = st.nlp100_date ";
this.sqlF += "AND t.nlp100_seq = st.nlp100_seq ";
this.sqlF += "AND t.nlp110_seq = st.nlp110_seq ";
this.sqlF += "AND t.nlp120_seq = st.nlp120_seq ";
this.sqlF += "INNER JOIN catpd.nlp120 s ";
this.sqlF += "ON pt.nlp10t_date = s.nlp100_date ";
this.sqlF += "AND pt.nlp10t_seq = s.nlp100_seq ";
this.sqlF += "AND pt.nlp11t_seq = s.nlp110_seq ";
this.sqlF += "AND st.nlp12t_seq = s.nlp120_seq ";
this.sqlF += "INNER JOIN catpd.nlp131 np ";
this.sqlF += "ON s.nlp100_date = np.nlp100_date ";
this.sqlF += "AND s.nlp100_seq = np.nlp100_seq ";
this.sqlF += "AND s.nlp110_seq = np.nlp110_seq ";
this.sqlF += "AND s.nlp120_seq = np.nlp120_seq ";
this.sqlF += "INNER JOIN catpd.tpd200 ts ";
this.sqlF += "ON np.nlp100_date = ts.nlp100_date ";
this.sqlF += "AND np.nlp100_seq = ts.nlp100_seq ";
this.sqlF += "AND np.nlp131_np = ts.tpd200_structure ";

this.sqlO = "ORDER BY s.nlp100_date, s.nlp100_seq, ts.tpd200_seq";
this.ssql = this.sqlS + this.sqlF + this.sqlO;
SQLHandler handler = new SQLHandler(this.ssql);
ResultSet rset = handler.getRSet();

try {
while (rset.next()) {
String structure = rset.getString("tpd200_structure");

DataHelper helper = new DataHelper();
int sindex, slog_index;

//Check if each structure exists
String table = "catpd.tpd200";
String field = "tpd200_seq";

String [] fields1 = {"nlp100_date", "nlp100_seq", "tpd200_structure"};
String [] values1 = {this.date, String.valueOf(this.serial), structure};
boolean [] isNumber1 = {false, true, false};
int iNoOfKeys1 = fields1.length;

boolean isExist = helper.isExist(table, iNoOfKeys1, fields1, values1, isNumber1);

//Import Each Structure
if (isExist) {
//If a structure exists already, obtain the serial number
String idx = helper.getString(table, field, fields1, values1, isNumber1);
sindex = Integer.parseInt(idx);
} else {
String [] fields2 = {"nlp100_date", "nlp100_seq"};
String [] values2 = {this.date, String.valueOf(this.serial)};
boolean [] isNumber2 = {false, true};
int iNoOfKeys2 = fields2.length;

//Import each structure
sindex = helper.getMaxSerial(field, table, iNoOfKeys2, fields2, values2, isNumber2);

this.sqlU = "UPDATE catpd.tpd200 ";
this.sqlS = "SET tpd200_structure = '" + structure + "' ";
this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND tpd200_seq = " + String.valueOf(sindex) + " ";

this.ssql = this.sqlU + this.sqlS + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();
}

//Record Each Structure Transformation History
table = "catpd.tpd20t";
field = "tpd20t_seq";

String [] fields2 = {"nlp100_date", "nlp100_seq", "tpd200_seq"};
String [] values2 = {this.date, String.valueOf(this.serial), String.valueOf(sindex)};
boolean [] isNumber2 = {false, true, true};
int iNoOfKeys2 = fields2.length;

//Create each structure transformation history
slog_index = helper.getMaxSerial(field, table, iNoOfKeys2, fields2, values2, isNumber2);

this.sqlU = "UPDATE catpd.tpd20t ";
this.sqlS = "SET nlp10t_date = '" + date + "', ";
this.sqlS += "nlp10t_seq = " + String.valueOf(serial) + " ";
this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND tpd200_seq = " + String.valueOf(sindex) + " ";
this.sqlW += "AND tpd20t_seq = " + String.valueOf(slog_index) + " ";

this.ssql = this.sqlU + this.sqlS + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();
}
} catch (SQLException e) {
System.out.println("Transformer.setStructures() SQL Error..." + e.getMessage());
e.printStackTrace();
} catch (Exception e) {
```

```
System.out.println("Transformer.setStructures() Error..." + e.getMessage());
e.printStackTrace();
} finally {
handler.destroy();
}
}

/**
* clearStructures()
* @param date: Functional Concept Primary Key of Document Date
* @param serial: Functional Concept Primary Key of Document Serial
* @param cindex: Functional Concept Primary Key of Concept Serial
*/
private void clearStructures(String date, int serial) {
//Delete Structure Transformation History Records
this.sqlD = "DELETE FROM catpd.tpd20t ";
this.sqlW = "WHERE nlp100_date = '" + this.date + "' ";
this.sqlW += "AND nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND nlp10t_date = '" + date + "' ";
this.sqlW += "AND nlp10t_seq = " + String.valueOf(serial) + " ";

this.ssql = this.sqlD + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();

//Delete Structures
this.sqlD = "DELETE s.* ";
this.sqlF = "FROM catpd.tpd200 s LEFT JOIN catpd.tpd20t st ";
this.sqlF += "ON s.nlp100_date = st.nlp100_date ";
this.sqlF += "AND s.nlp100_seq = st.nlp100_seq ";
this.sqlF += "AND s.tpd200_seq = st.tpd200_seq ";
this.sqlW = "WHERE s.nlp100_date = '" + this.date + "' ";
this.sqlW += "AND s.nlp100_seq = " + String.valueOf(this.serial) + " ";
this.sqlW += "AND st.tpd200_seq IS NULL ";

this.ssql = this.sqlD + this.sqlF + this.sqlW;
this.sqlh = new SQLHandler(this.ssql);
this.sqlh.destroy();
}

}
```

Figure 7.10 Listing of the class Transformer being used for Transformation Algorithm

7.3. Conclusion

7.3.1. Summary

This chapter presents the prototype system of the proposed methods in this research with the system architecture, a class diagram, database design, employed technologies, and representative source codes. To enable collaborative environment for the transformative product design, the system is designed to develop with the web technologies such as Java servlet, HTML, cascading style sheets, Apache Tomcat Server. The system is running on a machine in the Computational Intelligence and Design Informatics laboratory at Wayne State University.

To support the natural language processing technologies, the system adopts two Java interfaces for WordNet, which is available to public for free: Stanford Natural Language Parser and MIT Java WordNet Interface (JWI 2.2.3). These advanced NLP tools allowed the system to analyze an informal free text based functional requirement document effectively and provide the feasibility to implement the proposed similarity score and other algorithms. This chapter also describes the system flows of the proposed methods for better understanding. Three representative source codes of major classes associated with the proposed methods are presented in this chapter to provide a brief insight of how the system works.

7.3.2. Contribution

This research aims to develop a test-bed to give the transformative insights to designers, even general users who may not familiar with design. Proposed research methods to facilitate the design transformation process from the analysis of functional requirement written in a natural language are fully incorporated in the prototype system. Consequently, anyone who can obtain functional requirement descriptions across domains is able to explore a transformative design environment.

CHAPTER 8

DISCUSSION AND CONCLUSION

8.1. Discussion

8.1.1. Summary

The purpose of the transformative product design (TPD) is to provide a new way of designing, where design starts from the identification of technology advances for the discovery of a new direction of the existing product across domains. Unlike the traditional product design, the transformative product design does not focus on the identification of new design challenges. Instead, it is imperative to recognize technology advances of existing products, which eventually can be transformed into a new product with cross-disciplinary technology advances. Many conceptual tools have been reported to support the traditional innovation approach to solve the classified design challenges for either improvement or radical innovation. These tools are not appropriate to support the transformative product design because the initiatives of the development of the tools have been driven from the relationship between the problem identification and the solution optimization. Hence, there is a need to fulfill the requirement of the transformative product design in a different manner.

This research has focused on how to achieve the purpose of the proposed, new design paradigm with the following questions: (1) how a system systematically understands design knowledge in consideration of teleological knowledge and physical artifacts by reducing the human intervention to a minimum in an efficient manner, and (2) how a system systematically transforms a product from existing design knowledge.

The first question has to be preceded before the second question in order to obtain design knowledge for the input resource of the transformation process. Additionally, current conceptual tools have few reports to embrace the design modeling philosophy of the two different perspectives in an early stage of product design. To respond the necessity of the design knowledge representation and management, this research proposes a new network-based model, called *Interaction Network*. In other words, this research aims to support the design knowledge discovery from functional requirement documents, which are existing design knowledge and easily available in an early design stage, where physical components or drawings are not available specifically.

To implement this process systematically and efficiently, the natural language processing (NLP) technologies and object identification methods in software design have been applied. According to the advances in sematic dictionaries and the NLP tools, such methods have been getting more attention from researchers for the last few years. In Chapter 3, this research proposes an algorithm to determine the Interaction Network from a functional requirement document in an informal free text format in a natural language, i.e., English. The proposed algorithm shows an efficient way to extract design knowledge into an interaction network by identifying both teleological knowledge and physical artifacts. Using the proposed algorithm can practically take a load off from designers to understand a long, complicated design knowledge written in a natural language. Moreover, the machine-readable design knowledge is much beneficial in various ways in design engineering, especially for the development of a computer-aided design tool.

The second research question is the major part of this research to provide conceptual design alternatives for the transformative product design. To enrich the transformation process, a computer aided design tool has to support the semantic approach. The Interaction Network still remains in a natural language to describe any concept. Consequently, if someone tries to find any concept from the Interaction Network, it is necessary for a system to understand the conceptual search keyword (expressed in either a phrase or a sentence) in a natural language and to be intelligent to investigate which concept is appropriate the keyword. To resolve this challenge, this research proposes a similarity score on the basis of stem words (i.e. the origin of words) and its *part-of-speech* in a semantic approach. By doing this, the system is able to find a concept related to the search keyword. This process is named Primitive Concept Detection with the semantic approach.

The semantic search for the concept detection, however, is not enough to fulfill the transformation process. Such a direct search may eliminate the associated function of the product. The Interaction Network can assist this problem because it analytically classifies the relationship between functions and between structures with respect to behaviors. Another proposed method for the concept detection depicts this process to investigate the interactions between functions and the associated structures and behaviors. Finally, the system can acquire additional concept functions connected to the primitive concept functions.

Conceptual functions detected by the proposed methods in Chapter 4 become the starting point of the transformation process in TPD. Chapter 5 presents another algorithm to transform conceptual functions to a new product with cross-disciplinary technology advances (i.e. different conceptual functions in different domains) in a systematic manner. The key characteristic of the transformation algorithm is the *Degree of Transformability*. The *degree of transformability* illustrates how much one conceptual function is associated with and compatible to another concept function(s) for the transformation process. The degree also lies in a semantic approach. Multiple conceptual functions from the design knowledge repository across domains will be investigated according to the degree of transformability. Finally all conceptual functions that satisfy the predetermined threshold for the transformability check will be accepted for the transformation process.

Chapter 7 implements the proposed algorithms into a web-based computer aided design system as a test-bed for the transformation process. The prototype system for the transformative product design is significantly different from traditional conceptual design tools in a sense that the system seeks to find new directions from existing technology advances, even across domains. Additionally, most of the processes implemented in the system have been developed to support a systematic and efficient way for the transformation process by employing natural language processing technologies, semantic approach, and object identification techniques in software engineering.

8.1.2. Contribution

Overall contribution of this research is to present a new product design paradigm to pursue current product design trend in industries by pursuing a way of identifying applicable domains from existing design solutions. The ultimate purpose of this research is to realize a transformative product design environment. To fulfill this purpose, a major contribution of this research is to provide a computer-aided transformative design tool that takes into account cross-disciplinary technology advances and compatibility. It increases psychological inertia across domains beyond products. The inertia has often focused on a single product within its domain. Consequently, the prototype system of this research is able to explore new possible applications of existing design solutions.

To support the goal of a transformative design tool, this research takes a semantic strategy to accelerate the transformation process by developing a similarity score based on word stems and part-of-speech. A semantic strategy of the score takes into account two perspectives of the subject words to compare: word match and structure match. This strategy enables the semantic comparison, even if words are different with respect to their word classes. All the similarity score is dependent on the cardinality of possible matched semantic words such as synonyms, hypernyms, hyponyms, holonyms, and meronyms. The semantic strategy is also stretched to a complementary detection technique in consideration of the structure of interaction network. Even if the semantic strategy to retrieve design knowledge allows more freedom to designers, it could drop an important concept function that has to be associated with. The extended concept function detection algorithm prevents this challenge.

A contribution of such a transformative design tool is also to support designers who may have limited expertise on their own design knowledge, by systematically and efficiently guiding a new product design direction. To guide new design directions from existing design solutions, the proposed similarity score is adapted to check the degree of transformability. According to the degree of transformability, designers could have insights of which products may benefit from the determined functional concept of the existing design solution. This is valuable to those who are lack of professional knowledge of different domains. In other words, the system may support such beginners of conceptual design. A minor contribution is the finding that the interaction network can be applied for technologies or software product if they have appropriate functional requirement descriptions with respect to teleological knowledge and physical knowledge.

Another major contribution of this research is to increase information availability in an upfront design stage by utilizing existing design knowledge. Such a transformative design tool as a conceptual design tool has to be an efficient design tool to capture design knowledge. It is much beneficial for designers when such a design tool takes less time and efforts to translate informal free text based design knowledge. To fulfill this purpose, the de-documentation process with natural language processing techniques and the interaction network generation process allow designers to systematically analyze functional requirement descriptions written in a natural language. The proposed interaction network model is the knowledge container to follow the representative functional modeling in the design research. The design knowledge represented in the proposed network model becomes the source of the later transformation process.

This research also aims to develop a test-bed to give the transformative insights to designers, even general users who may not familiar with design. Proposed research methods to facilitate the design transformation process from the analysis of functional requirement written in a natural language are fully incorporated in the prototype system. Consequently, anyone who can obtain functional requirement descriptions across domains is able to explore a transformative design environment. This is again, a significant contribution of this research, which would result in great assistance for designers, who might be lack of knowledge beyond his/her domain expertise.

8.2. Future Research Direction

This research presents a new product design paradigm to meet the current demands in industries in the theoretical perspective in product design. Hence, this research lays a foundation stone of the new design paradigm conceptually. However, to realize the prototype system in practice, this research has to be extended to additional research works such as form-based design transformation, scalable design knowledge management, efficient network analysis, and empirical design study.

First of all, this research takes an approach to support conceptual design for transformative products on the basis of functions, which can be defined as design intentinos and/or rationales. This research focuses on the transformation process from the perspective of why customers need such transformed products. There has been another design approach and research works to understand products, which is the form-based design. Forms (or structures in the FBS model) are one of significant considerations in product design and they have also faced very challengeing problems to resolve conflicts between them. To explore further design engineering analytics, such conflict resolution process can be a good opportunity for the transformation process. Therefore, to completely support the design transformation process, this research has to take the form-based design approach.

Secondly, the research has to manage scalable design knowledge for the new design paradigm. This research intends to provide a collaborative environment that supports the transformation process across domains. In other words, to enable the transformation process across domains, by nature, it is necessary to gather huge design knowledge from multiple domains. Consequently, it results in big data issues associated with such a great deal of design knowledge. Furthermore, such design knowledge can be represented in heterogeneous formats like mathematic equations and tables in physics. This research limits the input source to an informal free text document. In addition to that, as the design knowledge repository accumulates good candidates for new directions of existing advanced technologies, the entropy of the semantic approach to the comparison of possible options for conceptual functions to continue the transformation process would become exponentially increasing. Therefore, this research should also be able to handle scalable design knowledge management issues.

This research proposes the *interaction networks* to represent the extracted design knowledge from functional requirements. This network has been utilized to support the detection of functional concepts and the transformation process semantically and structurally. This research has faced challenging issues when supporting a huge number of networks for multiple products in the design repository. As aforementioned in the scalability issue, to efficiently support the design transformation process, this research has to enhance the computational performance of the proposed methods with the network in consideration of exponential network growth, especially for network retrieval. This research pays attention to the feasibility test of the proposed design paradigm.

To provide practical feasibility of the concept to designers, it is also necessary for a conceptual design tool to be interoperable with a graphical design tool without losing any design rationale each other, even enhancing design knowledge each other. For the visualization of the generated conceptual design from the proposed system, Google SketchUp would be a good option for a graphic tool to realize the design concept since this design tool takes an intuitive approach to draw something and does not depend on parametric values, which can be easily limitation/barriers to the communication between conceptual design tools and graphic design tools. An empirical design study with the prototype system will give more meaningful insights in research and the system development. By doing this, this conceptual design tool may be able to communicate with other commercial design tools, which is another future research direction.

APPENDIX A. PARSED FUNCTIONAL REQUIREMENTS OF A GRID CONNECTED SOLAR PV SYSTEM

The following example of grid-connected solar PV systems comes from the web site of Sun Pirate Solar Training at http://www.sunpirate.com/. The document of the specifications can be retrieved at http://www.sunpirate.com/photovoltaic-system-types.html, which is available as of Jul 23, 2012.

---

1 1 [A grid-connected solar photovoltaic system goes by several other names , including utility interactive , grid inter-tied , and grid-tie systems -LRB- GTS -RRB-.]

[A/DT, grid-connected/JJ, solar/JJ, photovoltaic/JJ, system/NN, goes/VBZ, by/IN, several/JJ, other/JJ, names/NNS, ,/,, including/VBG, utility/NN, interactive/NNS, ,/,, grid/NN, inter-tied/NN, ,/,, and/CC, grid-tie/JJ, systems/NNS, -LRB-/-LRB-, GTS/NNP, -RRB-/-RRB-, ./.]

2 [The basic set-up of a grid-connected solar PV system involves solar panels that are linked to the local electric utility grid.]

[The/DT, basic/JJ, set-up/NN, of/IN, a/DT, grid-connected/JJ, solar/JJ, PV/NNP, system/NN, involves/VBZ, solar/JJ, panels/NNS, that/WDT, are/VBP, linked/VBN, to/TO, the/DT, local/JJ, electric/JJ, utility/NN, grid/NN, ./.]

3 [But of course , the connection between the solar array and the utility grid is a bit more complex than that.]

[But/CC, of/IN, course/NN, ,/,, the/DT, connection/NN, between/IN, the/DT, solar/JJ, array/NN, and/CC, the/DT, utility/NN, grid/NN, is/VBZ, a/DT, bit/RB, more/RBR, complex/JJ, than/IN, that/DT, ./.]

---

4 [The following are the required components for a grid-connected solar PV system : Solar panels , Inverter , DC disconnect , AC breaker panel , Kilowatt-hour meter , Utility disconnect , and Electrical wiring in your home]

[The/DT, following/NN, are/VBP, the/DT, required/VBN, components/NNS, for/IN, a/DT, grid-connected/JJ, solar/JJ, PV/NNP, system/NN, :/:, Solar/NNP, panels/NNS, ,/,, Inverter/NNP, ,/,, DC/NNP, disconnect/NNP, ,/,, AC/NNP, breaker/NN, panel/NN, ,/,, Kilowatt-hour/JJ, meter/NN, ,/,, Utility/NNP, disconnect/NNP, ,/,, and/CC, Electrical/JJ, wiring/NN, in/IN, your/PRP$, home/NN]

2

3 1 [All of these components function together in an intricate design to collect and distribute clean , renewable energy.]

[All/DT, of/IN, these/DT, components/NNS, function/VBP, together/RB, in/IN, an/DT, intricate/JJ, design/NN, to/TO, collect/VB, and/CC, distribute/VB, clean/JJ, ,/,, renewable/JJ, energy/NN, ./.]

2 [It starts with the solar panels that collect sunlight and convert it into an electrical current.]

[It/PRP, starts/VBZ, with/IN, the/DT, solar/JJ, panels/NNS, that/WDT, collect/VB, sunlight/NN, and/CC, convert/VB, it/PRP, into/IN, an/DT, electrical/JJ, current/NN, ./.]

3 [The electricity that these panels produce is direct current -LRB- DC -RRB- energy , but since your home and the grid function on alternating current -LRB- AC -RRB- energy , the raw solar energy needs to be converted]

[The/DT, electricity/NN, that/IN, these/DT, panels/NNS, produce/VBP, is/VBZ, direct/JJ, current/JJ, -LRB-/-LRB-, DC/VBN, -RRB-/-RRB-, energy/NN, ,/,, but/CC, since/IN, your/PRP$, home/NN, and/CC, the/DT, grid/NN, function/VBP, on/IN, alternating/VBG, current/JJ, -LRB-/-LRB-, AC/NNP, -RRB-/-RRB-, energy/NN, ,/,, the/DT, raw/JJ, solar/JJ, energy/NN, needs/VBZ, to/TO, be/VB, converted/VBN]

4

5 1 [For that , the PV system relies on an inverter -LRB- sometimes called a power conditioning unit , or PCU -RRB- , which is one of the most important components in your PV system.]

[For/IN, that/DT, ,/,, the/DT, PV/NNP, system/NN, relies/VBZ, on/IN, an/DT, inverter/NN, -LRB-/-LRB-, sometimes/RB, called/VBN, a/DT, power/NN, conditioning/NN, unit/NN, ,/,, or/CC, PCU/NNP, -RRB-/-RRB-, ,/,, which/WDT, is/VBZ, one/CD, of/IN, the/DT, most/RBS, important/JJ, components/NNS, in/IN, your/PRP$, PV/NNP, system/NN, ./.]

2 [The inverter will convert the DC energy to AC energy which can then be used inside your home and or be safely sent to the local utility grid.]

[The/DT, inverter/NN, will/MD, convert/VB, the/DT, DC/NNP, energy/NN, to/TO, AC/VB, energy/NN, which/WDT, can/MD, then/RB, be/VB, used/VBN, inside/IN, your/PRP$, home/NN, and/CC, or/CC, be/VB, safely/RB, sent/VBN, to/TO, the/DT, local/JJ, utility/NN, grid/NN, ./.]

3 [From there , the electricity travels via your home 's normal wiring through the AC breaker panel]

[From/IN, there/RB, ,/,, the/DT, electricity/NN, travels/VBZ, via/IN, your/PRP$, home/NN, 's/POS, normal/JJ, wiring/NN, through/IN, the/DT, AC/NNP, breaker/NN, panel/NN]

6

7 1 [A grid-tied solar PV system also requires an array DC disconnect , which is essentially a

switch that allows you to stop the flow of electricity from your solar panels.]

[A/DT, grid-tied/JJ, solar/JJ, PV/NN, system/NN, also/RB, requires/VBZ, an/DT, array/NN, DC/NNP, disconnect/NNP, ,/,, which/WDT, is/VBZ, essentially/RB, a/DT, switch/NN, that/WDT, allows/VBZ, you/PRP, to/TO, stop/VB, the/DT, flow/NN, of/IN, electricity/NN, from/IN, your/PRP$, solar/JJ, panels/NNS, ./.]

2 [This is used to shut the system down in emergencies or when maintenance needs to be performed.]

[This/DT, is/VBZ, used/VBN, to/TO, shut/VB, the/DT, system/NN, down/RB, in/IN, emergencies/NNS, or/CC, when/WRB, maintenance/NN, needs/VBZ, to/TO, be/VB, performed/VBN, ./.]

3 [You will also have a utility disconnect which is used by the local utility to stop the flow of energy when they need to perform maintenance on the utility grid.]

[You/PRP, will/MD, also/RB, have/VB, a/DT, utility/NN, disconnect/VB, which/WDT, is/VBZ, used/VBN, by/IN, the/DT, local/JJ, utility/NN, to/TO, stop/VB, the/DT, flow/NN, of/IN, energy/NN, when/WRB, they/PRP, need/VBP, to/TO, perform/VB, maintenance/NN, on/IN, the/DT, utility/NN, grid/NN, ./.]

4 [At this point , a kilowatt-hour meter provides a read-out of how much power your solar array has produced in order to calculate your monthly utility bill.]

[At/IN, this/DT, point/NN, ,/,, a/DT, kilowatt-hour/JJ, meter/NN, provides/VBZ, a/DT, read-out/NN, of/IN, how/WRB, much/JJ, power/NN, your/PRP$, solar/JJ, array/NN, has/VBZ, produced/VBN, in/IN, order/NN, to/TO, calculate/VB, your/PRP$, monthly/JJ, utility/NN, bill/NN, ./.]

5 [This leads us to one of the biggest benefits of a grid-tied solar panel system.]

[This/DT, leads/VBZ, us/PRP, to/TO, one/CD, of/IN, the/DT, biggest/JJS, benefits/NNS, of/IN, a/DT, grid-tied/JJ, solar/JJ, panel/NN, system/NN, ./.]

6 [When connected to the existing power grid and producing energy , your solar panel will pump all excess clean energy you produce into the grid.]

[When/WRB, connected/VBN, to/TO, the/DT, existing/VBG, power/NN, grid/NN, and/CC, producing/VBG, energy/NN, ,/,, your/PRP$, solar/JJ, panel/NN, will/MD, pump/VB, all/DT, excess/JJ, clean/JJ, energy/NN, you/PRP, produce/VBP, into/IN, the/DT, grid/NN, ./.]

7 [Through a program called net metering , which many -LRB- but not all -RRB- communities now have , you will get credit from your local utility for all of the power fed into the grid.]

[Through/IN, a/DT, program/NN, called/VBN, net/JJ, metering/NN, ,/,, which/WDT, many/JJ, -LRB-/-LRB-, but/CC, not/RB, all/DT, -RRB-/-RRB-, communities/NNS, now/RB, have/VBP, ,/,, you/PRP, will/MD, get/VB, credit/NN, from/IN, your/PRP$, local/JJ, utility/NN, for/IN, all/DT, of/IN, the/DT, power/NN, fed/VBN, into/IN, the/DT, grid/NN, ./.]

8 [In essence , the meter will spin backwards during these times]

[In/IN, essence/NN, ,/,, the/DT, meter/NN, will/MD, spin/VB, backwards/RB, during/IN, these/DT, times/NNS]

8

9 1 [Conversely , at night and during times when your solar system does n't make enough

energy for your home , you can draw power from the local utility grid.]

[Conversely/RB, ,/,, at/IN, night/NN, and/CC, during/IN, times/NNS, when/WRB, your/PRP$, solar/JJ, system/NN, does/VBZ, n't/RB, make/VB, enough/JJ, energy/NN, for/IN, your/PRP$, home/NN, ,/,, you/PRP, can/MD, draw/VB, power/NN, from/IN, the/DT, local/JJ, utility/NN, grid/NN, ./.]

2 [As a result , most grid-connected solar system do not include a battery since all of the energy is consumed as it is produced]

[As/IN, a/DT, result/NN, ,/,, most/RBS, grid-connected/JJ, solar/JJ, system/NN, do/VBP, not/RB, include/VB, a/DT, battery/NN, since/IN, all/DT, of/IN, the/DT, energy/NN, is/VBZ, consumed/VBN, as/IN, it/PRP, is/VBZ, produced/VBN]

APPENDIX B. PARSED FUNCTIONAL REQUIREMENTS OF A STREETLIGHT EXAMPLE

The following example of streetlight specifications comes from the city of Arlington, TX. The document of the specifications can be retrieved at http://www.arlingtontx.gov/publicworks/pdf/specialprovisions/SECT16%20-%20Streetlights.pdf, which is available as of Jul 23, 2012.

1

1 [Each steel streetlight pole shall include either single or twin davit type arms to support the luminaires.]

[Each/DT, steel/NN, streetlight/NN, pole/NN, shall/MD, include/VB, either/DT, single/JJ, or/CC, twin/JJ, davit/JJ, type/NN, arms/NNS, to/TO, support/VB, the/DT, luminaires/NNS, ./.]

2 [Each pole shall also include anchor bolts and , if required , transformer bases.]

[Each/DT, pole/NN, shall/MD, also/RB, include/VB, anchor/JJ, bolts/NNS, and/CC, ,/,, if/IN, required/JJ, ,/,, transformer/NN, bases/NNS, ./.]

3 [The Specification covers 2-section steel davit type luminaire poles.]

[The/DT, Specification/NNP, covers/VBZ, 2-section/JJ, steel/NN, davit/NN, type/NN, luminaire/NN, poles/NNS, ./.]

4 [The general design of the poles conform to the requirements of the plans and the typical d

awings with no guys , struts , rods , stay braces , or clamps of U-

bolts , except where notedotherwise.]

[The/DT, general/JJ, design/NN, of/IN, the/DT, poles/NNS, conform/VBP, to/TO, the/DT, requirements/NNS, of/IN, the/DT, plans/NNS, and/CC, the/DT, typical/JJ, drawings/NNS, with/IN, no/DT, guys/NNS, ,/,, struts/NNS, ,/,, rods/NNS, ,/,, stay/NN, braces/NNS, ,/,, or/CC, clamps/NNS, of/IN, U-bolts/NNP, ,/,, except/IN, where/WRB, noted/VBD, otherwise/RB, ./.]

5 [Each complete pole assembly shall be within the dimension limits shown on the drawings]

[Each/DT, complete/JJ, pole/NN, assembly/NN, shall/MD, be/VB, within/IN, the/DT, dimension/NN, limits/NNS, shown/VBN, on/IN, the/DT, drawings/NNS, ./.]

6 [All poles shall have a round cross-sectional design.]

[All/DT, poles/NNS, shall/MD, have/VB, a/DT, round/JJ, cross-sectional/JJ, design/NN, ./.]

7 [In the design of the pole structures , each luminaire shall follow the limits of 80 pounds in weight and 1.0 square foot for projected area.]

[In/IN, the/DT, design/NN, of/IN, the/DT, pole/NN, structures/NNS, ,/,, each/DT, luminaire/NN, shall/MD, follow/VB, the/DT, limits/NNS, of/IN, 80/CD, pounds/NNS, in/IN, weight/NN, and/CC, 1.0/CD, square/JJ, foot/NN, for/IN, projected/VBN, area/NN, ./.]

8 [Pole and arms shall be fabricated from United States standard 11 gauge -LRB- minimum thickness -RRB- coil stock ; weldable grade , hot-rolled commercial quality carbon steel with aguaranteed minimum yield strength of 55,000 P.S.I. after fabrication.]

[Pole/NNP, and/CC, arms/NNS, shall/MD, be/VB, fabricated/VBN, from/IN, United/NNP, States/NNPS, standard/NN, 11/CD, gauge/NN, -LRB-/-LRB-, minimum/JJ, thickness/NN, -RRB-/-RRB-, coil/JJ, stock/NN, ;/:, weldable/JJ, grade/NN, ,/,, hot-rolled/JJ, commercial/JJ, quality/NN, carbon/NN, steel/NN, with/IN, a/DT, guaranteed/VBN, minimum/JJ, yield/NN, strength/NN, of/IN, 55,000/NNP, P.S.I./NNP, after/IN, fabrication/NN, ./.]

9 [It shall be one piece construction with a full length longitudinal high frequency resistance weld.]

[It/PRP, shall/MD, be/VB, one/CD, piece/NN, construction/NN, with/IN, a/DT, full/JJ, length/NN, longitudinal/NNP, high/JJ, frequency/NN, resistance/NN, weld/VBN, ./.]

10 [Both the pole and the davit arms shall have a round cross-section having a uniform taper of approximately 0.14 inch of diameter change per foot of le ngth.]

[Both/DT, the/DT, pole/NN, and/CC, the/DT, davit/JJ, arms/NNS, shall/MD, have/VB, a/DT, round/NN, cross-section/NN, having/VBG, a/DT, uniform/JJ, taper/NN, of/IN, approximately/RB, 0.14/CD, inch/NN, of/IN, diameter/NN, change/NN, per/IN, foot/NN, of/IN, length/NN, ./.]

11 [The davit arm shall be provided with a " O.D. steel tenon for luminaire accommodation.]

[The/DT, davit/NN, arm/NN, shall/MD, be/VB, provided/VBN, with/IN, a/DT, "/'', O.D./NN, steel/NN, tenon/VBN, for/IN, luminaire/JJ, accommodation/NN, ./.]

12 [The angle above horizontal of the arm shaft at the point of luminaire attachment shall be 5 degrees nominal.]

[The/DT, angle/NN, above/IN, horizontal/NN, of/IN, the/DT, arm/NN, shaft/NN, at/IN, the/DT, point/NN, of/IN, luminaire/JJ, attachment/NN, shall/MD, be/VB, 5/CD, degrees/NNS, nominal/JJ, ./.]

13 [A pole assembly shall include the pole , the davit arm -LRB- s -RRB-

, the transformer base -LRB- if required -RRB- , the anchor bolts and the bolt covers.]

[A/DT, pole/NNP, assembly/NN, shall/MD, include/VB, the/DT, pole/NN, ,/,, the/DT, davit/JJ, arm/NN, -LRB-/-LRB-, s/PRP, -RRB-/-RRB-, ,/,, the/DT, transformer/NN, base/NN, -LRB-/-LRB-, if/IN, required/VBN, -RRB-/-RRB-, ,/,, the/DT, anchor/JJ, bolts/NNS, and/CC, the/DT, bolt/NN, covers/VBZ, ./.]

14 [The pole assembly shall provide a luminaire mounting height of 40 feet.]

[The/DT, pole/NNP, assembly/NN, shall/MD, provide/VB, a/DT, luminaire/JJ, mounting/VBG, height/NN, of/IN, 40/CD, feet/NNS, ./.]

15 [The allowance pole deflection , when fully loaded , shall not exceed an angular rotation o

one degree forty minutes , as measured between the intersections of the vertical centerline

throughthe top.]

[The/DT, allowance/NN, pole/NN, deflection/NN, ,/,, when/WRB, fully/RB, loaded/VBN, ,/,, shall/MD, not/RB, exceed/VB, an/DT, angular/NN, rotation/NN, of/IN, one/CD, degree/NN, forty/NN, minutes/NNS, ,/,, as/IN, measured/VBN, between/IN, the/DT, intersections/NNS, of/IN, the/DT, vertical/JJ, centerline/NN, through/IN, the/DT, top/NN, ./.]

16 [The deflection of the pole when loaded in this manner shall not exceed 0.35.]

[The/DT, deflection/NN, of/IN, the/DT, pole/NN, when/WRB, loaded/VBN, in/IN, this/DT, manner/NN, shall/MD, not/RB, exceed/VB, 0.35/CD, ./.]

17 [Adjustments to vertical by leveling nuts will be permitted.]

[Adjustments/NNS, to/TO, vertical/VB, by/IN, leveling/VBG, nuts/NNS, will/MD, be/VB, permitted/VBN, ./.]

18 [Each pole shall have a base plate -LRB- anchor base -RRB-.]

[Each/DT, pole/NN, shall/MD, have/VB, a/DT, base/NN, plate/NN, -LRB-/-LRB-, anchor/VB, base/NN, -RRB-/-RRB-, ./.]

19 [The anchor bolt holes in the pole base plate shall be -
inch larger than the anchor bolt diameters.]

[The/DT, anchor/JJ, bolt/NN, holes/NNS, in/IN, the/DT, pole/NNP, base/NN, plate/NN, shall/MD, be/VB, -/CD, inch/NN, larger/JJR, than/IN, the/DT, anchor/JJ, bolt/NN, diameters/NNS, ./.]

20 [Anchor bolt holes in the base plates shall be elongated.]

[Anchor/NNP, bolt/NN, holes/NNS, in/IN, the/DT, base/NN, plates/NNS, shall/MD, be/VB, elongated/VBN, ./.]

21 [Each non-
transformer base pole shall have an approximate 4 " x 6.5 " handhole with a reinforced fra
me located]

[Each/DT, non-transformer/JJ, base/NN, pole/NN, shall/MD, have/VB, an/DT, approximate/JJ, 4/CD, "/', x/NNS, 6.5/VBG, "/', handhole/NN, with/IN, a/DT, reinforced/JJ, frame/NN, located/VBN]

2

3 1 [The handhole shall be furnished with a cover and screws.]

[The/DT, handhole/NN, shall/MD, be/VB, furnished/VBN, with/IN, a/DT, cover/NN, and/CC, screws/NNS, ./.]

2 [20 percent spare handhole covers shall be furnished with each order.]

[20/CD, percent/NN, spare/VBP, handhole/JJ, covers/NNS, shall/MD, be/VB, furnished/VBN, with/IN, each/DT, order/NN, ./.]

3 [For small -LRB- less than 10 poles -RRB-

orders , at least one spare cover shall be furnished with each order.]

[For/IN, small/JJ, -LRB-/-LRB-, less/JJR, than/IN, 10/CD, poles/NNS, -RRB-/-RRB-, orders/NNS, ,/,, at/IN, least/JJS, one/CD, spare/JJ, cover/NN, shall/MD, be/VB, furnished/VBN, with/IN, each/DT, order/NN, ./.]

4 [The handhole opening shall be circumferentially welded in the pole shaft.]

[The/DT, handhole/JJ, opening/NN, shall/MD, be/VB, circumferentially/RB, welded/VBN in/IN, the/DT, pole/NN, shaft/NN, ./.]

5 [The handhole opening shall include two tabs for mounting a steel cover with hex head atta

chment screws.]

[The/DT, handhole/JJ, opening/NN, shall/MD, include/VB, two/CD, tabs/NNS, for/IN, mounting/VBG, a/DT, steel/NN, cover/NN, with/IN, hex/NN, head/NN, attachment/NN, screws/NNS, ./.]

6 [A nut holder shall be welded to the vertical side of the handhole.]

[A/DT, nut/NN, holder/NN, shall/MD, be/VB, welded/VBN, to/TO, the/DT, vertical/JJ, side/NN, of/IN, the/DT, handhole/NN, ./.]

7 [The handhole shall be at 90 degrees clockwise with respect to the luminaire are when view

ed from the top of the pole]

[The/DT, handhole/NN, shall/MD, be/VB, at/IN, 90/CD, degrees/NNS, clockwise/VBN, with/IN, respect/NN, to/TO, the/DT, luminaire/NN, are/VBP, when/WRB, viewed/VBN, from/IN, the/DT, top/NN, of/IN, the/DT, pole/NN]

4

5 1 [Each pole shall be furnished with four bolt covers.]

[Each/DT, pole/NN, shall/MD, be/VB, furnished/VBN, with/IN, four/CD, bolt/NN, covers/NNS, ./.]

2 [Bolt covers shall be zinc die cast.]

[Bolt/JJ, covers/NNS, shall/MD, be/VB, zinc/VBN, die/VB, cast/NN, ./.]

3 [Each bolt cover shall be fastened to the shaft by a " stainless steel , self-tapping , hex head screw.]

[Each/DT, bolt/NN, cover/NN, shall/MD, be/VB, fastened/VBN, to/TO, the/DT, shaft/NN, by/IN, a/SYM, "/', stainless/VBG, steel/NN, ,/,, self-tapping/JJ, ,/,, hex/JJ, head/NN, screw/NN, ./.]

4 [Four anchor bolts shall be furnished with each pole.]

[Four/CD, anchor/JJ, bolts/NNS, shall/MD, be/VB, furnished/VBN, with/IN, each/DT, pole/NN, ./.]

5 [One leveling nut , one anchor nut , two " leveling shims and two washers shall be furnishe d for each anchor bolt.]

[One/CD, leveling/NN, nut/NN, ,/,, one/CD, anchor/JJ, nut/NN, ,/,, two/CD, "/', leveling/NN, shims/NNS, and/CC, two/CD, washers/NNS, shall/MD, be/VB, furnished/VBN, for/IN, each/DT, anchor/JJ, bolt/NN, ./.]

6 [Anchor bolts shall be fabricated from steel with a minimum yield strength of 55,000 P.S.I

and a minimum ultimate tensile strength of 75,000 to 95,000 P.S.I. Anchor bolts shall hav

1 " diametersand 36 " lengths.]

[Anchor/NNP, bolts/NNS, shall/MD, be/VB, fabricated/VBN, from/IN, steel/NN, with/IN, a/DT, minimum/JJ, yield/NN, strength/NN, of/IN, 55,000/NNP, P.S.I./NNP, and/CC, a/DT, minimum/JJ, ultimate/JJ, tensile/NN, strength/NN, of/IN, 75,000/CD, to/TO, 95,000/CD, P.S.I./NNP, Anchor/NNP, bolts/NNS, shall/MD, have/VB, 1/CD, "/', diameters/VBZ, and/CC, 36/VB, "/', lengths/NNS, ./.]

7 [Anchor bolts shall develop strengths comparable to their respective poles.]

[Anchor/NNP, bolts/NNS, shall/MD, develop/VB, strengths/NNS, comparable/JJ, to/TO, their/PRP$, respective/JJ, poles/NNS, ./.]

8 [Each anchor bolt shall have a 4-inch , 90-degree bend at the unthreaded end.]

[Each/DT, anchor/JJ, bolt/NN, shall/MD, have/VB, a/DT, 4-inch/JJ, ,/,, 90-degree/JJ, bend/NN, at/IN, the/DT, unthreaded/JJ, end/NN, ./.]

9 [Anchor bolts shall be hot bent and shall meet the requirements of ASTM A-
36 modified to provide the strength specified above]

[Anchor/NNP, bolts/NNS, shall/MD, be/VB, hot/JJ, bent/NN, and/CC, shall/MD, meet/VB, the/DT, requirements/NNS, of/IN, ASTM/NNP, A-36/NNP, modified/VBN, to/TO, provide/VB, the/DT, strength/NN, specified/VBN, above/IN]

6

7 1 [Nuts shall meet the requirements of ASTM-563 grade A , and ANSI B18 .2.2 hex type.]

[Nuts/NNS, shall/MD, meet/VB, the/DT, requirements/NNS, of/IN, ASTM-563/NN, grade/NN, A/NN, ,/,, and/CC, ANSI/NNP, B18/NNP, .2.2/CD, hex/NN, type/NN, ./.]

2 [Bolts and nuts shall be galvanized in accordance with ASTM A-153]

[Bolts/NNS, and/CC, nuts/NNS, shall/MD, be/VB, galvanized/VBN, in/IN, accordance/NN, with/IN, ASTM/NNP, A-153/NNP]

8

9 1 [Each pole shall be equipped with a grounding lug , which will accommodate a A.W.G. -LRB- American Wire Gauge -RRB- # 6 ground wire.]

[Each/DT, pole/NN, shall/MD, be/VB, equipped/VBN, with/IN, a/DT, grounding/NN, lug/NN, ,/,, which/WDT, will/MD, accommodate/VB, a/DT, A.W.G./NNP, -LRB-/-LRB-, American/NNP, Wire/NNP, Gauge/NNP, -RRB-/-RRB-, #/#, 6/CD, ground/NN, wire/NN, ./.]

2 [The lug shall be electrically bonded to the pole and shall be located inside the pole opposi

e the handhole , if a non-transformer base pole , or within two inches -LRB- 2 " -RRB- of the pole base, if a transformer base pole.]

[The/DT, lug/NN, shall/MD, be/VB, electrically/RB, bonded/VBN, to/TO, the/DT, pole/NN, and/CC, shall/MD, be/VB, located/VBN, inside/IN, the/DT, pole/NN, opposite/IN, the/DT, handhole/NN, ,/,, if/IN, a/DT, non-transformer/JJ, base/NN, pole/NN, ,/,, or/CC, within/IN, two/CD, inches/NNS, -LRB-/-LRB-, 2/CD, "/', -RRB-/-RRB-, of/IN, the/DT, pole/NN, base/NN, ,/,, if/IN, a/DT, transformer/NN, base/VBP, pole/NN, ./.]

3 [The lug shall consist of a 0.5-inch long hex head bolt with nut.]

[The/DT, lug/NN, shall/MD, consist/VB, of/IN, a/DT, 0.5-inch/JJ, long/JJ, hex/NN, head/NN, bolt/NN, with/IN, nut/NN, ./.]

4 [The bolt shall have 13 UNC -LRB- Unified National Coarse -RRB- threads.]

[The/DT, bolt/NN, shall/MD, have/VB, 13/CD, UNC/NNS, -LRB-/-LRB-, Unified/NNP, National/NNP, Coarse/NNP, -RRB-/-RRB-, threads/NNS, ./.]

5 [The pole shall include a handling hook or ring located inside the pole at the top to facilitat e installation of the pole.]

[The/DT, pole/NN, shall/MD, include/VB, a/DT, handling/NN, hook/NN, or/CC, ring/NN, located/VBN, inside/IN, the/DT, pole/NN, at/IN, the/DT, top/NN, to/TO, facilitate/VB, installation/NN, of/IN, the/DT, pole/NN, ./.]

6 [The handling hook or ring shall not interfere with installation of the davit arms and shall h ave sufficient strength to support the weight of the pole.]

[The/DT, handling/NN, hook/NN, or/CC, ring/NN, shall/MD, not/RB, interfere/VB, with/IN, installation/NN, of/IN, the/DT, davit/JJ, arms/NNS, and/CC, shall/MD, have/VB, sufficient/JJ, strength/NN, to/TO, support/VB, the/DT, weight/NN, of/IN, the/DT, pole/NN, ./.]

7 [The pole shall be the davit type with single or twin arms.]

[The/DT, pole/NN, shall/MD, be/VB, the/DT, davit/JJ, type/NN, with/IN, single/JJ, or/CC, twin/JJ, arms/NNS, ./.]

8 [The single davit arm or twin davit arms shall be a separate section that telescopes the pole shaft by one foot.]

[The/DT, single/JJ, davit/NN, arm/NN, or/CC, twin/JJ, davit/NN, arms/NNS, shall/MD, be/VB, a/DT, separate/JJ, section/NN, that/WDT, telescopes/VBZ, the/DT, pole/NN, shaft/NN, by/IN, one/CD, foot/NN, ./.]

9 [Each davit arm shall have a 9-foot span and a nominal radius of 9 feet.]

[Each/DT, davit/NN, arm/NN, shall/MD, have/VB, a/DT, 9-foot/JJ, span/NN, and/CC, a/DT, nominal/JJ, radius/NNS, of/IN, 9/CD, feet/NNS, ./.]

10 [The centerline of the davit arm at the luminaire shall be five degrees above horizontal.]

[The/DT, centerline/NN, of/IN, the/DT, davit/JJ, arm/NN, at/IN, the/DT, luminaire/NN, shall/MD, be/VB, five/CD, degrees/NNS, above/IN, horizontal/NN, ./.]

11 [The ends of the circular sweeps of davit arms shall have inch outside diameter tenons to ac
commodate standard luminaires.]

[The/DT, ends/NNS, of/IN, the/DT, circular/JJ, sweeps/NNS, of/IN, davit/JJ, arms/NNS, shall/MD, have/VB, inch/NN, outside/IN, diameter/NN, tenons/NNS, to/TO, accommodate/VB, standard/JJ, luminaires/NNS, ./.]

12 [When arm spans of up to fifteen feet are required , the additional length shall be obtained
rom straight section extending out at the five-degree angle.]

[When/WRB, arm/NN, spans/NNS, of/IN, up/RB, to/TO, fifteen/JJ, feet/NNS, are/VBP, required/VBN, ,/,, the/DT, additional/JJ, length/NN, shall/MD, be/VB, obtained/VBN, from/IN, straight/JJ, section/NN, extending/VBG, out/RP, at/IN, the/DT, five-degree/JJ, angle/NN, ./.]

13 [This will result in a mounting height that is slightly higher than 40 feet]

[This/DT, will/MD, result/VB, in/IN, a/DT, mounting/VBG, height/NN, that/WDT, is/VBZ, slightly/RB, higher/JJR, than/IN, 40/CD, feet/NNS]

10

11 1 [The transformer base shall be cast aluminum with all necessary fittings and attachments.]

[The/DT, transformer/NN, base/NN, shall/MD, be/VB, cast/VBN, aluminum/NN, with/IN, all/DT, necessary/JJ, fittings/NNS, and/CC, attachments/NNS, ./.]

2 [The transformer base shall be so designed to afford the lighting standard the quality of `` b reaking away " under vehicular impact.]

[The/DT, transformer/NN, base/NN, shall/MD, be/VB, so/RB, designed/VBN, to/TO, afford/VB, the/DT, lighting/NN, standard/NN, the/DT, quality/NN, of/IN, ``/``, breaking/VBG, away/RP, "/", under/IN, vehicular/NN, impact/NN, ./.]

3 [The transformer base shall have a height of 17 " to provide bumper area contact with a pas senger vehicle.]

[The/DT, transformer/NN, base/NN, shall/MD, have/VB, a/DT, height/NN, of/IN, 17/CD, "/", to/TO, provide/VB, bumper/NN, area/NN, contact/NN, with/IN, a/DT, passenger/NN, vehicle/NN, ./.]

4 [The transformer base shall break upon impact of a vehicle weighing approximately two th ousand pounds and traveling at a speed of twenty miles per hour or faster.]

[The/DT, transformer/NN, base/NN, shall/MD, break/VB, upon/IN, impact/NN, of/IN, a/DT, vehicle/NN, weighing/VBG, approximately/RB, two/CD, thousand/CD, pounds/NNS, and/CC, traveling/VBG, at/IN, a/DT, speed/NN, of/IN, twenty/CD, miles/NNS, per/IN, hour/NN, or/CC, faster/JJR, ./.]

5 [The transformer base shall meet the structural and wind load requirements.]

[The/DT, transformer/NN, base/NN, shall/MD, meet/VB, the/DT, structural/JJ, and/CC, wind/NN, load/NN, requirements/NNS, ./.]

6 [The transformer base shall conform to the requirements of STANDARD SPECIFICATIO

NS FOR STRUCTURAL SUPPORTS FOR HIGHWAY SIGNS , LUMINAIRES AND T

RAFFIC SIGNALS , includingall addenda thereto , published by AASHTO.]

[The/DT, transformer/NN, base/NN, shall/MD, conform/VB, to/TO, the/DT, requirements/NNS, of/IN, STANDARD/NNP, SPECIFICATIONS/NNP, FOR/IN, STRUCTURAL/NNP, SUPPORTS/NNP, FOR/IN, HIGHWAY/NNP, SIGNS/NNP, ,/,, LUMINAIRES/NNP, AND/CC, TRAFFIC/NNP, SIGNALS/NNP, ,/,, including/VBG, all/DT, addenda/NN, thereto/NN, ,/,, published/VBN, by/IN, AASHTO/NNP, ./.]

7 [A door opening with a removable door shall be provided in the side of the base approxima

tely 8.56 " x 8.94 " x 11 " in size.]

[A/DT, door/NN, opening/NN, with/IN, a/DT, removable/JJ, door/NN, shall/MD, be/VB, provided/VBN, in/RP, the/DT, side/NN, of/IN, the/DT, base/NN, approximately/RB, 8.56/CD, "/', x/SYM, 8.94/CD, "/', x/SYM, 11/CD, "/', in/IN, size/NN, ./.]

8 [Aluminum base material for the transformer base shall conform to ASTM B-

108 alloy SC 70A-T6.]

[Aluminum/NNP, base/NN, material/NN, for/IN, the/DT, transformer/NN, base/NN, shall/MD, conform/VB, to/TO, ASTM/NNP, B-108/NNP, alloy/NN, SC/NNP, 70A-T6/NNP, ./.]

9 [Each transformer base shall be furnished with four galvanized , one-

inch diameter bolts , each with a hex nut and two washers suitable for attaching the pole to the transformer base.]

[Each/DT, transformer/NN, base/NN, shall/MD, be/VB, furnished/VBN, with/IN, four/CD, galvanized/JJ, ,/,, one-inch/JJ, diameter/NN, bolts/NNS, ,/,, each/DT, with/IN, a/DT, hex/NN, nut/NN, and/CC, two/CD, washers/NNS, suitable/JJ, for/IN, attaching/VBG, the/DT, pole/NN, to/TO, the/DT, transformer/NN, base/NN, ./.]

10 [The bolts used for the transformer base shall satisfy ASTM A-325]

[The/DT, bolts/NNS, used/VBN, for/IN, the/DT, transformer/NN, base/NN, shall/MD, satisfy/VB, ASTM/NNP, A-325/NNP]

12

13 1 [The optical system of the luminaire shall consist of reflectors , lenses , and lamps.]

[The/DT, optical/JJ, system/NN, of/IN, the/DT, luminaire/NN, shall/MD, consist/VB, of/IN, reflectors/NNS, ,/,, lenses/NNS, ,/,, and/CC, lamps/NNS, ./.]

2 [The reflector shall be aodic surfaced aluminum secured with a spring latch for each positio

ning and positive seal.]

[The/DT, reflector/NN, shall/MD, be/VB, aodic/JJ, surfaced/VBD, aluminum/NN, secured/VBN, with/IN, a/DT, spring/NN, latch/VBP, for/IN, each/DT, positioning/NN, and/CC, positive/JJ, seal/NN, ./.]

3 [The lens shall be a flat glass lens.]

[The/DT, lens/NN, shall/MD, be/VB, a/DT, flat/JJ, glass/NN, lens/NN, ./.]

4 [The luminaire shall be furnished with a lens producing the specified IES light distribution patterns.]

[The/DT, luminaire/NN, shall/MD, be/VB, furnished/VBN, with/IN, a/DT, lens/NN, producing/VBG, the/DT, specified/JJ, IES/NNP, light/NN, distribution/NN, patterns/NNS, ./.]

5 [IES light distribution patterns Type II , Type II-4 way and Type III shall be available.]

[IES/NNP, light/NN, distribution/NN, patterns/NNP, Type/NNP, II/NNP, ,/,, Type/JJ, II-4/NN, way/NN, and/CC, Type/NNP, III/NNP, shall/MD, be/VB, available/JJ, ./.]

6 [The lamp shall be a high pressure sodium lamp with a standard roadway luminaire brass b ase.]

[The/DT, lamp/NN, shall/MD, be/VB, a/DT, high/JJ, pressure/NN, sodium/NN, lamp/NN, with/IN, a/DT, standard/JJ, roadway/NN, luminaire/NN, brass/NN, base/NN, ./.]

7 [The luminaire shall be furnished in 100 watt-9 ,500 lumens , 150 watt-16 ,000 lumens , 200 watt-27 ,000 lumens , 400 watt-50 ,000 lumens , and 1,000 watt-130 ,000 lumens with rated life of24,000 hours as specified in the order or on the plans]

[The/DT, luminaire/NN, shall/MD, be/VB, furnished/VBN, in/IN, 100/CD, watt-9/JJ, ,500/CD, lumens/NNS, ,/,, 150/CD, watt-16/JJ, ,000/CD, lumens/NNS, ,/,, 200/CD, watt-27/JJ, ,000/CD, lumens/NNS, ,/,, 400/CD, watt-50/JJ, ,000/CD, lumens/NNS, ,/,, and/CC, 1,000/CD, watt-130/JJ, ,000/CD, lumens/NNS, with/IN, rated/JJ, life/NN, of/IN, 24,000/CD, hours/NNS, as/IN, specified/JJ, in/IN, the/DT, order/NN, or/CC, on/IN, the/DT, plans/NNS]

14 1 [The electrical system of the luminaire shall consist of lamp socket support , terminal bloc , secondary lightning arrestor , ballast , electronic starter , and photoswitch.]

[The/DT, electrical/JJ, system/NN, of/IN, the/DT, luminaire/NN, shall/MD, consist/VB, of/IN, lamp/NN, socket/NN, support/NN, ,/,, terminal/NN, block/NN, ,/,, secondary/JJ, lightning/NN, arrestor/NN, ,/,, ballast/NN, ,/,, electronic/JJ, starter/NNS, ,/,, and/CC, photoswitch/NN, ./.]

2 [The lamp socket support shall be adjustable.]

[The/DT, lamp/NN, socket/NN, support/NN, shall/MD, be/VB, adjustable/JJ, ./.]

3 [Adjustments of the lamp socket shall be pre-

set at the factory for the type of light distribution on the plans.]

[Adjustments/NNS, of/IN, the/DT, lamp/NN, socket/NN, shall/MD, be/VB, pre-set/JJ, at/IN, the/DT, factory/NN, for/IN, the/DT, type/NN, of/IN, light/JJ, distribution/NN, on/IN, the/DT, plans/NNS, ./.]

4 [The luminaire housing shall include a terminal block with captive stainless hardware and p

ressure pads.]

[The/DT, luminaire/JJ, housing/NN, shall/MD, include/VB, a/DT, terminal/NN, block/NN, with/IN, captive/JJ, stainless/JJ, hardware/NN, and/CC, pressure/NN, pads/NNS, ./.]

5 [A factory installed secondary lightning arrestor shall be included in the luminaire housing]

[A/DT, factory/NN, installed/VBD, secondary/JJ, lightning/NN, arrestor/NN, shall/MD, be/VB, included/VBN, in/IN, the/DT, luminaire/JJ, housing/NN]

15

16 1 [A multi-voltage ballast shall be furnished in the luminaire.]

[A/DT, multi-voltage/JJ, ballast/NN, shall/MD, be/VB, furnished/VBN, in/IN, the/DT, luminaire/NN, ./.]

2 [A ballast shall be the regulated type designed to operate high pressure sodium lamps in an essentially horizontal position at either 120 or 240 volts unless noted otherwise.]

[A/DT, ballast/NN, shall/MD, be/VB, the/DT, regulated/JJ, type/NN, designed/VBN, to/TO, operate/VB, high/JJ, pressure/NN, sodium/NN, lamps/NNS, in/IN, an/DT, essentially/RB, horizontal/JJ, position/NN, at/IN, either/DT, 120/CD, or/CC, 240/CD, volts/NNS, unless/IN, noted/VBD, otherwise/RB, ./.]

3 [Lamp wattage shall average within plus or minus 3 % of wattage measured at nominal line voltage applid to the ballast and shall not have less than 90 % power factor.]

[Lamp/JJ, wattage/NN, shall/MD, average/VB, within/IN, plus/CC, or/CC, minus/CC, 3/CD, %/NN, of/IN, wattage/NN, measured/VBN, at/IN, nominal/JJ, line/NN, voltage/NN, applid/VBN, to/TO, the/DT, ballast/NN, and/CC, shall/MD, not/RB, have/VB, less/JJR, than/IN, 90/CD, %/NN, power/NN, factor/NN, ./.]

4 [Each ballast shall permanently and clearly indicate the type , catalog number , voltage rati ng , and a connection diagram]

[Each/DT, ballast/NN, shall/MD, permanently/RB, and/CC, clearly/RB, indicate/VB, the/DT, type/NN, ,/,, catalog/NN, number/NN, ,/,, voltage/JJ, rating/NN, ,/,, and/CC, a/DT, connection/NN, diagram/NN]

17

18 1 [The luminaire shall be equipped with an electronic starter.]

[The/DT, luminaire/NN, shall/MD, be/VB, equipped/VBN, with/IN, an/DT, electronic/JJ, starter/NN, ./.]

2 [The electronic starter shall be a solid state device capable of withstanding ambient temper atures of 85 degrees Celsius.]

[The/DT, electronic/JJ, starter/NN, shall/MD, be/VB, a/DT, solid/JJ, state/NN, device/NN, capable/JJ, of/IN, withstanding/JJ, ambient/NN, temperatures/NNS, of/IN, 85/CD, degrees/NNS, Celsius/NNS, ./.]

3 [All components of the electronic starter shall be sealed and protected from dirt , moisture

or other foreign material.]

[All/DT, components/NNS, of/IN, the/DT, electronic/JJ, starter/NN, shall/MD, be/VB, sealed/VBN, and/CC, protected/VBN, from/IN, dirt/NN, ,/,, moisture/NN, ,/,, or/CC, other/JJ, foreign/JJ, material/NN, ./.]

4 [The electronic starter shall have a minimum pulse repetition rate of one pulse per cycle.]

[The/DT, electronic/JJ, starter/NN, shall/MD, have/VB, a/DT, minimum/JJ, pulse/NN, repetition/NN, rate/NN, of/IN, one/CD, pulse/NN, per/IN, cycle/NN, ./.]

5 [The minimum amplitude of the pulse of the electronic starter shall be 2,500 volts for 100 -
through 400-watt lamps and 3,000 volts for 1,000-watt lamps.]

[The/DT, minimum/JJ, amplitude/NN, of/IN, the/DT, pulse/NN, of/IN, the/DT, electronic/JJ, starter/NN, shall/MD, be/VB, 2,500/CD, volts/NNS, for/IN, 100/CD, -/:, through/IN, 400-watt/JJ, lamps/NNS, and/CC, 3,000/CD, volts/NNS, for/IN, 1,000-watt/JJ, lamps/NNS, ./.]

6 [The minimum amplitude of the pulse of the electronic starter shall be applied within 20 ele
ctrical degrees of the center of the open circuit voltage wave]

[The/DT, minimum/JJ, amplitude/NN, of/IN, the/DT, pulse/NN, of/IN, the/DT, electronic/JJ, starter/NN, shall/MD, be/VB, applied/VBN, within/IN, 20/CD, electrical/JJ, degrees/NNS, of/IN, the/DT, center/NN, of/IN, the/DT, open/JJ, circuit/NN, voltage/NN, wave/NN]

19

20 1 [If specified in the order or on the plans , the luminaire shall be equipped with a photo-electric cell switch mounted on the top of the luminaire housing.]

[If/IN, specified/VBN, in/IN, the/DT, order/NN, or/CC, on/IN, the/DT, plans/NNS, ,/,, the/DT, luminaire/NN, shall/MD, be/VB, equipped/VBN, with/IN, a/DT, photo-electric/JJ, cell/NN, switch/NN, mounted/VBN, on/IN, the/DT, top/NN, of/IN, the/DT, luminaire/JJ, housing/NN, ./.]

2 [The photo-electric cell switch shall be a 3-terminal , polarized , locking type conforming to NEMA standards.]

[The/DT, photo-electric/JJ, cell/NN, switch/NN, shall/MD, be/VB, a/DT, 3-terminal/JJ, ,/,, polarized/JJ, ,/,, locking/JJ, type/NN, conforming/VBG, to/TO, NEMA/NNP, standards/NNS, ./.]

3 [The luminaire mechanical system shall consist of housing , slipfitter , and hardware.]

[The/DT, luminaire/NN, mechanical/JJ, system/NN, shall/MD, consist/VB, of/IN, housing/NN, ,/,, slipfitter/NN, ,/,, and/CC, hardware/NN, ./.]

4 [The luminaire housing shall have an upper and a lower portion which shall be die-cast aluminum joined by an integrally cast pin hinge at the mounting end and a one-hand latch at the openingend.]

[The/DT, luminaire/JJ, housing/NN, shall/MD, have/VB, an/DT, upper/JJ, and/CC, a/DT, lower/JJR, portion/NN, which/WDT, shall/MD, be/VB, die-cast/JJ, aluminum/NN, joined/VBN, by/IN, an/DT, integrally/JJ, cast/NN, pin/NN, hinge/VB, at/IN, the/DT, mounting/VBG, end/NN, and/CC, a/DT, one-hand/JJ, latch/NN, at/IN, the/DT, opening/NN, end/NN, ./.]

5 [The housing shall be equipped with a gasket made of high temperature polyester fiber to fi

lter air entering the optical assembly.]

[The/DT, housing/NN, shall/MD, be/VB, equipped/VBN, with/IN, a/DT, gasket/NN, made/VBN, of/IN, high/JJ, temperature/NN, polyester/NN, fiber/NN, to/TO, filter/VB, air/NN, entering/VBG, the/DT, optical/JJ, assembly/NN, ./.]

6 [The housing shall be finished with baked-on acrylic enamel having a color equivalent to American Electric Statuary Bronze.]

[The/DT, housing/NN, shall/MD, be/VB, finished/VBN, with/IN, baked-on/JJ, acrylic/JJ, enamel/NN, having/VBG, a/DT, color/JJ, equivalent/JJ, to/TO, American/NNP, Electric/NNP, Statuary/NNP, Bronze/NN, ./.]

21 1 [The luminaire shall be equipped with a slipfit mounting bracket that shall be adaptable to 1.25 " and 2.00 " diameter arms.]

[The/DT, luminaire/NN, shall/MD, be/VB, equipped/VBN, with/IN, a/DT, slipfit/NN, mounting/VBG, bracket/NN, that/WDT, shall/MD, be/VB, adaptable/JJ, to/TO, 1.25/CD, "/', and/CC, 2.00/CD, "/', diameter/VBG, arms/NNS, ./.]

2 [All hardware shall be corrosion-resistant]

[All/DT, hardware/NN, shall/MD, be/VB, corrosion-resistant/JJ]

22

23 1 [Each ballast and luminaire shall be factory tested with a nominal , center-rated lamp.]

[Each/DT, ballast/NN, and/CC, luminaire/NN, shall/MD, be/VB, factory/NN, tested/VBN, with/IN, a/DT, nominal/JJ, ,/,, center-rated/JJ, lamp/NN, ./.]

2 [The lamp used for testing shall have been `` burned in " for a period of at least 100 hours.]

[The/DT, lamp/NN, used/VBN, for/IN, testing/NN, shall/MD, have/VB, been/VBN, ``/``, burned/VBN, in/RP, "/', for/IN, a/DT, period/NN, of/IN, at/IN, least/JJS, 100/CD, hours/NNS, ./.]

3 [Each lighting system controller shall be equivalent to South Bend Controls , Incorporated , Remote Control Outdoor Circuit model MR-
UG , 30 or 60 ampere , single or double pole relay rated forvolt control.]

[Each/DT, lighting/NN, system/NN, controller/NN, shall/MD, be/VB, equivalent/JJ, to/TO, South/NNP, Bend/NNP, Controls/NNP, ,/,, Incorporated/NNP, ,/,, Remote/NNP, Control/NNP, Outdoor/NNP, Circuit/NNP, model/NN, MR-UG/NN, ,/,, 30/CD, or/CC, 60/CD, ampere/JJ, ,/,, single/JJ, or/CC, double/JJ, pole/NN, relay/RB, rated/VBN, for/IN, volt/NN, control/NN, ./.]

4 [The lighting system controller shall include a cast aluminum outdoor cabinet]

[The/DT, lighting/NN, system/NN, controller/NN, shall/MD, include/VB, a/DT, cast/NN, aluminum/NN, outdoor/JJ, cabinet/NN]

24

25 1 [The outdoor cabinet shall be finished with acrylic enamel.]

[The/DT, outdoor/JJ, cabinet/NN, shall/MD, be/VB, finished/VBN, with/IN, acrylic/JJ, enamel/NN, ./.]

2 [The Contractor shall furnish in-
line fuses , disconnecting fuse holders and disconnect kits.]

[The/DT, Contractor/NNP, shall/MD, furnish/VB, in-line/JJ, fuses/NNS, ,/,, disconnecting/VBG, fuse/NN, holders/NNS, and/CC, disconnect/VB, kits/NNS, ./.]

3 [When wire and cable is provided on reels , the reels shall be non-returnable.]

[When/WRB, wire/NN, and/CC, cable/NN, is/VBZ, provided/VBN, on/IN, reels/NNS, ,/,, the/DT, reels/NNS, shall/MD, be/VB, non-returnable/JJ, ./.]

4 [The wire and cable shall be furnished in the appropriate American Wire Gage -LRB- A.W.G. -RRB- sizes shown on the plans or in the order.]

[The/DT, wire/NN, and/CC, cable/NN, shall/MD, be/VB, furnished/VBN, in/IN, the/DT, appropriate/JJ, American/NNP, Wire/NNP, Gage/NNP, -LRB-/-LRB-, A.W.G./NNP, -RRB-/-RRB-, sizes/NNS, shown/VBN, on/IN, the/DT, plans/NNS, or/CC, in/IN, the/DT, order/NN, ./.]

5 [The types of wire and cable are ground wire , streetlight cable for direct burial , and street

ight cable for aerial.]

[The/DT, types/NNS, of/IN, wire/NN, and/CC, cable/NN, are/VBP, ground/NN, wire/NN, ,/,, streetlight/JJ, cable/NN, for/IN, direct/JJ, burial/NN, ,/,, and/CC, streetlight/JJ, cable/NN, for/IN, aerial/NNP, ./.]

6 [Ground wire shall be used to connect poles and other devices to grounding electrodes -LRB- ground rods -RRB-.]

[Ground/NN, wire/NN, shall/MD, be/VB, used/VBN, to/TO, connect/VB, poles/NNS, and/CC, other/JJ, devices/NNS, to/TO, grounding/NN, electrodes/NNS, -LRB-/-LRB-, ground/NN, rods/NNS, -RRB-/-RRB-, ./.]

7 [Ground wire shall be bare soft drawn copper wire having a size of A.W.G. # 6 unless othe

rwise noted.]

[Ground/NN, wire/NN, shall/MD, be/VB, bare/JJ, soft/JJ, drawn/JJ, copper/NN, wire/NN, having/VBG, a/DT, size/NN, of/IN, A.W.G./NNP, #/#, 6/CD, unless/IN, otherwise/RB, noted/VBD, ./.]

8 [Streetlight cable shall be used to connect streetlights to their power source or controller.]

[Streetlight/JJ, cable/NN, shall/MD, be/VB, used/VBN, to/TO, connect/VB, streetlights/NNS, to/TO, their/PRP$, power/NN, source/NN, or/CC, controller/NN, ./.]

9 [Direct burial streetlight cable shall be copper 3-

wire , two insulated and one bare wire , consisting of a complete assembly of conductors o

the types and sizes specified.]

[Direct/JJ, burial/NN, streetlight/NN, cable/NN, shall/MD, be/VB, copper/NN, 3-

wire/NN, ,/,, two/CD, insulated/NNS, and/CC, one/CD, bare/JJ, wire/NN, ,/,, consisting/VBG, of/IN, a/DT, complete/JJ, assembly/NN, of/IN, conductors/NNS, of/IN, the/DT, types/NNS, and/CC, sizes/NNS, specified/VBN, ./.]

10 [Bare wire grounding conductors shall be annealed , uncoated copper conforming to the N

ATIONAL ELECTRICAL CODE]

[Bare/JJ, wire/NN, grounding/NN, conductors/NNS, shall/MD, be/VB, annealed/VBN, ,/,, uncoated/JJ, copper/NN, conforming/VBG, to/TO, the/DT, NATIONAL/NNP, ELECTRICAL/NNP, CODE/NNP]

26

27 1 [Insulated conductors shall be stranded annealed , coated or uncoated copper conforming t

the NATIONAL ELECTRICAL CODE Type XHHW.]

[Insulated/JJ, conductors/NNS, shall/MD, be/VB, stranded/VBN, annealed/JJ, ,/,, coated/JJ, or/CC, uncoated/JJ, copper/NN, conforming/NNS, to/TO, the/DT, NATIONAL/NNP, ELECTRICAL/NNP, CODE/NNP, Type/NNP, XHHW/NNP, ./.]

2 [Insulated conductors shall be color coded for circuit identification.]

[Insulated/JJ, conductors/NNS, shall/MD, be/VB, color/NN, coded/VBN, for/IN, circuit/NN, identification/NN, ./.]

3 [The individual conductors shall be marked in accordance with NATIONAL ELECTRICA L CODE standards]

[The/DT, individual/JJ, conductors/NNS, shall/MD, be/VB, marked/VBN, in/IN, accordance/NN, with/IN, NATIONAL/NNP, ELECTRICAL/NNP, CODE/NNP, standards/NNS]

28 1 [The ground wire for all circuits shall be # 6 bare]

[The/DT, ground/NN, wire/NN, for/IN, all/DT, circuits/NNS, shall/MD, be/VB, #/#, 6/CD bare/CD]

29

30 1 [The streetlight cable shall be the size indicated on the plans or in the contract documents.]

[The/DT, streetlight/NN, cable/NN, shall/MD, be/VB, the/DT, size/NN, indicated/VBN, on/IN, the/DT, plans/NNS, or/CC, in/IN, the/DT, contract/NN, documents/NNS, ./.]

2 [The complete assembly shall be packaged on reels having sufficient diameter to prevent in ducing permanent set or injury to the cable.]

[The/DT, complete/JJ, assembly/NN, shall/MD, be/VB, packaged/VBN, on/IN, reels/NNS having/VBG, sufficient/JJ, diameter/NN, to/TO, prevent/VB, inducing/VBG, permanent/JJ, set/NN, or/CC, injury/NN, to/TO, the/DT, cable/NN, ./.]

3 [Each reel shall be adequately labeled to indicate the voltage , the insulation type , the num ber and size of conductors , the length of cable on the reel , and the trade name of the manu facturer.]

[Each/DT, reel/NN, shall/MD, be/VB, adequately/RB, labeled/VBN, to/TO, indicate/VB, the/DT, voltage/NN, ,/,, the/DT, insulation/NN, type/NN, ,/,, the/DT, number/NN, and/CC, size/NN, of/IN, conductors/NNS, ,/,, the/DT, length/NN, of/IN, cable/NN, on/IN, the/DT, reel/NN, ,/,, and/CC, the/DT, trade/NN, name/NN, of/IN, the/DT, manufacturer/NN, ./.]

4 [Aerial streetlight cable shall not be allowed for arterial lighting]

[Aerial/JJ, streetlight/NN, cable/NN, shall/MD, not/RB, be/VB, allowed/VBN, for/IN, arterial/JJ, lighting/NN]

APPENDIX C. NLP TOOL SOURCE CODE (1) – PARAGRAPH SPLITTER CLASS

The following listing shows the source code of the Paragraph Splitter class of the system.

```
public class Paragraph {

    private static final String parser = "edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz";

    private LexicalizedParser lp = LexicalizedParser.loadModel(parser);


    // constants for finding nearest sentence boundary

    private static final int SEEK_FORWARD = 1;

    private static final int SEEK_BACK = -1;


    private int index = 0;

    private String paragraph = "";

    private int NoOfSentences = 0;


    private LinkedList<Sentence> llSentences = new LinkedList<Sentence>();


    public Paragraph(int index, String paragraph) {

        this.index = index;

        this.paragraph = paragraph;


        this.prep();

    }
```

```
public Paragraph(int index, String paragraph, String parser) {
    this.index = index;
    this.paragraph = paragraph;
    this.lp = LexicalizedParser.loadModel(parser);

    this.prep();
}

public Paragraph(int index, String paragraph, LexicalizedParser parser) {
    this.index = index;
    this.paragraph = paragraph;
    this.lp = parser;

    this.prep();
}

private void prep() {
    int startIndex = 0;
    int endIndex = 0;
    String sentence = "";

    this.NoOfSentences = 0;
```

```
        while (endIndex < this.paragraph.length()) {
            this.NoOfSentences++;
            endIndex = this.nearestDelimiter(this.paragraph, startIndex, SEEK_FORWARD);
            sentence = this.paragraph.substring(startIndex, endIndex);
            this.llSentences.add(new Sentence(this.NoOfSentences, sentence, this.lp));
            endIndex++;
            startIndex = endIndex++;
        }
    }

    public int getParagraphIndex() {
        return this.index;
    }

    public int getNoOfSentences() {
        return this.NoOfSentences;
    }

    public String toString() {
        return this.paragraph;
    }

    public String toTaggedString() {
        String out = "";


        for (int index = 0; index < this.llSentences.size(); index++) {

            Sentence sentence = this.llSentences.get(index);

            out += sentence.toTaggedString() + "\n";

        }


        return out;

    }


    public LinkedList<Sentence> getSentences() {

        return this.llSentences;

    }


    private int nearestDelimiter(String text, int start, int seekDir) {

        if (seekDir != SEEK_BACK && seekDir != SEEK_FORWARD) {

            throw new IllegalArgumentException("Unknown seek direction " + seekDir);

        }


        Reader reader = new StringReader(text);

        DocumentPreprocessor dp = new DocumentPreprocessor(reader);

        List<Integer> boundaries = new ArrayList<Integer>();
    for (List<HasWord> sentence : dp) {
        if (sentence.size() == 0)
            continue;

        if (!(sentence.get(0) instanceof HasOffset)) {
            throw new ClassCastException("Expected HasOffsets from the
DocumentPreprocessor");
        }

        if (boundaries.size() == 0) {
            boundaries.add(0);
        } else {
            HasOffset first = (HasOffset) sentence.get(0);
            boundaries.add(first.beginPosition());
        }
    }

    boundaries.add(text.length());

    for (int i = 0; i < boundaries.size() - 1; ++i) {
        if (boundaries.get(i) <= start && start < boundaries.get(i + 1)) {
            if (seekDir == SEEK_BACK) {
                return boundaries.get(i) - 1;
                } else if (seekDir == SEEK_FORWARD) {

                    return boundaries.get(i + 1) - 1;

                }

            }

        }


        // The cursor position at the end is actually one past the text length.

        // We might highlight the last interval in that case.

        if (boundaries.size() >= 2 && start >= text.length()) {

            if (seekDir == SEEK_BACK) {

                return boundaries.get(boundaries.size() - 2) - 1;

            } else if (seekDir == SEEK_FORWARD) {

                return boundaries.get(boundaries.size() - 1) - 1;

            }

        }


        return -1;

    }


}
```

APPENDIX D. NLP TOOL SOURCE CODE (2) – SENTENCE SPLITTER CLASS

The following listing shows the source code of the Sentence Splitter class of the system.

```
public class Sentence {

    private final String parser = "edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz";

    private LexicalizedParser lp;


    private int index = 0;

    private String sentence = "";

    private ArrayList<TaggedWord> taggedSentence;


    private LinkedList<String> llWords = new LinkedList<String>();

    private LinkedList<String> llTags = new LinkedList<String>();

    private LinkedList<String> refinedwords = new LinkedList<String>();

    private LinkedList<String> refinedtags = new LinkedList<String>();


    public Sentence(int index, String sentence) {

        this.index = index;

        this.sentence = sentence;

        this.lp = LexicalizedParser.loadModel(this.parser);


        this.prep();

    }
```

```
public Sentence(int index, String sentence, String parser) {
    this.index = index;
    this.sentence = sentence;
    this.lp = LexicalizedParser.loadModel(parser);

    this.prep();
}

public Sentence(int index, String sentence, LexicalizedParser parser) {
    this.index = index;
    this.sentence = sentence;
    this.lp = parser;

    this.prep();
}

private void prep() {
    setTaggedSentence();

    for (TaggedWord tagword : this.taggedSentence) {
        // Tag Index
        int tagIndex = tagword.toString().indexOf("/");
```

```
            // Word and Tag
            this.llWords.add(tagword.toString().substring(0, tagIndex));
            this.llTags.add(tagword.tag());
        }

        this.refine();
    }

    private void refine() {
        int size = this.llTags.size();

        for (int index = 0; index < size; index++) {
            boolean pass = false;
            String tag = this.getTag(index);
            String tempnoun = "";

            String [] prefix = {"NN", "VB", "RB", "JJ"};
            int i = 0;
            for (i=0; i < prefix.length; i++) {
                if (tag.startsWith(prefix[i])) {
                    pass = true;
                    break;
                }
            }


            if (i == 0 && pass) {
                // The first noun if there are following nouns
                tempnoun = this.getWord(index);


                // To combine consecutive nouns into one word
                while ((index + 1 < size) && this.getTag(index + 1).startsWith("NN")) {
                    index++;
                    tempnoun += " " + this.getWord(index);
                }
            }


            if (pass) {
                // Refined Word and Tag
                this.refinedwords.add((i == 0) ? tempnoun : this.getWord(index));
                this.refinedtags.add(this.getTag(index));
            }
        }
    }


    private void setTaggedSentence() {
        TokenizerFactory<CoreLabel> tokenizerFactory = PTBTokenizer.factory(new
CoreLabelTokenFactory(), "");
        List<CoreLabel> rawWords2 = tokenizerFactory.getTokenizer(new
StringReader(this.sentence)).tokenize();
        Tree parse = this.lp.apply(rawWords2);
        this.taggedSentence = parse.taggedYield();
    }

    public int getSentenceIndex() {
        return this.index;
    }

    public String toString() {
        return this.sentence;
    }

    public String toTaggedString() {
        return this.taggedSentence.toString();
    }

    public String toRefinedString() {
        String sentence = "";

        int size = this.refinedwords.size();
        for (int index = 0; index < size; index++) {
            sentence += sentence.equals("") ? "" : ", ";
            sentence += this.getRefinedWord(index);
            sentence += "/";
            sentence += this.getRefinedTag(index);
        }

        return sentence;
    }

    public ArrayList<TaggedWord> getTaggedSentence() {
        return this.taggedSentence;
    }

    public String getWord(int index) {
        String word = "";

        try {
            word = this.llWords.get(index);
        } catch (Exception e) {
            word = "";
        }
        return word;
    }

    public String getRefinedWord(int index) {
        String word = "";

        try {
            word = this.refinedwords.get(index);
        } catch (Exception e) {
            word = "";
        }

        return word;
    }

    public LinkedList<String> getWords() {
        return this.llWords;
    }

    public LinkedList<String> getRefinedWords() {
        return this.refinedwords;
    }
```

```
public String printWords() {

    return this.llWords.toString();

}


public String printRefinedWords() {

    return this.refinedwords.toString();

}


public String getTag(int index) {

    String tag = "";


    try {

        tag = this.llTags.get(index);

    } catch (Exception e) {

        tag = "";

    }


    return tag;

}


public String getRefinedTag(int index) {

    String tag = "";
```

```
        try {
            tag = this.refinedtags.get(index);
        } catch (Exception e) {
            tag = "";
        }

        return tag;
    }

    public LinkedList<String> getTags() {
        return this.llTags;
    }

    public LinkedList<String> getRefinedTags() {
        return this.refinedtags;
    }

    public String printTags() {
        return this.llTags.toString();
    }

    public String printRefinedTags() {
        return this.refinedtags.toString();
    }


}
```

APPENDIX E. NLP TOOL SOURCE CODE (3) – WORD PARSER CLASS

The following listing shows the source code of the Word Parser class of the system.

```
public class WordParser {

    private String word = "";

    private String stem = "";


    public WordParser() {

        this.word = "";

    }


    public WordParser(String word) {

        this.word = word;

    }


    private void getStem(POS pos) throws IOException {

        // Construct the URL to the WordNet dictionary directory

        String wnhome = System.getenv("WNHOME");

        String path = wnhome + File.separator + "dict";

        URL url = new URL("file", null, path);


        // Construct the dictionary object and open it

        IDictionary dict = new Dictionary(url);

        dict.open();
        WordnetStemmer wnstemmer = new WordnetStemmer(dict);

        List<String> stems = wnstemmer.findStems(this.word, null);

        int cntStems = stems.size();


        for (String stem : stems) {

            try {

                IIndexWord idxWord = dict.getIndexWord(stem, pos);

                IWordID wordID = idxWord.getWordIDs().get(0);

                IWord word = dict.getWord(wordID);


                this.stem = word.getLemma();

            } catch (Exception e) {

                System.out.println(e.getMessage() + "\t" + pos);

            }

        }


        dict.close();

    }


    public String getWord() {

        return this.word;

    }
    public String getStem() {
        return this.stem;
    }
}
```

APPENDIX F. NLP TOOL SOURCE CODE (4) – WORD TAG CHECKER CLASS

The following listing shows the source code of the Word Tag Checker class of the system.

```
public class WordTagChecker {

    private String word;

    private String tag;


    private String [] NOUN = {"NN", "NNP", "NNPS", "NNS"};

    private String [] VERB = {"VB", "VBD", "VBG", "VBN", "VBP", "VBZ"};

    private String [] ADJ = {"JJ", "JJR", "JJS"};

    private String [] ADV = {"RB", "RBR", "RBS"};


    private POS posnoun = POS.NOUN;

    private POS posverb = POS.VERB;

    private POS posadj = POS.ADJECTIVE;

    private POS posadv = POS.ADVERB;


    public WordTagChecker(String word, String tag) {

        this.word = word;

        this.tag = tag;

    }


    public String getWord() {

        return this.word;
```

```
}


public String getTag() {
    return this.tag;
}


public POS getPOS() {
    POS wordclass = null;


    for (String pos : this.NOUN) {
        if (this.tag.equals(pos)) {
            wordclass = this.posnoun;
            break;
        }
    }


    if (wordclass == null) {
        for (String pos : this.VERB) {
            if (this.tag.equals(pos)) {
                wordclass = this.posverb;
                break;
            }
        }
```

```
            if (wordclass == null) {
                for (String pos : this.ADJ) {
                    if (this.tag.equals(pos)) {
                        wordclass = this.posadj;
                        break;
                    }
                }

                if (wordclass == null) {
                    for (String pos : this.ADV) {
                        if (this.tag.equals(pos)) {
                            wordclass = this.posadv;
                            break;
                        }
                    }
                }
            }
        }

        return wordclass;
    }
```

```
public String getStem() {

    String stemword = this.word;


    if (getPOS() != null) {

        try {

            // Construct the URL to the WordNet dictionary directory

            String wnhome = System.getenv("WNHOME");

            String path = wnhome + File.separator + "dict";


            // Open the URL

            URL url = new URL("file", null, path);


            // WordNet Dictionary

            IDictionary dict = new Dictionary(url);


            try {

                // Open the dictionary object

                dict.open();


                WordnetStemmer wnstemmer = new WordnetStemmer(dict);

                List<String> stems = wnstemmer.findStems(this.word, getPOS());


                for (int index = 0; index < stems.size(); index++) {
```

```
                                if (index == 0) {
                                    stemword = stems.get(index).toString();
                                    break;
                                }
                            }
                        } catch (IOException e) {
                            System.out.println("WordTagChecker.getStem(): Loading the WordNet
dictionary has been failed...");
                            e.printStackTrace();
                        } finally {
                            dict.close();
                        }
                    } catch (MalformedURLException e) {
                        System.out.println("WordTagChecker.getStem(): The URL for the dictionary is
incorrect...");
                        e.printStackTrace();
                    }
                }

                return stemword;
            }

}
```

APPENDIX G. GEXF FILE OF A GRID-CONNECTED SOLAR PV SYSTEM

The following listing represents a graph exchange XML format of a grid-connected solar PV system for the visualization of its interaction network.

<?xml version='1.0' encoding='UTF-8'?><gexf xmlns="http://www.gexf.net/1.2draft" xmlns:viz="http://www.gexf.net/1.2draft/viz" version="1.2"><meta lastmodifieddate="2012-07-12"><creator>CInDI.pro</creator><description>Interaction Network: Grid-Connected Solar PV System</description></meta><graph defaultedgetype="undirected" idtype="string" mode="static"><attributes class="node" mode="static"><attribute id="0" title="description" type="string" /></attributes><nodes count="29"><node id="0" label="S1:home"><attvalues><attvalue for="0" value="home" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="1" label="S2:panel"><attvalues><attvalue for="0" value="panel" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="2" label="S3:PV system"><attvalues><attvalue for="0" value="PV system" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="3" label="S4:utility grid"><attvalues><attvalue for="0" value="utility grid" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="4" label="S5:grid"><attvalues><attvalue for="0" value="grid" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="5" label="S6:inverter"><attvalues><attvalue for="0" value="inverter" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="6" label="S7:meter"><attvalues><attvalue for="0" value="meter" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="7" label="S8:power"><attvalues><attvalue for="0" value="power" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="8" label="S9:utility"><attvalues><attvalue for="0" value="utility" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="9" label="S10:ac breaker panel"><attvalues><attvalue for="0" value="ac breaker panel" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="10" label="S11:array"><attvalues><attvalue for="0" value="array" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="11"

label="S12:wiring"><attvalues><attvalue for="0" value="wiring" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="12" label="S13:array dc disconnect"><attvalues><attvalue for="0" value="array dc disconnect" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="13" label="S14:battery"><attvalues><attvalue for="0" value="battery" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="14" label="S15:dc disconnect"><attvalues><attvalue for="0" value="dc disconnect" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="15" label="S16:panel system"><attvalues><attvalue for="0" value="panel system" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="16" label="S17:PCU"><attvalues><attvalue for="0" value="PCU" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="17" label="S18:power conditioning unit"><attvalues><attvalue for="0" value="power conditioning unit" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="18" label="S19:power grid"><attvalues><attvalue for="0" value="power grid" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="19" label="S20:Solar panel"><attvalues><attvalue for="0" value="Solar panel" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="20" label="S21:sunlight"><attvalues><attvalue for="0" value="sunlight" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="21" label="S22:utility bill"><attvalues><attvalue for="0" value="utility bill" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="22" label="S23:utility disconnect"><attvalues><attvalue for="0" value="utility disconnect" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="23" label="S24:energy"><attvalues><attvalue for="0" value="energy" /></attvalues><viz:color r="255" g="69" b="0" /></node><node id="24" label="F1"><attvalues><attvalue for="0" value="To collect and distribute clean, renewable energy" /></attvalues><viz:color r="0" g="0" b="205" /></node><node id="25" label="F2"><attvalues><attvalue for="0" value="To convert direct current (DC) electricity to alternating current (AC) electricity" /></attvalues><viz:color r="0" g="0" b="205" /></node><node id="26" label="F3"><attvalues><attvalue for="0" value="To stop the flow of electricity from solar panels" /></attvalues><viz:color r="0" g="0" b="205" /></node><node id="27"

label="F4"><attvalues><attvalue for="0" value="To perform maintenance on the utility grid" /></attvalues><viz:color r="0" g="0" b="205" /></node><node id="28" label="F5"><attvalues><attvalue for="0" value="To calculate monthly utility bill" /></attvalues><viz:color r="0" g="0" b="205" /></node></nodes><edges count="172"><edge id="0" source="0" target="1" type="undirected" /><edge id="1" source="0" target="2" type="undirected" /><edge id="2" source="0" target="3" type="undirected" /><edge id="3" source="0" target="4" type="undirected" /><edge id="4" source="0" target="5" type="undirected" /><edge id="5" source="0" target="6" type="undirected" /><edge id="6" source="0" target="7" type="undirected" /><edge id="7" source="0" target="9" type="undirected" /><edge id="8" source="0" target="11" type="undirected" /><edge id="9" source="0" target="14" type="undirected" /><edge id="10" source="0" target="19" type="undirected" /><edge id="11" source="0" target="22" type="undirected" /><edge id="12" source="0" target="23" type="undirected" /><edge id="13" source="1" target="0" type="undirected" /><edge id="14" source="1" target="2" type="undirected" /><edge id="15" source="1" target="3" type="undirected" /><edge id="16" source="1" target="4" type="undirected" /><edge id="17" source="1" target="12" type="undirected" /><edge id="18" source="1" target="18" type="undirected" /><edge id="19" source="1" target="20" type="undirected" /><edge id="20" source="1" target="23" type="undirected" /><edge id="21" source="2" target="0" type="undirected" /><edge id="22" source="2" target="1" type="undirected" /><edge id="23" source="2" target="3" type="undirected" /><edge id="24" source="2" target="5" type="undirected" /><edge id="25" source="2" target="6" type="undirected" /><edge id="26" source="2" target="9" type="undirected" /><edge id="27" source="2" target="11" type="undirected" /><edge id="28" source="2" target="12" type="undirected" /><edge id="29" source="2" target="14" type="undirected" /><edge id="30" source="2" target="16" type="undirected" /><edge id="31" source="2" target="17" type="undirected" /><edge id="32" source="2" target="19" type="undirected" /><edge id="33" source="2" target="22" type="undirected" /><edge id="34" source="3" target="0" type="undirected" /><edge id="35" source="3" target="1" type="undirected" /><edge id="36" source="3" target="2" type="undirected" /><edge id="37"

source="3" target="5" type="undirected" /><edge id="38" source="3" target="7" type="undirected" /><edge id="39" source="3" target="8" type="undirected" /><edge id="40" source="3" target="10" type="undirected" /><edge id="41" source="3" target="23" type="undirected" /><edge id="42" source="4" target="0" type="undirected" /><edge id="43" source="4" target="1" type="undirected" /><edge id="44" source="4" target="7" type="undirected" /><edge id="45" source="4" target="8" type="undirected" /><edge id="46" source="4" target="18" type="undirected" /><edge id="47" source="4" target="23" type="undirected" /><edge id="48" source="5" target="0" type="undirected" /><edge id="49" source="5" target="2" type="undirected" /><edge id="50" source="5" target="3" type="undirected" /><edge id="51" source="5" target="6" type="undirected" /><edge id="52" source="5" target="9" type="undirected" /><edge id="53" source="5" target="11" type="undirected" /><edge id="54" source="5" target="14" type="undirected" /><edge id="55" source="5" target="16" type="undirected" /><edge id="56" source="5" target="17" type="undirected" /><edge id="57" source="5" target="19" type="undirected" /><edge id="58" source="5" target="22" type="undirected" /><edge id="59" source="5" target="23" type="undirected" /><edge id="60" source="6" target="0" type="undirected" /><edge id="61" source="6" target="2" type="undirected" /><edge id="62" source="6" target="5" type="undirected" /><edge id="63" source="6" target="7" type="undirected" /><edge id="64" source="6" target="9" type="undirected" /><edge id="65" source="6" target="10" type="undirected" /><edge id="66" source="6" target="11" type="undirected" /><edge id="67" source="6" target="14" type="undirected" /><edge id="68" source="6" target="19" type="undirected" /><edge id="69" source="6" target="21" type="undirected" /><edge id="70" source="6" target="22" type="undirected" /><edge id="71" source="7" target="0" type="undirected" /><edge id="72" source="7" target="3" type="undirected" /><edge id="73" source="7" target="4" type="undirected" /><edge id="74" source="7" target="6" type="undirected" /><edge id="75" source="7" target="8" type="undirected" /><edge id="76" source="7" target="10" type="undirected" /><edge id="77" source="7" target="21" type="undirected" /><edge id="78" source="7" target="23" type="undirected" /><edge id="79" source="8" target="3" type="undirected" /><edge id="80" source="8" target="4"

type="undirected" /><edge id="81" source="8" target="7" type="undirected" /><edge id="82"
source="8" target="23" type="undirected" /><edge id="83" source="9" target="0" type="undirected"
/><edge id="84" source="9" target="2" type="undirected" /><edge id="85" source="9" target="5"
type="undirected" /><edge id="86" source="9" target="6" type="undirected" /><edge id="87"
source="9" target="11" type="undirected" /><edge id="88" source="9" target="14" type="undirected"
/><edge id="89" source="9" target="19" type="undirected" /><edge id="90" source="9" target="22"
type="undirected" /><edge id="91" source="10" target="3" type="undirected" /><edge id="92"
source="10" target="6" type="undirected" /><edge id="93" source="10" target="7" type="undirected"
/><edge id="94" source="10" target="21" type="undirected" /><edge id="95" source="11" target="0"
type="undirected" /><edge id="96" source="11" target="2" type="undirected" /><edge id="97"
source="11" target="5" type="undirected" /><edge id="98" source="11" target="6" type="undirected"
/><edge id="99" source="11" target="9" type="undirected" /><edge id="100" source="11" target="14"
type="undirected" /><edge id="101" source="11" target="19" type="undirected" /><edge id="102"
source="11" target="22" type="undirected" /><edge id="103" source="12" target="1" type="undirected"
/><edge id="104" source="12" target="2" type="undirected" /><edge id="105" source="13" target="23"
type="undirected" /><edge id="106" source="14" target="0" type="undirected" /><edge id="107"
source="14" target="2" type="undirected" /><edge id="108" source="14" target="5" type="undirected"
/><edge id="109" source="14" target="6" type="undirected" /><edge id="110" source="14" target="9"
type="undirected" /><edge id="111" source="14" target="11" type="undirected" /><edge id="112"
source="14" target="19" type="undirected" /><edge id="113" source="14" target="22"
type="undirected" /><edge id="114" source="16" target="2" type="undirected" /><edge id="115"
source="16" target="5" type="undirected" /><edge id="116" source="16" target="17" type="undirected"
/><edge id="117" source="17" target="2" type="undirected" /><edge id="118" source="17" target="5"
type="undirected" /><edge id="119" source="17" target="16" type="undirected" /><edge id="120"
source="18" target="1" type="undirected" /><edge id="121" source="18" target="4" type="undirected"
/><edge id="122" source="18" target="23" type="undirected" /><edge id="123" source="19" target="0"

type="undirected" /><edge id="124" source="19" target="2" type="undirected" /><edge id="125" source="19" target="5" type="undirected" /><edge id="126" source="19" target="6" type="undirected" /><edge id="127" source="19" target="9" type="undirected" /><edge id="128" source="19" target="11" type="undirected" /><edge id="129" source="19" target="14" type="undirected" /><edge id="130" source="19" target="22" type="undirected" /><edge id="131" source="20" target="1" type="undirected" /><edge id="132" source="21" target="6" type="undirected" /><edge id="133" source="21" target="7" type="undirected" /><edge id="134" source="21" target="10" type="undirected" /><edge id="135" source="22" target="0" type="undirected" /><edge id="136" source="22" target="2" type="undirected" /><edge id="137" source="22" target="5" type="undirected" /><edge id="138" source="22" target="6" type="undirected" /><edge id="139" source="22" target="9" type="undirected" /><edge id="140" source="22" target="11" type="undirected" /><edge id="141" source="22" target="14" type="undirected" /><edge id="142" source="22" target="19" type="undirected" /><edge id="143" source="23" target="0" type="undirected" /><edge id="144" source="23" target="1" type="undirected" /><edge id="145" source="23" target="3" type="undirected" /><edge id="146" source="23" target="4" type="undirected" /><edge id="147" source="23" target="5" type="undirected" /><edge id="148" source="23" target="7" type="undirected" /><edge id="149" source="23" target="8" type="undirected" /><edge id="150" source="23" target="13" type="undirected" /><edge id="151" source="23" target="18" type="undirected" /><edge id="171" source="24" target="23" type="undirected"><viz:color r="0" g="255" b="0" /><viz:shape value="dotted" /></edge><edge id="152" source="25" target="0" type="undirected" /><edge id="153" source="25" target="1" type="undirected" /><edge id="154" source="25" target="3" type="undirected" /><edge id="155" source="25" target="4" type="undirected" /><edge id="156" source="25" target="5" type="undirected" /><edge id="157" source="25" target="23" type="undirected" /><edge id="158" source="26" target="1" type="undirected" /><edge id="159" source="26" target="2" type="undirected" /><edge id="160" source="26" target="3" type="undirected" /><edge id="161" source="26" target="8" type="undirected" /><edge id="162" source="26" target="12" type="undirected" /><edge id="163" source="26" target="23" type="undirected" /><edge id="164"

source="27" target="3" type="undirected" /><edge id="165" source="27" target="8" type="undirected" /><edge id="166" source="27" target="23" type="undirected" /><edge id="167" source="28" target="6" type="undirected" /><edge id="168" source="28" target="7" type="undirected" /><edge id="169" source="28" target="10" type="undirected" /><edge id="170" source="28" target="21" type="undirected" /></edges></graph></gexf>

REFERENCES

Abbot L, Quality and Competition, New York: Columbia University Press, 1995.

Abbott RJ, Program design by informal English descriptions, Communications of the ACM, Vol. 26, No. 11, 1983, pp.882–894.

Adner R and Levinthal DA, Technology Speciation and the Path of Emerging Technologies, Wharton on Managing Emerging Technologies, Gary S. Day and Paul J.H. Schoemaker (eds), New York, Chichester: John Wiley, 2000, pp.57-74.

Ahmed S, Kim S, and Wallace KM, A Methodology for Creating Ontologies for Engineering Design, Journal of Computing and Information Science in Engineering, Vol. 7, 2007, pp.132-140.

Al-Safadi LAE, Natural language processing for conceptual modeling, International Journal of Digital Content Technology and its Applications, Vol. 3, No. 3, 2009, pp.47–59.

Allen RH, Nidamarthi S, Regalla SP, and Sriram RD, Enhancing collaboration using an Internet integrated workbench, In Proceedings of DETC99—ASME Design Engineering Technical Conference, Las Vegas, NV, 1999.

Bajwa IS and Choudhary MA, A rule based system for speech language context understanding, Journal of Dong Hua University (English Edition), Vol. 23, No. 6, 2006, pp.39–42.

Bajwa IS, Samad A, and Mumtaz S, Object oriented software modeling using NLP based knowledge extraction, European Journal of Scientific Research, Vol. 35, No. 1, 2009, pp.22–33.

Bastian M, Heymann S, and Jacomy M, Gephi: an open source software for exploring and manipulating networks, International AAAI Conference on Weblogs and Social Media, San Jose, CA, USA, 2009.

Beck K and Cunningham W, A laboratory for teaching object-oriented thinking, In: OOPSLA '89: Conference Proceedings on Object-Oriented Programming Systems, Languages and Applications , New York, NY, USA, ACM, 1989.

Bentley PJ and Wakefield JP, Conceptual Evolutionary Design by a Genetic Algorithm, Engineering Design and Automation, Vol. 2, No. 3, 1996, pp.119-131.

Bettis RA and Hitt MA, The New Competitive Landscape, Strategic Management Journal, Vol. 16, 1995, pp.7-19.

Bettis RA, Comment on "Redefining Industry Structure for the Information Age" by J.L. Sampler, Strategic Management Journal, Vol. 19, 1998, pp.357-362.

Bird S, Klein E, and Loper E, Introduction to Natural Language Processing, University of Pennsylvania, 2007.

Bohm MR, Stone RB, Simpson TW, and Steva ED, Introduction of a data schema to support a design repository, Computer-Aided Design, Vol. 40, No. 7, 2008, pp.801-811.

Booch G, Object-oriented Analysis and Design with Applications, Addison-Wesley Longman Publishing, Inc., 1991.

Booch G, Rumbaugh J, and Jacobson I, The Unified Modeling Language User Guide, Addison-Wesley Object Technology Series, 2nd ed. Addison-Wesley Professional, 2005.

Bower JL, Not all M&Ss are alike - and that matters, Harvard Business Review, Vol. 80, No. 2, 2002, pp.93-101.

Bracewell RH, Wallace KM, Moss M, and Knott D, Capturing design rationale, Computer-Aided Design, Vol. 41, No. 3, 2009, pp.173-186

Bresnahan TF and Greenstein S, Technological Competition and the Structure of the Computer Industry, Journal of Industrial Economics, Vol. XLVII, No. 1, 1999, pp.1-40.

Brown D and Chandrasekaran B, Knowledge and control for a mechanical design expert system, IEEE Computer, Vol. 19, No. 7, 1986, pp.92–100.

Bryant CR, Stone RB, McAdams DA, Kurtoglu T, and Campbell MI, Concept Generation from the Functional Basis of Design, International Conference on Engineering Design (ICED '05), Melbourne, August 15-18, 2005.

Campbell MI, Cagan J, and Kotovsky K, A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment, Research in Engineering Design, Vol. 11, 1999, pp.172–192

Chandler AD, Inventing the Electronic Century, New York: Free Press, 2001.

Chandrasekaran B, Representing function: relating functional representation and functional modeling research streams. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 19, 2005, pp.65–74.

Charniak E and McDermott D, Introduction to artificial intelligence, Addison-Wesley, Reading, Mass, 1985.

Chen P, English sentence and entity-relationship diagrams, Information Sciences, Vol. 29, Issues. 2–3, 1983, pp.127–149.

Chen P, Entity-relationship modeling: historical events, future trends, and lessons learned. In: Software Pioneers: Contributions to Software Engineering, Springer-Verlag, New York, NY, USA, 2002, pp. 296–310.

Chiu I and Shu LH, Using language as related stimuli for concept generation, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 21, 2007, pp.103-121.

Collis DJ, Bane PW, and Bradley SP, Winners and Losers: Industry Structure in the Converging World of Telecommunications, Computing, and Entertainment, In: Yoffie, D.B. (ed.), Competing in the Age of Digital Convergence, Boston, 1997, pp.159-199.

Colombo G, Mosca A, and Sartori F, Towards the design of intelligent CAD systems: An ontological approach, Advanced Engineering Informatics, Vol. 21, 2007, pp.153-168

de Marneffe M-C, MacCartney B, and Manning CD, Generating Typed Dependency Parses from Phrase Structure Parses, In Proceedings of 5th International Conference on Language Resources and Evaluation (LREC2006), Genoa, Italy; 2006.

Dennis A, Wixom BH, and Tegarden D, Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach, 2nd ed. John Wiley and Sons, 2009.

Dong A and Agogino AM, Text analysis for constructing design representations, Artificial Intelligence in Engineering, Vol. 11, No. 2, 1997, pp.65-75 (AID 96 Prize Paper).

Doran W, Stokes N, Carthy J, and Dunnion J, Assessing the Impact of Lexical Chain Scoring Methods and Sentence Extraction Schemes on Summarization, Conference on Intelligent Text Processing and Computational Linguistics (CICLING), Lecture Notes in Computer Science, Vol. 2945, 2004, pp.627-635.

Dosi G, Sources, Procedures and Microeconomic Effects of Innovation, Journal of Economic Literature, Vol. 36, 1988, pp.1126-1171.

Economist, The Great Convergence Gamble, Vol. 357, Issue 8200, Dec 9, 2000, pp.67-68.

Elbendak M, Vickers P, and Rossiter N, Parsed use case descriptions as a basis for object-oriented class model generation, The Journal of Systems and Software, Vol. 84, 2011, pp.1209-1223.

Erden MS, Komoto H, Van Beek TJ, D'Amelio V, Echavarria E, and Tomiyama T, A review of function modeling: Approaches and applications, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol.22, 2008, pp.147–169.

Fellbaum C, WordNet: An Electronic Lexical Database, Cambridge, MA: MIT Press, 1998.

Ficarola F, A Java library for the GEXF file format, 2012, available at https://github.com/francesco-ficarola/gexf4j (as of Jul 11, 2012).

Gambardella A and Torrisi S, Does technological convergence simply convergence in markets? Evidence from the electronics industry, Research Policy, Vol. 27, 1998, pp.445-463.

Garbacza P, Borgob S, Carrarac M, and Vermaas PE, Two ontology-driven formalisations of functions and their comparison, Journal of Engineering Design, Vol. 22, Issues. 11-12, 2011, pp.733-764.

Gartner Press Releases, Gartner Says Android to Command Nearly Half of Wolrdwide Smartphone Operating System Market by Year-End 2012, Egham, UK, Apr 2011, available at http://www.gartner.com/it/page.jsp?id=1622614 (as of Jul 11, 2012).

Gero JS, Design prototypes: a knowledge representation schema for design, AI Magazine, Vol. 11, No. 4, 1990, pp.26–36.

Gero JS and Kannengiesser U, A Function-Behaviour-Structure ontology of processes, AIEDAM, Vol. 21, No. 4, 2007, pp.379-391.

Gero JS and Maher ML, Mutation and analogy to support creativity in computer-aided design, in G. N. Schmitt (ed.), CAAD Futures '91, ETH, Zurich, 1991, pp. 241-249.

Geroski PA, Thinking creatively about markets, International Journal of Industrial Organization, Vol. 16, 1998, pp.677-695.

Giganto R, Generating class models through controlled requirements. In: NZCSRSC-08, New Zealand Computer Science Research Student Conference Christchurch, New Zealand, 2008.

Goel AK, Rugaber S, and Vattam S, Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 23, 2009, pp.23-35.

Gomes-Casseres B, The Alliance Revolution, Cambridge, MA: Harvard University Press, 1996.

Gomez F, Segami C, Delaune C, A system for the semiautomatic generation of E-R models from natural language specifications. Data & Knowledge Engineering, Vol. 29, No. 1, 1999, pp.57–81.

Greenstein S and Khanna T, What dows Industry Convergence Mean?, Competing in the Age of Digital Convergence, David B. Yoffie (ed.), Boston: Harvard Business School Press, 1997, pp.201-226.

Gupta S and Okudan GE, Computer-aided generation of modularised conceptual designs with assembly and variety considerations, Journal of Engineering Design, Vol.19, No.6, Dec 2008 , pp.533-551.

Hague MJ, Taleb-Bendiab A, and Brandish MJ, An adaptive machine learning system for computer supported conceptual engineering design, AI System Support for Conceptual Design. In: Sharpe J, editor. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, London, UK: Springer, 1996.

Hamel G and Prahalad CK, Competing for the Future, Boston: Harvard Business Press, 1994.

Harmain HM and Gaizauskas R., CM-Builder: a natural language-based CASE tool for object-oriented analysis, Automated Software Engineering, Vol. 10, No. 2, 2003, pp.157–181.

Hartmann S and Link S, English sentence structures and EER modeling. In: APCCM ‘07: Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling, Darlinghurst. Australian Computer Society, Inc, 2007, pp. 27–35.

Helfat CE and Lieberman MB, The Birth of Capabilities: Market Entry and the Importance of Pre-history, Industrial and Corporate Change, Vol. 11, 2002, pp.725-760.

Helpman E, General Purpose Technologies and Economic Growth, Cambridge, MA: MIT Press, 1999.

Huang Y, Investigating the cognitive behavior of generating idea sketches through neural network systems, Design Studies, Vol. 29, No. 1, 2008, pp.70-92.

Iansiti M and West J, Technology Integration: Turning Great Research into Great Products, Harvard Business Reivew, Vol. 75, No. 3, 1997, pp.69-79.

Iansiti M, Technology Integration, Boston: Harvard Business School Press, 1997.

Jin Y and Li W, Design Concept Generation: A Hierarchical Coevolutionary Approach, Transactions of the ASME, Vol. 129, Oct 2007, pp.1012-1022.

Katz ML, Remarks on the Economic Implications of Convergence, Industrial and Corporate Change, Vol. 3, 1996, pp.1079-1095.

Kim K-Y, Chin S, Kwon O, and Ellis RD, Ontology-based Integration of Morphological Information of Assembly Joints for Network-based Collaborative Assembly Design, Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM), Vol. 23, No. 1, 2009, pp. 71-88.

Kim K-Y, Manley DG, and Yang HJ, Ontology-based Assembly Design and Information Sharing for Collaborative Product Development, Computer-Aided Design (CAD), Vol. 38, 2006, pp. 1233-1250.

Kitamura Y, Kashiwase M, Fuse M, and Mizoguchi R, Deployment of ontological framework of functional design knowledge, Advanced Engineering Informatics, Vol. 18, No. 2, 2004, pp.115–127.

Klein M, Capturing geometry rationale for collaborative design, In Proceedings of the IEEE Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises (WET ICE'97), 1997.

Klein D and Manning CD, Accurate Unlexicalized Parsing, Proceedings of the 41st Meeting of the Association for Computational Linguistics, 2003, pp. 423-430.

Klein D and Manning CD, Fast Exact Inference with a Factored Model for Natural Language Parsing, In Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge, MA: MIT Press, 2003, pp. 3-10.

Klepper S and Simons KL, Dominance by Birthright: Entry of Prior Radio Producers and Competitive Ramifications in the US Television Receiver Industry, Strategic Management Journal, Vol. 21, 2000, pp.997-1016.

Kodama F, Japanese innovation in mechatronics technology, Science and Public Policy, Vol. 13, No. 1, 1986, pp.44-51.

Kodama F, Technology Fusion and The New R&D, Harvard Business Review, Jul-Aug 1992, pp.70-78.

Lancaster KJ, A New Approach to Consumer Theory, Journal of Political Economy, Vol. 14, 1966, pp.133-156.

Leon N, The future of computer-aided innovation, Computers in Industry, Vol.60, 2009, pp.539–550.

Leon N and Bumas M, Pattern of evolution and industrial design: the pen, In: Proceedings of TRIZCON2004, Abril, 2004, pp. 1–10.

Lewis DD, Natural language processing for information retrieval, Communications of the ACM, Vol. 39, No. 1, 1996, pp. 92-101.

Liang JS and Wei PW, Conceptual design system in a Web-based virtual interactive environment for product development, International Journal of Advanced Manufacturing Technology, Vol. 30, No. 11-12, 2006.

Malak Jr. RJ, Aughenbaugh JM, and Paredis CJJ, Multi-attribute utility analysis in set-based conceptual design, Computer-Aided Design, Vol. 41, No. 3, 2009, pp.214-227.

McAdams DA, Stone RB, and Wood KL, Functional Interdependence and Product Similarity Based on Customer Needs, Research in Engineering Design Vol. 11, 1999, pp.1-19.

McNeil T, Gero JS, and Warren J, Understanding conceptual electronic design using protocol analysis, Research in Engineering Design, Vol. 10, 1998, pp.129-140.

Merriam-Webster, Webster's Third New International Dictionary, Springfield: Merriam-Webster, 2000.

Meziane F, From English to Formal Specifications, PhD thesis, University of Salford, UK, 1994.

Meziane F and Vadera S, Obtaining E-R diagrams semi-automatically from natural language specifications. In: Sixth International Conference on Enterprise Information Systems (ICEIS 2004) , Universidade Portucalense, Porto, Portugal, 14–17 April, 2004, pp. 638–642.

Mich L, NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA, Natural Language Engineering, Vol. 2, No. 2, 1996, pp.161–187.

Mich L and Garigliano R, Nl-oops: a requirements analysis tool based on natural language processing. In: Proceedings of Third International Conference on Data Mining Methods and Databases for Engineering, Southampton, UK, 2002. WIT Press, pp. 321–330.

Miller GA, WordNet: A Lexical Database for English, Communications of the ACM, Vol. 38, No. 11, 1995, pp.39-41.

Mowery DC, The International Computer Software Industry, New York: Oxford University Press, 1996.

Nanda J, Simpson TW, Kumara SRT, and Shooter SB, A Methodology for Product Family Ontology Development Using Formal Concept Analysis and Web Ontology Language, Journal of Computing and Information Science in Engineering, Vol. 6, No. 2, 2006, pp.103-113.

Ng TTH and Leng GSB, Application of genetic algorithms to conceptual design of a micro-air vehicle, Engineering Applications of Artificial Intelligence, Vol. 15, 2002, pp.439-445.

No HJ and Park Y, Trajectory Patterns of Technology Fusion: Trend Analysis and Taxonomical Grouping in Nano-Biotechnology, Technological Forecasting & Social Change, Vol. 77, 2010, pp. 63-75.

Pahl G and Beitz W, Engineering Design: A Systematic Approach. Berlin: Springer–Verlag, 1988.

Partridge D and Rowe J, Computers and Creativity, Intellect: Oxford, 1994.

Passos A and Wainer J, Wordnet-based metrics do not seem to help document clustering, Paper presented at the II International Workshop on Web and Text Intelligence (WTI - 2009), São Carlos, Brazil, Sep 7-11, 2009.

Pavitt K, Patterns of Technical Change: Towards a Taxonomy and a Theory, Research Policy, Vol. 13, 1984, pp.343-373.

Petroski H, The Evolution of Useful Things, Knopf: New York, 1992.

Porter ME, Competitive Advantage, New York: Free Press, 1985.

Qin SF, Wright DK, and Jordanov IN, From on-line sketching to 2D and 3D geometry: a system based on fuzzy knowledge, Computer-Aided Design, Vol. 32, No. 14, Dec 2000, pp.851-866.

Rao PM, Convergence and Unbundling of Corporate R&D in Telecommunications: is Software taking the Helm?, Telecommunications Policy, Vol. 23, 1999, pp.83-93.

Renner G and Ekárt A, Genetic algorithms in computer-aided design, Computer-Aided Design, Vol. 35, No. 8, 2003, pp 709-726.

Rodgers PA, Huxor AP, and Caldwell NHM, Design support using distributed Web-based AI tools, Research in Engineering Design, Vol. 11, 1999, pp. 31–44.

Rogers EM, Diffusion of Innovations, The Free Press: New York, 1995.

Rosenberg N, Technological Change in the Machine Tool Industry, 1840-1910, The Journal of Economic History, Vol. 23, 1963, pp.414-446.

Roy U, Bharadwaj B, Kodkani SS, and Cargian M, Product development in a collaborative design environment, Concurrent Engineering: Research and Applications, Vol. 5, No. 4, 1997, pp. 347–365.

Sahal D, Technological Guideposts and Innovation Avenues, Research Policy, Vol. 14, 1985, pp.61-82.

Salah F and Abdalla H, Creative design tools (CDT): stimulating creative design thinking, Int. J. Design Engg, Vol. 1, No. 2, 2008, pp.125-148.

Saridakis K, Dimopoulos S, Karacapilidis N, and Dentsoras A, Web-Co2Des: a web-based system for collaborative conceptual design, in: 6th International Conference on Computer-Aided Industrial Design and Conceptual Design, Delft, Netherlands, 2005.

Saviotti P, Technological Evolution, Variety and the Economy, Cheltenham, UK and Brookfield, US: Edward Elgar, 1996.

Saviotti P and Metcalfe S, A Theoretical Approach to the Construction of Technological Output Indicators, Research Policy, Vol. 13, 1984, pp.141-151.

Simonton DK, Creativity in Science: Chance, Logic, Genius, and Zeitgeist, Cambridge University Press: Cambridge, 2004.

Sosa R and Gero J, Computational explorations of compatibility and innovation, trends in computer aided innovation, 2nd IFIP Working Conference on Computer Aided Innovation (IFIP International Federation for Information Processing) Springer, ISBN 0387754555, October 8–9 2007, Michigan USA, pp. 13–22.

Spow E, Chrysler's concurrent engineering challenge, Manufacturing Engineering, Vol. 108, No. 4, 1992, pp.35-42.

Stieglitz N, Digital Dynamics and Types of Industry Convergence: The Evolution of the Handheld Computers Market, The Industrial Dynamics of the New Digital Economy, J.F. Christensen and P. Maskell (eds.), Cheltenham, UK and Northampton, MA, USA: Edward Elgar, 2003, pp.179-208.

Stone RB and Wood KL, Development of a Functional Basis for Design, Journal of Mechanical Design, Vol. 122, 2000, pp.359-370.

Sturges RH, O'Shaughnessy K, and Reed RG, A systematic approach to conceptual design, Concurrent Engineering: Research and Applications, Vol. 1, 1993, pp.93-105.

Suarez FF and Utterback JM, Dominant Designs and the Survival of Firms, Strategic Management Journal, Vol. 16, 1995, pp.415-431.

Takala T, Design transactions and retrospective planning tools for conceptual design. In: Akman V, ten Hagen PJW, Veerkamp PJ, editors, Intelligent CAD systems II, Springer Verlag, 1989, pp.262-272.

Teece DJ, Dosi G, Rumelt R, and Winter S, Understanding Corporate Coherence: Theory and Evidence, Journal of Economic Behavior and Organization, Vol. 23, 1994, pp.1-30.

Tjoa MA and Berger L, Transformation of requirement specifications expressed in natural language into an EER model, In: Entity-Relationship Approach – ER '93: 12th International Conference on the Entity-Relationship Approach Arlington, Texas, USA, December 15–17. Springer, Berlin/Heidelberg, 1994, pp. 206–217.

Tseng KC, Abdalla H, and Shehab EM, A Web-based integrated design system: its applications on conceptual design stage, International Journal of Advanced Manufacturing Technology, Vol. 35, No. 9-10, 2008.

Tushman M and Anderson P, Technological Discontinuities and Organizational Environments, Administrative Science Quarterly, Vol. 31, 1986, pp.439-465.

Umeda Y and Tomiyama T, FBS modeling: modeling scheme of function for conceptual design. Proc. Working Papers of the 9th Int. Workshop on Qualitative Reasoning About Physical Systems, Amsterdam, 1995, pp. 271-278.

Umeda Y and Tomiyama T, Functional reasoning in design, IEEE Expert: Intelligent Systems and Their Applications, Vol. 12, No. 2, 1997, pp.42–48.

Ur-Rahman N, Textual Data Mining Applications for Industrial Knowledge Management Solutions, PhD Thesis, Loughborough University, June 2010.

Wahono RS and Far BH, OOExpert: distributed expert system for automatic object-oriented software design. In: Proceedings of the 13th Annual Conference of Japanese Society for Artificial Intelligence, 1999, pp. 456–457.

Wang L, Shen W, Xie H, Neelamkavil J, and Pardasani A, Collaborative conceptual design – state of the art and future trends, Computer-Aided Design, Vol. 34, 2002, pp. 981-996.

Wang Q, Rao M, and Zhou J, Intelligent Systems for Conceptual Design of Mechanical Products, In: Mital A, Anand S, editors. Handbook of Expert Systems Applications in Manufacturing: Structures and Rules, New York: Chapman & Hall, 1994.

Warschauer M and Healey D, Computers and language learning: An overview, Language Teaching, Vol. 31, 1998, pp.57-71.

Welch RV and Dixon JR, Representing function, behavior and structure during conceptual design, In Design Theory and Methodology—DTM'92 (Taylor, D.L., & Stauffer, L.A., Eds.), New York: ASME, 1992, pp. 11–18.

Wirtz B, Reconfiguration of Value Chains in Converging Media and Communications Markets, Long Range Planning, Vol. 34, 2001, pp.489-507.

Yang MC, Observations on concept generation and sketching in engineering design, Research in Engineering Design, Vol. 20, No. 1, 2009, pp.1-11.

Yang MC, Wood III WH, and Cutkosky MR, Design information retrieval: a thesauri-based approach for reuse of informal design information, Engineering with Computers, Vol. 21, No. 2, 2005, pp.177-192.

Yoffie DB, Introduction: CHESS and Competing in the Age of Digital Convergence, Competing in the Age of Digital Convergence, David B. Yoffie (ed.), Boston: Harvard Business School Press, 1997, pp.1-35.

Yoshioka M, Umeda Y, Takeda H, Shimomura Y, Nomaguchi Y, and Tomiyama T, Physical concept ontology for the knowledge intensive engineering framework, Advanced Engineering Informatics, Vol. 18, No. 2, 2004, pp.69–127.

Zdrahal Z and Domingue J, The World Wide Design Lab: An Environment for Distributed Collaborative Design, International Conference on Engineering Design (ICED ’97), Tampere, August 19-21, 1997.

Zha XF and Sriram RD, Platform-Based Product Design and Development: A Knowledge Intensive Support Approach, Knowledge-Based Systems, Vol 19, No 7, 2006, pp.524-545.

ABSTRACT

DEVELOPMENT OF A COMPUTER-AIDED DESIGN SUPPORTING SYSTEM FOR TRANSFORMATIVE PRODUCT DESIGN

by

KEUNHO CHOI

August 2012

Advisor: Dr. Kyoung-Yun Kim

Major: Industrial Engineering

Degree: Doctor of Philosophy

Modern product innovation is often delivered by fusing technologies from different domains. To meet the current speedy pressure of product innovation, the conceptual design is a very critical stage to develop an innovative product. This research presents a new design paradigm, Transformative Product Design (TPD), to meet the demands in the conceptual design. TPD aims to design a new product from a combination out of a base product and reference products, which have been developed.

To expedite the TPD paradigm, *Interaction Network* is introduced to represent a product design by following a representative product design representation method in the teleological perspective. However, generating an interaction network of a product is very cumbersome and somewhat subjective. In the past decade, a few research works have been reported to identify functions of a product in a systematic way. They are not satisfactory to develop an interaction network for TPD systematically. To systematically generate an interaction network, this research adopts functional requirements in design documents as the source of the proposed method. The proposed method in this research aims to identify functions, structures and dynamic behaviors

between structures from the functional descriptions in natural (English) language by adapting natural language-based object identification methods in the software engineering.

Another aim of this research is to provide semantic capability to utilize the use of a natural language throughout the design transformation process, taking into account cross-disciplinary design knowledge that domain experts could struggle if the knowledge is beyond their expertise. To fulfill the demand, this research builds research methods for the design transformation, based on the proposed similarity score, i.e., *Stem-POS based Similarity Score*. By employing the proposed score, this research detects concept functions and generates transformative design alternatives with interaction networks. Additionally, for the design transformation, the *degree of transformability* is to check how much products are semantically related each other, by adapting the semantic similarity score.

Finally the proposed methods have been implemented in a *Computer-Aided Transformative Design (CATPD)* supporting system with minor human involvement to a minimum. The proposed methods and the system are validated according to Stieglitz's convergence types.

AUTOBIOGRAPHICAL STATEMENT

I was born in Seoul, Republic of Korea, in 1979. I received my bachelor's degrees from the School of Management and Economics at Handong Global University in 2004 and my master degree from the Department of International Management at Kyung Hee University in 2006. During my master’s work, I focused on a context-aware based recommendation system development by applying the RFID technology to detect the context information of users and their preferences through web services. During my doctoral study in the Department of Industrial and Systems Engineering at Wayne State University, I have gained a broad range of interests spanning from computational intelligence, semantics, and natural language processing, for the design engineering, proposed a new design paradigm, called Transformative Product Design, and developed a computer-aided conceptual design supporting system for the design paradigm. Throughout these diverse experiences, I have pursued a consistent goal of applying data science and computational intelligence to support innovative product designs. To reach this goal, I have focused on conceptual design as the fundamental aspect of innovative product designs. My approach to research focuses on the application of conceptual product design and computer-aided design supporting systems.