

1-1-2010

Causal Product Knowledge Management

Yun Seon Kim
Wayne State University,

Follow this and additional works at: http://digitalcommons.wayne.edu/oa_dissertations

Recommended Citation

Kim, Yun Seon, "Causal Product Knowledge Management" (2010). *Wayne State University Dissertations*. Paper 135.

This Open Access Dissertation is brought to you for free and open access by DigitalCommons@WayneState. It has been accepted for inclusion in Wayne State University Dissertations by an authorized administrator of DigitalCommons@WayneState.

CAUSAL PRODUCT KNOWLEDGE MANAGEMENT

by

YUN SEON KIM

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

2010

MAJOR: INDUSTRIAL ENGINEERING

Advisor

Date

© COPYRIGHT BY

YUN SEON KIM

2010

All Rights Reserved

ACKNOWLEDGEMENTS

In acknowledging my appreciation and thanks, I am most grateful to the Almighty God for His divine guidance and blessings. Specially, He designs my life to complete this work and has brought many wonderful people during my research.

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

My special thanks go to all of the Computational Intelligence and Design Informatics Laboratory members for their support and suggestions in this research. I acknowledge the help of Keunho Choi and Jihoon Kim in providing useful suggestions and assistance.

Finally, I wish to thank to my parents who have always provided endless love throughout my life with their continuous prayers. My deepest appreciation has been reserved for my wife YooSoo for all her love, unwavering support, and continued prayers. I would like to thank my son David. You have always made me smile, David. Thank you.

TABLE OF CONTENTS

Acknowledgement	ii
Chapter 1 Introduction and Objectives	1
1.1 Introduction	1
1.2 Research Objectives	11
1.3 Research Organization	14
Chapter 2 Expected Significance	15
Chapter 3 Literature Reviews	17
3.1 Trends in Product Development	18
3.2 Knowledge in Product Development	20
3.3 Issues in Product Development Knowledge	23
3.3.1 Design Knowledge Reuse Issues	23
3.3.2 Product Development Knowledge Management Issues	24
3.4 Procedural Knowledge and Causal Knowledge	27
3.4.1 Procedural Knowledge	27
3.4.2 Causal Knowledge	30
3.4.2.1 Causality	30
3.4.2.2. Bayesian Belief Network	31
3.5 Ontology and Semantic Web	39
3.6 Analysis in Product Development	43
Chapter 4 Preliminary Study	47
4.1 Systematic Knowledge Elicitation and FCM-BBN Constructor	47
4.1.1 Comparison of FCM and BBN	48
4.1.2 Systematic Construction of FCM	51
4.1.2.1 Eliciting Variables Relevant to the Problem	51
4.2 Eliciting Causal Structure between Variables	52
4.2.1 Defining the Augmented Matrix	54

4.2.2 Constructing the Additive Matrix	56
4.2.3 Converting Bipolar Values into Linguistic Weights	57
4.3 Systematic Generation of BBN from FCM	58
4.3.1 Constructing BBN form FCM	59
4.3.1.1 Building BBN Compatible Causal Structure form CM	60
4.3.1.2 Constructing CPTs of BBN from Causal Weights of FCM	62
4.4 Case study: Fault Diagnosis for Fuel Nozzle	65
4.5 Conclusion	69
Chapter 5 Product Design Knowledge Representation and Transformation ...	71
5.1 Definitions and Relationships in Knowledge	72
5.2 Mathematical Comparison of PK and CK	77
5.3 Implementation: Knowledge Modeling with SysML	89
5.4 Demonstration: Representation and Reasoning Capability of Causal Knowledge	96
5.5 Conclusion	101
Chapter 6 Degree of Causal Representation	103
6.1 Causal Design Knowledge Evaluation and Support	103
6.2 Mathematical Representation of Causal Knowledge	106
6.3 Evaluation of Causal Knowledge Network with Weighted Vertices	109
6.4 Evaluation of Causal Knowledge Network with Weighted Edges	112
6.5 Validation of DCR-based Causal Knowledge Evaluation	118
6.5.1 Comparison of Different Causal Knowledge	118
6.5.2 Case Study	123
6.5.3 Implementation: Knowledge Network Optimality Evaluation System ...	127
6.5 Conclusion	132
Chapter 7 DCR Index and Knowledge Integration	134
7.1 DCR Index	134
7.2 Knowledge Integration	139

7.2.1 Inter-actor Knowledge	141
7.2.2 Inter-process Knowledge	143
7.2.3 Inter-product Knowledge	146
7.2.4 Integration of Causal Knowledge	148
7.2.4.1 Ontological Knowledge Mapping	150
7.2.5 Utilization of Causal Knowledge Integration	151
7.3 Conclusion	163
Chapter 8 Implementation	165
8.1 Causal Design Knowledge Management System	165
8.2 Case Study	176
Chapter 9 Conclusion	180
Appendix A	185
Appendix B	221
Appendix C	225
Appendix D	227
References	234
Abstract	253
Autobiographical Statement	254

LIST OF FIGURES

Figure 1-1: Knowledge accumulation in product development processes	5
Figure 3-1: A Bayesian Belief Network representing Causal influence among five variables	34
Figure 4-1: Augmented matrix for previous two matrices	55
Figure 4-2: Additive matrix after combining causal knowledge of two experts	57
Figure 4-3: Causal Structure	58
Figure 4-4: BBN Compatible Causal Structure	61
Figure 4-5: Network from FCM-BBN	66
Figure 4-6: Network from BBN	67
Figure 5-1: Knowledge transformation	77
Figure 5-2: Procedural knowledge representation for two-object-welding knowledge ..	78
Figure 5-3: Causal knowledge representation for joining knowledge	80
Figure 5-4: The Four Pillars of SysML	90
Figure 5-5: Causal network for the fuel nozzle knowledge	96
Figure 5-6: Examples of decision alternative using causal knowledge diagnosis	97
(a) Decision alternative with one observation	97
(b) Decision alternative with four observations	98
(c) Decision alternative with five observations	98
Figure 5-7: The examples of the effects of the causal knowledge prediction	100
(a) The effects of material modification	100
(b) The effects of diameter modification	100

Figure 6-1: Framework of the causal design knowledge evaluation and support system	106
Figure 6-2: Examples of causal knowledge networks	108
Figure 6-3: Examples of knowledge network with weighted vertices	112
Figure 6-4: Examples of knowledge network with weighted edges	115
Figure 6-5: Processes of the causal knowledge evaluation	118
Figure 6-6: Examples of causal knowledge network with weighted edges for comparison	119
Figure 6-7: Comparison results for causal knowledge network with weighted edges	120
(a) Probability comparison	120
(b) Network comparison	120
Figure 6-8: Examples of causal knowledge network with weighted vertices for comparison	122
Figure 6-9: Comparison results for causal knowledge network with weighted vertices	122
(a) Probability comparison	122
(b) Network comparison	123
Figure 6-10: Examples of causal knowledge	124
(a) Causal knowledge for assembly design	124
(b) Causal knowledge for wheel design and maintenance	125
(c) Causal knowledge for fuel nozzle design and maintenance	125
Figure 6-11: Example of UGS NX5 valve design	128
Figure 6-12: Web-based causal design knowledge evaluation and support system	128
Figure 6-13: Example of the effects of the design modification	129
Figure 6-14: Example of the design factors from maintenance issues	129

Figure 7-1: Examples of network for DCR	135
Figure 7-2: DCR indexing process	137
Figure 7-3: Example of DCR index test	138
Figure 7-4: Current product development knowledge acquisition and loss	140
Figure 7-5: Knowledge relationship for product development	141
Figure 7-6: Inter-actor knowledge integration for product development	142
Figure 7-7: Inter-actor knowledge framework for product development	143
Figure 7-8: Inter-process knowledge framework vs. current knowledge framework for product development	144
Figure 7-9: Missing knowledge in current product development knowledge framework	145
Figure 7-10: Heterogeneous product development knowledge framework	147
Figure 7-11: Inter-product knowledge framework with unsupervised learning	147
Figure 7-12: Knowledge integration cases	149
Figure 7-13: Ontological BBN design knowledge	150
Figure 7-14: Snapshot of causal product design knowledge management system for knowledge integration	152
Figure 7-15: Result of knowledge integration with DCR evaluation	153
Figure 7-16: Knowledge network for integration	154
(a) Knowledge network for Bad weld based	154
(b) Knowledge network for Voids based	154
(c) Knowledge network for newNetwork-wheel	154
Figure 7-17: Effects of the design modification	156

(a) Bad Weld based wheel knowledge	156
(b) Voids based wheel knowledge	156
Figure 7-18: Effects of design modification in integrated wheel knowledge	157
Figure 7-19: Knowledge network for integration with different domains	158
(a) Bad weld based wheel knowledge	158
(b) Tire knowledge	159
(c) Integrated wheel tire knowledge	159
Figure 7-20: Effects of the design modification	160
(a) Bad weld based wheel knowledge	160
(b) Tire knowledge	161
Figure 7-21: Effects of design modification in integrated wheel-tire knowledge	162
(a) For bad weld based wheel knowledge	162
(b) For tire knowledge	163
Figure 8-1: .net based developing environment	166
Figure 8-2: Basic concept: design support processes	167
Figure 8-3: System architecture of the causal product design knowledge management system	168
Figure 8-4: Systematic knowledge acquisition	169
(a) Elicitation of important variables	169
(b) Causal relationship between variables	169
(c) Systematical generated Fuzzy Cognitive Map	170
(d) Bayesian Belief Network from Fuzzy Cognitive Map	170
Figure 8-5: Result of evaluation (DCR)	171

Figure 8-6: Causal knowledge integration example	173
Figure 8-7: Integrated knowledge (newNetwork.xdsl)	173
Figure 8-8: integrated knowledge evaluation	174
Figure 8-9: knowledge network interface engine between knowledge inference engine and causal product design knowledge management system	175
Figure 8-10: Example of assemble design	177
Figure 8-11: Example of wheel for automotive	177
Figure 8-12: Example of fuel nozzle for aircraft engine	178
Figure 9-1: Contribution of the research in knowledge management system requirement	184

LIST OF TABLES

Table 3-1: Knowledge Perspectives and Their Implications	21
Table 4-1: A Summary of Comparison Results	50
Table 4-2: Adjacency Matrix for FCM	59
Table 4-3: BBN Compatible Adjacency Matrix	59
Table 4-4: Three-step Construction of CPT for Assembly Cost (a,b,c)	63
Table 4-5: The Test Results with Fuel Nozzle	69
Table 5-1: Comparison result between procedural knowledge and causal knowledge	88
Table 5-2: SysML implementation for PK and CK	92
Table 6-1: The results of causal knowledge network analysis	126
Table 7-1: Result of DCR index with vertices 3 to 11	137
Table 7-2: Result of DCR index test	139
Table 8-1: Evaluation index (DCR index)	172
Table 8-2: Causal design knowledge repository	175
Table 8-3: Results of case study (DCR)	179

CHAPTER 1

INTRODUCTION AND OBJECTIVES

1.1 Introduction

Today, manufacturing enterprises are globalized with the world-wide availability of technology, capital, information, and labor. True competitive advantage can only result from the ability to bring highly customized quality products to the market at lower cost and in less time. Product development has become a very complicated process. Discrete product manufacturers are under pressure from customers and the market to move away from the traditional make to stock production model to a build to demand model. Many customers are no longer satisfied with mass produced goods. They are demanding customization and rapid delivery of innovative products [FIPER 2001, ISIGHT 2002]. Faster change in market demand drives faster obsolescence of established products. Industries now realize that the best way to reduce life cycle costs is to evolve a more effective product development paradigm using the Internet and web-based technologies. Yet, there remains a gap between these current market demands and current product development paradigms.

In additionally, global marketing competition makes manufacturers more conscious of quality, cost, and time-to-market. This global economical and technological environment poses a challenge of how to realize a true collaborative environment. In the collaborative environment, engineers can cooperate globally during the overall product development processes. However, one survey found that

74% of respondents believed that their organization's best knowledge was not accessible and reusable, and 68% thought that mistakes were reproduced several times [Gazeau 1998]; more than 75% of product design activities have been conducted due to the lack of product development knowledge reuse and it has been long recognized as a critical problem in modern product development [DeLong 2004]; this problem is still recently indicated around industries according to professional meetings and interviews that we conducted [PDSEC 2007]. Busby [Busby 1999] notes three issues: that design reuse was desirable but not practiced, that inevitable additional efforts to reuse the design are required, and that knowledge loss, inappropriate replication, and errors are all-too-common issues encountered when attempting to reapply existing but incomplete knowledge to a new design. Furthermore, problems in various product life-cycle activities may arise since expertise is often unavailable or the knowledge has been forgotten. This situation contributes to long delays in recognizing potential failures in product development [Dieter 2001]. When the potential failure is not promptly identified in the early stages of the product development process, it causes greatly increased downstream costs, such as warranty and maintenance. Because traditionally product development knowledge remains un-codified, mapping the internal expertise is a potential research challenge in knowledge management [Ruggles 1998, Arkell 2007].

Such perception of the failure to apply existing knowledge is an incentive for developing a knowledge-driven decision support system. Specifically, there is

strong need for a framework, which aims to understand product development knowledge and to develop fast and efficient information/knowledge database for the better product development. Recently, the Information Technology (IT) has evolved rapidly and has made enormous impact on the whole spectrum of industries. Various IT applications and CAx (Computer Aided technology) tools in manufacturing have been considered and employed to overcome the following challenges in the practice of collaborative product development processes: 1) lack of information from suppliers and working partners; 2) incompleteness and inconsistency of product knowledge within the collaborating group; 3) incapability of processing information/data from other parties due to interoperability.

However, it seems currently available tools and techniques are not entirely suitable and effective enough to handle the challenges/pressures faced by product development processes. Previous research on product development knowledge reuse has focused on searches, by matching keyword and file name, or searching by specific indices (e.g., part number, relationship among parts, etc.). However, these methods indicate various drawbacks based on [Iyer 2005]. First, product development knowledge is often incomplete or is not adequately defined at a detailed level for current information search methods. Second, it is often not true that proper initial information (e.g., project name or part name) is known before an actual search. Third, the search space and time requirements are often cumbersome and hence impractical, generating search results that are either too detailed or too broad. The product development knowledge, which is a blood

stream of development cycles, is still not fully captured, maintained, and reused. Problems mostly come from the lack of unified protocol of knowledge acquisition and diffusion. Therefore, developing the product development knowledge reuse framework becomes one of the important issues in product development research.

Knowledge loss because of retirement, downsizing, and turnover is not only one of the costliest problems, but it also one of the most widely ignored problems facing organizations today. Figure 1-1 shows the knowledge accumulation and loss between preserved knowledge and missing knowledge in the product development processes. Preservation of knowledge is a daunting challenge. Different strategies for knowledge preservation have been considered [Bott 2007]. As information is stored increasingly in electronic formats, there is a need to reexamine the principles of preservation under which we have traditionally been trained and under which we are still often guided in decision making. These need to be reevaluated and compared so that "points of convergence or divergence can be evaluated." [Cloonan 1993]. In addition, the need to optimize organization processes rather than individual benefits poses challenges [Rangan 2005]. The importance of a lifecycle-wide knowledge sourcing strategy in support of the Enterprise System investment is articulated [Gable 2005]. Considerable research has been done in knowledge engineering and using new technologies [Matsumoto 2005, Barnard 2003, O'Hara 2002]. Knowledge Management specialists have sometimes failed to recognize the synergy that knowledge engineering methodologies and tools hold to enhance the state of the art in practical domains [Liebowitz 2001].

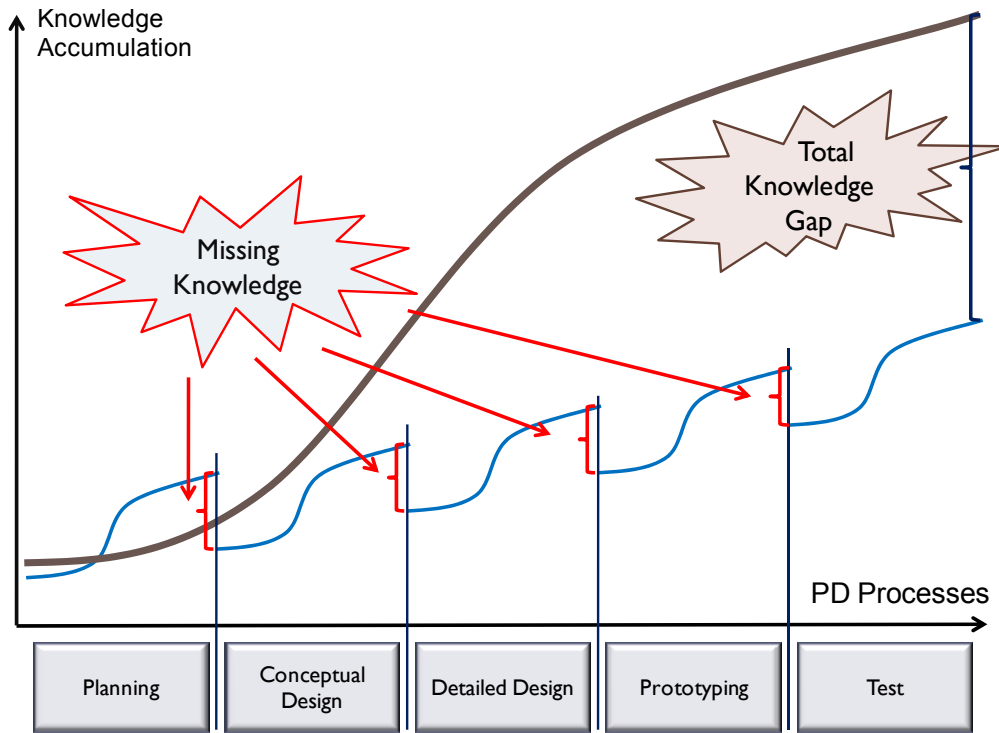


Figure 1-1 Knowledge accumulation in product development processes

Recently, a paradigm called mass collaboration is emerging for harnessing the knowledge and creativity. It is based on the collaboration and competition of large groups of people in innovative ways [Tapscott 2006]. An example of the success of mass collaboration is the free encyclopedia, Wikipedia.com. Currently, about 10 million volunteers collaborate to create an encyclopedia which consists of about 9.5 million articles in 256 languages. Its accuracy is comparable to that of Encyclopedia Britannica [Giles 2005]. Further, the large communities on sites such as Facebook for general networking (58 million users), Flickr for photo sharing (4 million users), LinkedIn for business networking (17 million users), and soundpedia

(3.5 million users) for music sharing show that individuals are increasingly participating in collaborations over the Internet at massive scales. Recently, there have been a few efforts on applying the concepts of open-source development to physical product development [OpenMoko 2008, Open 2008, Baker 2006]. Based on current mass collaboration paradigm, the product development management required web-based knowledge acquisition and reuse to handle discrete product development knowledge from currently working or retired stakeholders.

To acquire product development knowledge, the representation of knowledge is one of core requirements. One view of knowledge is that of a condition of access to information [McQueen 1998]. According to this view, product development knowledge must be organized to facilitate access to and retrieval of content. Product development knowledge shows three different knowledge ways: declarative, procedural, and contextual knowledge. Declarative knowledge (DK) is knowledge of facts or is knowledge that answers the question of “*what*”. Procedural knowledge (PK) addresses the question of “*how*”. Contextual knowledge (CoK) addresses “*when*” and “*why*” to use the declarative knowledge [Roberts 2003, Yoo 2006]. As an illustration, if we suppose an assembly method of certain parts, then this method and the parts themselves from DK. When we start consider how the parts are assembled, the DK of the assembly method becomes PK. If we consider the context of CoK (i.e., how the parts can be assembled under which conditions (when and why) as well as how the resultant outputs would be), then the CoK embed PK. Therefore, PK can represent an assembly, which has objects, method,

conditions, and output.

Most product development knowledge is represented by PK, since the PK includes both declarative and CoK. Product design knowledge can be represented by PK. However, PK is broad and requires unwieldy processes to define itself discretely. Furthermore, during product development processes, PK cannot fully represent product design knowledge [Kim 2008]. Causal knowledge (CK), which utilizes causal reasoning, is particularly useful for overcoming these challenges. By modeling causal relationships, causes of certain events are diagnosed and their effects are predicted [Gopnik 2002, Gopnik 2004, Liu 2001]. The author's previous study [Kim 2008] concludes that CK's characteristics are more beneficial in representing product development knowledge than PK, and that CK provides more functions of the knowledge practices. However, CK is still rarely captured in the product development processes because eliciting this type of knowledge the domain expert is a very time-consuming process. Furthermore, product development requires multi-disciplinary, domain knowledge. Systematic extraction of integrated CK is very difficult [Kim 2008]. However, even with these known difficulties in terms of collecting the knowledge elicitation from domain experts, CK's value outweighs its difficulties and costs.

To fully capture and diffuse the product development knowledge, this dissertation research aims to develop knowledge representation formalism, knowledge evaluation method and evaluation index, knowledge integration method, and web-based product design knowledge management system. The knowledge

acquisition method is conducted by a preliminary study with colleges in South Korea. This preliminary study addresses two core function of the knowledge acquisition, systematic knowledge acquisition from domain experts and systematic knowledge conversion from fuzzy cognitive map to Bayesian belief network.

The first topic is knowledge representation formalism. This topic addresses which knowledge representation formalism can express the product development knowledge in order to utilize existing knowledge. First, this research is starting with the mathematical definitions of the procedural product development knowledge, the causal product development knowledge, and the knowledge transformation by set theory. Based on this mathematical definitions, the comparison of PK and CK with the knowledge perspectives indicated in product development knowledge and discussion of CK's effectiveness in realizing an integrated representation of the product development knowledge are presented. This comparison is presented the mathematical effectiveness of causal and procedural knowledge from four perspectives: knowledge expression, decision alternative representation, reasoning, and knowledge cultivation. After understanding and comparing the relationship between PK and CK, the causal knowledge representation formalism can present product development knowledge. However, most product development knowledge is represented by PK. Therefore, the knowledge transformation method from PK to CK is developed and defined. The features of CK with an actual case study, a fuel nozzle on an aerospace engine, is demonstrated.

To confidently use causal knowledge for the product development knowledge, the evaluation of causal knowledge is required. This research presents a new causal design knowledge evaluation and management system that uses a causal knowledge evaluation method to quickly and easily design a new product and to help prevent potential future failure. One of the new system's core functions is causal knowledge evaluation. The developed causal knowledge evaluation method compares knowledge networks using degree of causal representation (DCR). In this research, causality (C) and network connectivity (NC) are used for the causal knowledge network with weighted vertices, and weighted network connectivity (WNC) for the causal knowledge network with weighted edges. Causality (C) is a measure of how the causal knowledge network represents a causal relationship, taking into consideration the incoming and outgoing edges of each vertex. Network connectivity (NC) represents the connection of the network with the ratio of total connections in the causal knowledge network. WNC is composed with the network connectivity and the normalized edge weights of vertices. This developed method is tested with three real causal knowledge cases.

To use DCR evaluation method, one limitation of this method should be overcome. The limitation of DCR method is that it is strongly dependant with the number of vertices in causal knowledge network because the causal knowledge is represented by network as defined in chapter 5. This limitation restricts the comparison of multiple causal knowledge for selecting better design knowledge in product development. To overcome this limitation, new evaluation index, which is

called DCR index, is developed. Using DCR index, multiple causal knowledge with different number of vertices are compared in chapter 7.1. Also, the validation of DCR index is conducted in Chapter 7.

The evaluated knowledge can be used for determining which knowledge is more appropriate for a new design knowledge. For the new design knowledge, it can be selected from existing knowledge or be generated with existing knowledge. To generate the new knowledge from existing knowledge, knowledge integration method is required. In this research, based on knowledge relationship, the new knowledge integration method is developed. The knowledge relationship classifies product development knowledge into three categories: inter-process, inter-actor, and inter-product knowledge in order to integrate heterogeneous existing product development knowledge. To systematically integrate the product development knowledge associated to these three categories, appropriate knowledge integration methods are required. With these three categories, a new knowledge framework, which is called inter-relational knowledge framework, is developed. First, inter-process knowledge framework acquires and reuses different domains knowledge, which have different constraints for each domain, using causal network structure update method during the product development processes. Second, inter-actor knowledge framework acquires and reuses the same domain knowledge with different actors (e.g., designers, engineers, etc.) using causal network integration. Third, inter-product knowledge framework acquires and reuses different domains knowledge and different products knowledge using causal network and structure

integration between different structures. In this framework, a classification of inter-product knowledge for similar product is required because product knowledge of all kinds of products cannot be integrated to one general causal network. Based on this frameworks, the cases of the causal knowledge integration is developed as shown in chapter 7.2. Finally, the innovative knowledge integration method is validated with real case.

In the summary of the this research, this research develops the new causal product design knowledge management system to acquire, represent, store, integrate, and reuse the existing knowledge for new product design. The web-based system makes a communication among the stakeholders, who are currently working or retired. Also, this system can be a model of web-based collaboration environments with discrete knowledge and stakeholders in product development. During the implementation of the system, two useful outputs are developed, knowledge network interface engine between causal product design knowledge management system and causal knowledge inference engine (GeNIe) and causal design knowledge repository with causal knowledge evaluation results, which include DCR, DCR index, and other considerable factors.

1.2 Research Objectives

Based on much research in several areas, such as product development, knowledge management, information technology, and decision support systems, following needs are required for establish an efficient product development knowledge management system: 1) there is a need for a framework, which aims to

understand and capture recursive product development knowledge, 2) it is needed the extension of causal network to update recursive product development knowledge and to integrate existing product development knowledge to reuse it, 3) collaborative IT tools are needed to improve collaboration among distributed product development groups, enhance knowledge sharing, and assist in better decision making, 4) integration of appropriate information, decision technology, and domain knowledge in decision-making processes of distributed network-based collaboration, is required in a seamless and automated manner. Furthermore, current product development knowledge management framework cannot fully manage recursive product development knowledge in the product development processes because of complexity of product development knowledge, lack of product development knowledge management, and lack of representation for recursive product development knowledge. Therefore, the overall hypothesis of this research is that causal product design knowledge management system can be developed to capture and represent casual product design knowledge, to integrate knowledge in product lifecycle, and to utilize casual product design knowledge for better product development decision making.

The research objectives for causal product design knowledge management are as follows:

1. *Causal Design Knowledge Acquisition and Representation*: Investigate 1) knowledge acquisition method that can be used to guide the process of collecting experiential knowledge and information in a systematic manner from

domain experts, 2) mathematical representation for procedural knowledge, causal knowledge, and knowledge transformation, 3) mathematical comparison between procedural knowledge and causal knowledge to select better knowledge representation formalism, 4) and knowledge transformation from procedural knowledge to causal knowledge.

2. *Causal Design Knowledge Evaluation and Integration*: Investigate 1) evaluation method that can be assess the causal design knowledge network, 2) comparison standard for multiple causal design knowledge to select better knowledge of the product design, 3) and knowledge network integration method that can be accumulate existing causal design knowledge to utilize new product design.

3. *Implementation of Causal Design Knowledge Management System and Case Study*: Develop 1) web-based knowledge network optimality evaluation system that can be assist a designer for providing design analysis information, 2) knowledge network interface engine between causal design knowledge management system and causal knowledge reasoning system, 3) causal design knowledge repository that includes design analysis information and it can be utilized to other application or system separately, 4) and case study for causal design knowledge management system validation.

1.3 Research Organization

In this documentation, Chapter 2 provides a significance of this research. Chapter 3 provides a background and literature review of relevant research areas and important aspects of this research. Chapter 4 addresses the preliminary study for systematic knowledge acquisition and knowledge conversion. Chapter 5 discusses the mathematical definitions of this research (procedural knowledge, causal knowledge, and knowledge transformation) and comparison and transformation between procedural knowledge and causal knowledge. Chapter 6 explains the causal knowledge evaluation method. Chapter 7 mentions the causal knowledge evaluation index for multiple knowledge comparison and knowledge integration for generating a new knowledge from existing knowledge. Chapter 8 shows implementation of the web-based causal product design knowledge management system. Finally, Chapter 9 concludes this dissertation with the contributions and areas of future research.

CHAPTER 2

SIGNIFICANCE OF RESEARCH

The US engineering industry base is facing a significant loss of knowledge due to large numbers of employees retiring in the next decade. Problems in various product developments including product design may arise when the expertise is no longer available or the knowledge is forgotten. Also, most of product design knowledge is not reusable, because product design knowledge in an organization remains un-codified. Previous research on design knowledge reuse has been focused on search by matching keyword and file name or search by specific indexes (e.g., part number, relationship among parts, etc.). However, these methods indicate various drawbacks [Iyer 2005]. First, product design knowledge is often incomplete or is not defined detailed enough for the current information search methods. Second, it is often not true knowing proper initial information (e.g., project name or part name) before an actual search. Third, often the search space and time is cumbersome and it may generate too detailed search results or too broad results. Generally, knowledge-based system can solve or infer these drawbacks. However, knowledge-based systems have been developed solely through the use of rule-based approach, which allows for easy modeling of expert reasoning, but such a approach is not general and for a specific use; thus, existing experience and analyses show that this approach has serious limitations on associations between observable findings and diagnostic hypotheses [Chen 2001]. Furthermore, the product development knowledge cannot be appropriately

acquired, represented, and reused by these techniques. To address these challenges, this research develops new methodologies and tools to capture, represent, store, and reuse domain knowledge from experts and implement a novel web-based causal product design knowledge management system to systematically utilize the knowledge from experts, who are currently working or retired. The particular emphasis is on these research areas: 1) design knowledge acquisition, 2) causal knowledge representation, 3) causal knowledge evaluation and index, 4) causal knowledge integration, 5) and causal design knowledge management system.

This research aims to extend design, technological and computational innovations in knowledge acquisition, knowledge representation, integration of knowledge, web-based knowledge management system to design problem solving processes. Results from this research are expected to advance our understanding of 1) capturing domain knowledge from experts, 2) systematic knowledge acquisition for current working engineering knowledge retention and for keeping retired professionals engaged in industry, 3) capturing and transforming existing procedural engineering knowledge to better knowledge representation formalism, 4) evaluating causal knowledge to make design decision, 5) comparing multiple design knowledge in heterogeneous product, 6) integrating existing design knowledge to generate refined knowledge, 7) and systematic knowledge management using information technologies and tools. Thus, this research leads to discovery and integration across these frontiers.

CHAPTER 3

LITERATURE REVIEWS

3.1 Trends in Product Development

Product manufacturers are under pressure from customers to move away from the traditional make-to-stock production model to a build-to-demand model. True competitive advantage can only result from the ability to bring highly customized quality products to the market at lower cost and in less time. Product development has become a very complicated process. Many customers are no longer satisfied with mass-produced goods. They are demanding customization and rapid delivery of innovative products. Industries now realize that the best way to reduce product life-cycle costs is to evolve a more effective product development paradigm using the IT and web-based technologies [Engineous 2005]. Yet, there remains a gap between these current market demands and current product development paradigms. One possible approach to fill this gap is to seamlessly integrate product development processes into a collaborative environment.

Recently, the scope of design participants has been increased. In particular, persons (e.g., customers) who are not necessarily experienced in product engineering can informally partake in the design process by providing input from an existing or potential product's end operating environment. Furthermore, other organization members, who are not traditionally a member of product development, can have a role in product design and development. Also, designers are no longer

merely exchanging geometric data, but knowledge about design and the product development process, including specifications, design rules, constraints, and rationale. As design becomes increasingly knowledge-intensive and collaborative, the need for computational frameworks to enable engineering product development by effectively supporting the formal representation, acquisition, and reuse of all product development knowledge, becomes more critical [Lutters 1997, Szykman 2001]. However, the cumulative, creative, iterative, evolutionary product development knowledge and rationale behind the product are infrequently captured or retained. Although a few researchers [Lin 1996, Horváth 1998, Kitamura 2004] have attempted to systematically capture design and functional knowledge, manufacturing industries are still struggling with this knowledge integration issue, while they are globalized and highly competitive.

The global economical and technological environment poses a challenge of how to realize a true collaborative environment. In recent years, the Information Technology (IT) has evolved rapidly and has made enormous impact on the whole spectrum of industries. To overcome the following challenges in the practice of collaborative product development processes, various IT applications and CAX (Computer Aided technology) tools in manufacturing are required: 1) lack of information from suppliers and working partners; 2) incompleteness and inconsistency of product knowledge within the collaborating group; 3) incapability of processing information/data from other parties due to interoperability. Furthermore, improving collaboration using collaborative tools among distributed

design groups is one of critical issues in product development decision. Lack of all product development knowledge in the design stage causes many problems in the different stages.

The product development process, one of the most critical business processes, is foster corporate success in today's global market environment. Design rationale plays an important role in the product development of large and complex systems. Design rationale has many benefits. It can be used to verify and trace the design of a product. Despite its usefulness, design rationale is often not documented and the knowledge is evaporated or eroded after the product design is completed. Without such knowledge, impacts of proposed changes to the system cannot be assessed accurately. Problems in various product life-cycle including product design may arise when the expertise is no longer available or the knowledge is forgotten. The lack of product development knowledge reuse has been long recognized as a critical problem in product development [Ullman 1997]

There is need for a framework, which aims to understand and capture product development knowledge, to integrate multi-disciplinary knowledge of multiple stakeholders, and to establish causal knowledge management system for the better product. It seems currently available tools and techniques are not entirely suitable and effective enough to handle the challenges/pressures faced by product development processes.

3.2 Knowledge in Product Development

One view of knowledge is that of a condition of access to information [McQueen 1998]. According to this view, product development knowledge must be organized to facilitate access to and retrieval of content [Maryam 2001]. This view may be thought of as an extension of the view of product development knowledge as an object, with a special emphasis on the accessibility of the knowledge objects. If product development knowledge is viewed as an object, or is equated with information access, then knowledge management should focus on building and managing knowledge. If product development knowledge is a process, then the implied knowledge management focus is on knowledge flow and the processes of creation, sharing, and distribution of knowledge. The view of knowledge as a capability suggests a knowledge management perspective centered on building core competencies, understanding the strategic advantage of know-how, and creating intellectual capital.

The major implication of these various conceptions of knowledge is that each perspective suggests a different strategy for managing the knowledge and a different perspective of the role of systems in support of knowledge management. Table 3-1 summarizes the various views of knowledge just discussed and their implications for knowledge management and knowledge management systems [Maryam 2001].

Table 3-1 Knowledge Perspectives and Their Implications [adopted from Maryam 2001]

Perspectives		Implications for Knowledge Management (KM)	Implications for Knowledge Management System (KMS)
Knowledge vis-à-vis data and information	Data is facts, raw numbers. Information is processed / interpreted data. Knowledge is personalized information	KM focuses on exposing individuals to potentially useful information and facilitating assimilation of information	KMS will not appear radically different from existing IS, but will be extended toward helping in user assimilation of information
State of mind	Knowledge is the state of knowing and understanding	KM involves enhancing individual's learning and understanding through provision of information	Role of IT is to provide access to sources of knowledge rather than knowledge itself
Object	Knowledge is an object to be stored and manipulated.	Key KM issue is building and managing knowledge stocks	Role of IT involves gathering, storing, and transferring knowledge
Process	Knowledge is a process of applying expertise	Km focus is on knowledge flows and the process of creation, sharing, and distributing knowledge	Role of IT is to provide link among sources of knowledge to create wider breadth and depth of knowledge flows
Access to information	Knowledge is a condition of access to information	KM focus is organized access to and retrieval of content	Role of IT is to provide effective search and retrieval mechanisms for locating relevant information
Capability	Knowledge is the potential to influence action.	KM is about building core competencies and understanding strategic know-how	Role of IT is to enhance intellectual capital by supporting development of individual and organizational competencies.

Product development knowledge management systems refer to a class of information systems applied to managing product development knowledge. They are IT-based systems developed to support and enhance the product development processes of knowledge creation, storage/retrieval, transfer, and application [Maryam 2001]. Many KM initiatives rely on IT as an important enabler. While IT does not apply to all of the issues of KM, it can support KM in sundry ways. Examples include finding an expert or a recorded source of design knowledge

using online directories and searching databases; sharing knowledge and working together in collaborative teams; access to design case/information on past product development projects; and learning about customer needs and behavior by analyzing transaction data among others [KPMG 1998].

One of the most common applications is internal benchmarking with the aim of transferring internal best practices [KPMG 1998; O.Dell 1998]. For example, a common application of knowledge management is the creation of corporate directories, also referred to as the mapping of internal expertise. Because much knowledge in an organization remains un-codified, mapping the internal expertise is a potentially useful application of knowledge management [Ruggles 1998]. Such perception of the failure to apply existing knowledge is an incentive for mapping internal expertise. Another common application of knowledge management systems is the creation of knowledge networks [Ruggles 1998]. For example, when Chrysler reorganized from functional to platform based organizational units, they realized quickly that unless the suspension specialists could communicate easily with each other across platform types, expertise would deteriorate. Chrysler formed Tech Cul, bringing people together virtually and face-to-face to exchange and build their collective knowledge in each of the specialty areas. In this case, the knowledge management effort was less focused on mapping expertise or benchmarking than it was on bringing the experts together so that important knowledge was shared and amplified. Providing online forums for communication and discussion may form knowledge networks. In another case, Ford found that

just by sharing knowledge, the development time for cars was reduced from 36 to 24 months, and through knowledge sharing with dealers, the delivery delay reduced from 50 to 15 days [Gazeau 1998].

3.3 Issues in Product Development Knowledge

The advent of the Internet and World Wide Web ushered in a new wave of research on the collaborative product development environment. There are two major research areas in this field: one, research on how to manage product life-cycle knowledge effectively within a distributed enterprise environment; two, how to reuse design and manufacturing knowledge and repurpose it to new product design.

3.3.1 Design Knowledge Reuse Issues

Baxter and Gao's research addresses design knowledge reuse issues and the next step of design reuse research [Baxter 2007]. They noted that approximately 20% of the designer's time is spent searching for and absorbing information. Furthermore, approximately 40% of all design information requirements are met by personal information storage, despite the fact that more appropriate information may be available from other sources. Even if knowledge stored in computer based systems is accessed, if it is to be reused, several additional factors must be met: reusability, availability, and relevance. Efficient exploitation of past designs has been prohibited by the lack of a complete or consistent methodology to structure past designs and information [Shahin 1999]. With a well-structured library of past designs and a method to make new design reusable, the issue of design reuse

would be greatly simplified. Busby provided a detailed study into problems with design reuse [Busby 1999]. Most reuse issues that Busby presented were cases of reuse not taking place, belief that reuse was desirable but not practiced. The next most common problem was an unexpected amount of additional effort to reuse. Others were knowledge loss through inappropriate replication, and error where existing designs were reapplied to new purposes.

Design reuse remains a developing area, and many approaches have been developed. Further effort is required to understand the needs of knowledge users and producers so that appropriate methods can be applied [Busby 1999, Markus 2001, Finger 1998]. Existing methods to reuse design knowledge are generally not compatible with the whole product design process: some are suitable in conceptual design; most are focused on detail design. Further research is needed to explore the potential of an integrated product development knowledge approach. This should include non-geometric knowledge such as problem solving methods, solution generation strategies, design intent and project knowledge. These knowledge types are associated with the variety of tasks in today's design process.

3.3.2 Product Development Knowledge Management Issues

In looking at managing product life-cycle knowledge, research topics have focused on integrating product and process information temporally and spatially. The product information for the whole life cycle needs to be stored, retrieved, and integrated enterprise-wide. The accessibility, security, and integrity of information are the major concerns. By merging the processes of the design documentation

and the design data management via linking CAD drawings with external, network-accessible relational databases, integrated geometric information and related documentation can be shared enterprise-wide [Dong 1998, Gable 2005, Huang 1999, Kan 2001]. This research utilizes the existing network protocols to achieve enterprise-wide communication. Other research focuses on agent-based communication methodology over networks. Those researchers [Kumar 1994, Sriram 1993, Huang 2000] considered the following issues vis-à-vis the collaborative design system: multimedia engineering documentation, messages and annotations organization, negotiation/constraint management, design, visualization, interfaces, and web communication and navigation among agents.

Knowledge loss because of retirement, downsizing, and turnover is not only one of the costliest problems, but it also one of the most widely ignored problems facing organizations today. The Accenture Institute for Strategic Change [DeLong 2003] found that organizational innovation is often compromised due to knowledge loss. The special importance of an organizational memory has been stressed by many management thinkers recently. Memory is described "as a system of knowledge and capabilities that preserves and stores perceptions, actions and experiences over time and secures the possibility of recall for the future" [Romhardt 1997].

However, preservation of knowledge is a daunting challenge. Different strategies for knowledge preservation have been considered [Bott 2007]. As information is stored increasingly in electronic formats, there is a need to

reexamine the principles of preservation under which we have traditionally been trained and under which we are still often guided in decision making. These need to be reevaluated and compared so that "points of convergence or divergence can be evaluated." [Cloonan 1993]. In addition, the need to optimize organization processes rather than individual benefits poses challenges [Rangan 2005]. The importance of a lifecycle-wide knowledge sourcing strategy in support of the Enterprise System (ES) investment is articulated [Gable 2005]. Considerable research has been done in knowledge engineering (KE) and using new technologies [Matsumoto 2005, Barnard 2003, O'Hara 2002]. Knowledge Management (KM) specialists have sometimes failed to recognize the synergy that KE methodologies and tools hold to enhance the state of the art in practical domains [Liebowitz 2001].

According to an article published in Boeing Frontiers [Arkell 2007], 80 percent of a company's knowledge resides only within the minds of its employees. There is a threat of lost knowledge from an aging workforce [DeLong 2004]. A few tools currently being used at Boeing and other companies include tools such as an initiatives database to allow employees to search best practices, communities of practice for employees to share success stories, internal wiki services, video-taped training sessions, and recruiting retired scientists as expert consultants [Ledbetter Ledbetter 2007, Blanton 2007, Shneiderman 2007]. There is a need for research in expanding these and integration with engineering workflow to allow continuing capture, retention, and utilization of this knowledge.

Recently, a paradigm called mass collaboration is emerging for harnessing the knowledge and creativity. It is based on the collaboration and competition of large groups of people in innovative ways [Tapscott 2006]. An example of the success of mass collaboration is the free encyclopedia, Wikipedia.com. Currently, about 10 million volunteers collaborate to create an encyclopedia which consists of about 9.5 million articles in 256 languages. Its accuracy is comparable to that of Encyclopedia Britannica [Giles 2005]. Further, the large communities on sites such as Facebook for general networking (58 million users), Flickr for photo sharing (4 million users), LinkedIn for business networking (17 million users), and Soundpedia (3.5 million users) for music sharing show that individuals are increasingly participating in collaborations over the Internet at massive scales. Recently, there have been a few efforts on applying the concepts of open-source development to physical product development [OpenMoko 2008, Open 2008, Baker 2006].

3.4 Procedural Knowledge and Causal Knowledge

3.4.1 Procedural Knowledge

Declarative knowledge (DK) and procedural knowledge (PK) are not terms that directly describe aspects or systems of the mind. Instead, they have meaning within a particular theoretical model of cognitive structure and function. Existing theories span a wide range of possibilities. Some theories make this distinction in simple and direct form, whereas others entirely lack the distinction. The nature of information is that an individual acquires, processes, stores in memory, and uses in judgment. PK represents the processes that act on DK; the sequences of

interrelated operations that transform, store, retrieve, or make decision based on DK. Ryle introduced into philosophy the distinction between knowing how and knowing that [Ryle 1949]. Similarly, Polanyi distinguished between tacit and explicit knowledge and argued that science depends heavily on tacit knowledge that cannot be made explicit [Polanyi 1958, Polanyi 1967]. In the late 1960s, researchers in the field of artificial intelligence introduced a distinction between declarative and procedural representations of knowledge, where the latter consisted of programmed functions for answering particular questions [Barr 1983].

The distinction between DK and PK was carried over into psychology by researchers such as Anderson (1997), although his procedures consist of specifiable rules [Anderson 1997]. In contrast, Ryle, Polanyi, and the “Artificial Intelligence three proceduralists” would reject the claim that PK can be captured by explicit rules. A related distinction was proposed by psychologists in the 1980s: implicit vs. explicit memory [Schacter 1996]. In contrast, Mandler (2004) decomposed generalized task knowledge into declarative and procedural components [Mandler 2004]. The declarative structure captures abstract knowledge about the task (e.g., to pick up an object, we must first find the object, reach to it, and then grasp it). The procedural structure captures knowledge about how to instantiate the abstract policy in a particular setting (e.g., we must use our left hand to pick up the object and use an enveloping grasp). With such decomposition, it is possible to represent task knowledge in a general, robust, and fault-tolerant way. The declarative structure of a task defines an abstract schema

that can guide an agent's behavior in the world, while the procedural substrate decorates this abstract schema with resources based on environmental context. Because different classes of psychological models suggest different conceptions of PK, there are four basic classes of models: flowcharts, stored-program models, proceduralization models, and parallel distributed processing models. To perform complex, real-world tasks, agents must constantly react to changes in highly complex, dynamic environments by selecting appropriate goals and performing actions to achieve and maintain those goals. Frequently, PK is represented by a collection of operators composed of preconditions and effects [Fikes 1971]. Aforementioned studies conclude that PK is essentially a set of learned behavioral routines. These learned activity sequences, or cognitive scripts, are distinctive in terms of activity or event content, activity ordering or sequence, or both.

One advantage of PK is that it can involve more senses, such as hands-on experience, practice at solving problems, understanding of the limitations of a specific solution, etc. Thus PK can frequently eclipse theory. However, one limitation of PK is its job-dependence and so it tends to be less general than DK. For example, a product designer might have knowledge about a joining system (e.g., welding, riveting, adhesive bonding, fastening, etc.) for assembly design, whereas a welding designer might only know about a specific welding process for assembly. Thus the hands-on expertise and experience of the welding assembly designer might be of commercial value only to welding job-shops.

3.4.2 Causal Knowledge

It is always essential but difficult to capture incomplete, partial or uncertain product development knowledge during the product development processes to achieve interoperability among heterogeneous product development processes. This chapter presents one method to capture these product development knowledge based on cause and effect representation.

3.4.2.1 Causality

Causality has taken many journeys in the minds of men for over human history. One of the world-view is determinism, which is no more than a chain of events following one after another according to the law of cause and effect. Interpreting causation as a deterministic relation means that if A causes B, then A must always be followed by B. Informally, A probabilistically causes B if A's occurrence increases the probability of B. Though philosophers have pointed out the difficulties in establishing theories of the validity of causal relations, there is yet the plausible example of causation afforded daily which is our own ability to be the cause of events. When experiments are infeasible, the derivation of cause effect relationship from observational studies must rest on some qualitative theoretical assumptions, for example, the symptoms do not cause diseases with expression in the form of missing arrows in causal graphs such as Bayesian Belief Networks. The theory of "Causal Calculus" [Pearl 2000] permits one to infer interventional probabilities from conditional probabilities in causal Bayesian Belief Networks with unmeasured variables. One very practical result of this theory is the

characterization of confounding variables, which are a sufficient set of variables that would yield the correct causal effect between variables of interest.

While derivations in causal calculus rely on the structure of the causal graph, parts of the causal structure can be learned from statistical data under certain assumptions. The basic idea goes back to a recovery algorithm developed by Rebane and Pearl [Rebane 1987] and rests on the distinction between the three possible types of causal substructures allowed in a directed acyclic graph (DAG): $X \rightarrow Y \rightarrow Z$ (type 1), $X \leftarrow Y \rightarrow Z$ (type 2), $X \rightarrow Y \leftarrow Z$ (type3). Type 1 and type 2 represent the same statistical dependencies (i.e., X and Z are independent given Y) and are indistinguishable. However, type 3 can be uniquely identified, since X and Z are marginally independent and all other pairs are dependent. Thus, while the skeletons (the graphs stripped of arrows) of these three triplets are identical, the directionality of the arrows is partially identifiable. Algorithms have been developed to systematically determine the skeleton of the underlying graph and, then, orient all arrows whose directionality is dictated by the conditional independencies observed [Pearl 2000, Spirtes 1991, Spirtes 1993, Verma 1990].

3.4.2.2 Bayesian Belief Network

For over two decades, Artificial Intelligence (AI) researchers have used Bayesian Belief Networks (BBN) to encode expert knowledge and AI researchers and statisticians developed methods for learning Bayesian Belief Networks. BBN is an annotated directed graph that encodes probabilistic relationships among distinctions of interest in an uncertain reasoning problem [Howard 1981, Pearl

1998]. The representation formally encodes the joint probability distribution for its domain. BBN uses a graphical structure to represent causal relationships and probability calculus to quantify these relationships and update beliefs given new information. Pearl, in 1986 [Pearl 1986] and later in 1988 [Pearl 1988], introduced the concept of conditional independence for a more tractable and efficient evidence propagation mechanism. Since then, BBN has become a practical tool for reasoning under uncertainty. BBN has had considerable number of real-world applications, such as MIT's Heart Disease Program for differential therapy of cardiovascular disorders [Long 1989], Microsoft's Lumiere Project for inferring the goals and needs of software users [Horvitz 1998], Hewlett Packard's SACSO project for automatic customer support operations [Skaanning 2000], and change impact analysis in architecture design [Tang 2007].

Despite the efficient evidence propagation mechanism and powerful reasoning capability, knowledge elicitation from domain experts has never been easy in BBN, for two main reasons [Das 2004]. First, the number of probability values required to populate a Conditional Probability Table (CPT) grows exponentially with the number of parent nodes associated with the table. Second, the elicitation of conditional probability distributions from a domain expert is a very complex task and it requires a systematic approach to handle. Even though there are many applications of BBN in various decision support systems, to the best of our knowledge, there is no existing research ever applied BBN to product design decision support. So far the closest to our work is the application of BBN in change

impact analysis in the domain of architecture design [Tang 2007].

This chapter presents the BBN [Pearl 2000]. Figure 3-1 illustrates a simple typical BBN. It describes the causal relationships among the season of the year (X_1), whether it's raining (X_2), whether the sprinkler is on (X_3), whether the pavement is wet (X_4), and whether the pavement is slippery (X_5). Here, the absence of a direct link between X_1 and X_5 , for example, captures our understanding that there is no direct influence of season on slipperiness – the influence is mediated by the wetness of the pavement. (If freezing is a possibility, then a direct link could be added.)

Perhaps the most important aspect of a BBN is that they are direct representations of the world, not of reasoning processes. The arrows in the diagram represent real causal connections and not the flow of information during reasoning (as in rule-based systems and neural networks). Reasoning processes can operate on BBN by propagating information in any direction. For example, if the sprinkler is on, then the pavement is probably wet (prediction); if someone slips on the pavement, that also provides evidence that it is wet (abduction). On the other hand, if we see that the pavement is wet, that makes it more likely that the sprinkler is on or that it is raining (abduction); but if we then observe that the sprinkler is on, that reduces the likelihood that it is raining (explaining away). It is this last form of reasoning, explaining away, that is especially difficult to model in rule-based systems and neural networks in any natural way.

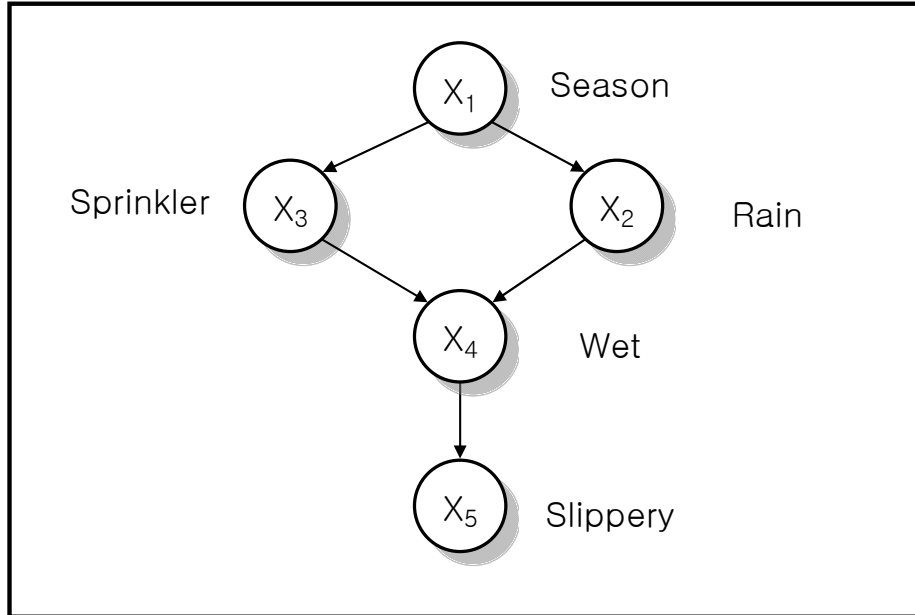


Figure 3-1 A Bayesian Belief Network representing Causal influence among five variables

Probabilistic semantics: any complete probabilistic model of a domain must, either explicitly or implicitly, represent the joint distribution - the probability of every possible event as defined by the values of all the variables. There are exponentially many such events, yet BBN achieve compactness by factoring the joint distribution into local, conditional distributions for each variable given its parents. If x_i denotes some value of the variable X_i and pa_i denotes some set of values for X_i 's parents, then $P(x_i|pa_i)$ denotes this conditional distribution. For example, $P(x_4|x_2,x_3)$ is the probability of wetness given the values of sprinkler and rain. The global semantics of BBN specifies that the full joint distribution is given by the product

$$P(x_1, \dots, x_n) = \prod_i P(x_i | pa_i) \quad (1)$$

In this example network, we have

$$P(x_1, x_2, x_3, x_4, x_5) = P(x_1)P(x_2|x_1)P(x_3|x_1)P(x_4|x_2, x_3)P(x_5|x_4) \quad (2)$$

Provided the number of parents of each node is bounded, it is easy to see that the number of parameters required grows only linearly with the size of the network, whereas the joint distribution itself grows exponentially. Further savings can be achieved using compact parametric representations – such as noisy-OR models, decision trees, or neural networks – for the conditional distributions.

There is also an entirely equivalent local semantics, which asserts that each variable is independent of its non-descendants in the network given its parents for example, the parents of X_4 in Figure 3-1 are X_2 and X_3 and they render X_4 independent of the remaining non-descendant, X_1 . that is,

$$P(x_4|x_1, x_2, x_3) = P(x_4|x_2, x_3) \quad (3)$$

The collection of independence assertions formed in this way suffices to derive the global assertion in Equation 1, and vice versa. The local semantics is most useful in constructing BBN, because selecting as parents the direct cause of a given variable automatically satisfies the local conditional independence conditions. The global semantics leads directly to a variety of algorithms for reasoning.

Evidential reasoning: From the product specification in Equation 1, one can express the probability of any desired proposition in terms of the conditional probabilities specified in the network, for example, the probability that the sprinkler is on, given that the pavement is slippery, is

$$\begin{aligned}
P(X_3 = on \mid X_5 = true) &= \frac{P(X_3 = on, X_5 = true)}{P(X_5 = true)} \\
&= \frac{\sum_{x_1, x_2, x_4} P(x_1, x_2, X_3 = on, x_4, X_5 = true)}{\sum_{x_1, x_2, x_3, x_4} P(x_1, x_2, x_3, x_4, X_5 = true)} \\
&= \frac{\sum_{x_1, x_2, x_4} P(x_1)P(x_2 \mid x_1)P(X_3 = on \mid x_1)P(x_4 \mid x_2, X_3 = on)P(X_5 = true \mid x_4)}{\sum_{x_1, x_2, x_3, x_4} P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1)P(x_4 \mid x_2, x_3)P(X_5 = true \mid x_4)}
\end{aligned} \tag{4}$$

These expressions can often be simplified in ways that reflect the structure of the network itself.

Learning in BBN: The conditional probabilities $P(x_i \mid pa_i)$ can be updated continuously from observational data using gradient-based method that uses just local information derived from inference [Lauritzen 1995] – in much the same way as weights are adjusted in neural networks. It is also possible to learn the structure of the network, using methods that trade off network complexity against degree of fit to the data [Friedman 1998]

Uncertainty over time: Entities that live in a changing environment must keep track of variables whose values change over time. Dynamic BBN [Dean 1989] capture this process by representing multiple copies of the state variables, one for each time step. A set of variables X_t denotes the world state at time t and a set of sensor variables E_t denotes the observations available at time t . The sensor model $P(E_t \mid X_t)$ is encoded in the conditional probability distributions for the observable variables, given the state variables. The transition model $P(X_{t+1} \mid X_t)$ relates the state at time t to the state at time $t+1$. Keeping track of the world means computing the current probability distribution over world states given all past observations, i.e.,

$P(X_t|E_1, \dots, E_t)$. Dynamic BBN are strictly more expressive than other temporal probability models such as hidden Markov models and Kalman filters.

Causal networks: Most probabilistic models, including general BBN, describe a distribution over possible observed events – as in Equation 1 – but say nothing about what will happen if a certain intervention occurs. For example, what if I turn the sprinkler on? What effect does that have on the season, or on the connection between wetness and slipperiness? A causal network, intuitively speaking, is a BBN with the added property that the parents of each node are its direct causes – as in Figure 1. In such a network, the result of an intervention is obvious: the sprinkler node is set to $X_3=on$ and the causal link between the season X_1 and the sprinkler X_3 is removed. All other causal links and conditional probabilities remain intact, so the new model is

$$P(x_1, x_2, x_4, x_5) = P(x_1)P(x_2|x_1)P(x_4|x_2, X_3=on)P(x_5|x_4) \quad (5)$$

Causal networks are more properly defined, then, as BBN in which the correct probability model after intervening to fix any node's value is given simply by deleting links from the node's parents. For example, Fire \rightarrow smoke is a causal network whereas Smoke \rightarrow Fire is not, even though both networks are equally capable of representing any joint distribution on the two variables. Causal networks model the environment as a collection of stable component mechanisms. These mechanisms may be reconfigured locally by interventions, with correspondingly local changes in the model. This, in turn, allows causal networks to be used very naturally for prediction by an agent that is considering various courses of action.

Causal discovery: One of the most exciting prospects in recent years has been the possibility of using BBN to discover causal structures in raw statistical data [Spirtes 1993, Pearl 2000] – a task previously considered impossible without controlled experiments. Consider, for example, the following intransitive pattern of dependencies among three events: A and B are dependent, B and C are dependent, Yet A and C are independent. If you ask a person to supply an example of three such events, the example would invariably portray A and C as two independent causes and B as their common effect, namely, $A \rightarrow B \leftarrow C$. (For instance, A and C could be the outcomes of two fair coins, and B represents a bell that rings whenever either coin comes up heads.) Fitting this dependence pattern with a scenario in which B is the cause and A and C are the effects is mathematically feasible but very unnatural, because it must entail fine tuning of the probabilities undergo a slight change.

Such thought experiments tell us that certain patterns of dependency, which are totally void of temporal information, are conceptually characteristic of certain causal directionalities and not others. When put together systematically, such patterns can be used to infer causal structures from raw data and to guarantee that any alternative structure compatible with the data must be less stable than the one(s) inferred; namely, slight fluctuations in parameters will render that structure incompatible with the data

Bayesian Belief Networks for Supervised Learning and Unsupervised Learning: The local distribution functions are essentially classification models.

Therefore, if we are doing supervised learning where the explanatory (input) variables cause the outcome (target) variable and data is complete, then the Bayesian-network and classification approaches are identical. When data is complete but input/target variables do not have a simple cause/effect relationship, tradeoffs emerge between the BBN approach and other methods.

The search algorithms of Spirtes et al. (1993) provide one method for identifying possible hidden variables in such situations. Martin and VanLehn (1995) suggest another method. Their approach is based on the observation that if a set of variables are mutually dependent, then a simple explanation is that these variables have a single hidden common cause rendering them mutually independent. Thus, to identify possible hidden variables, we first apply some learning technique to select a model containing no hidden variables. Then, we look for sets of mutually dependent variables in this learned model. For each such set of variables (and combinations thereof), we create a new model containing a hidden variable that renders that set of variables conditionally independent. We then score the new models, possibly finding one better than the original.

3.5 Ontology and Semantic Web

The original version of Tim Berners-Lee's WWW included meta-data above and beyond the current web, that is, additional information that was machine-interpretable [W3C-WWW 1992]. The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. In other words, the Semantic Web is the Web with

inference capabilities. The point of the Semantic Web is not just to make applications smarter, but also to make data smarter [Daconta 2003]. Data does not/should not reside in application specific databases. Data can become smarter through the use of higher semantics from technologies such as concept maps or ontologies. Ontologies are explicit formal specifications of the terms in the domain and relations among them [Gruber 1993]; a formal, explicit specification of a shared conceptualization. "Conceptualization" refers to an abstract model of some phenomenon in the world which identifies the relevant concepts of that phenomenon. "Formal" refers to the fact that the ontology should be machine-readable [Fensel 2001]. Ontological engineering is the successor of knowledge engineering and is viewed as a challenge to enabling knowledge sharing and reuse, which knowledge engineering failed to realize. Mizoguchi [Mizoguchi 2003] presented the roles of an ontology as common vocabulary, data structure, explication of what is left implicit, semantic interoperability, explication of design rationale, systemization of knowledge, meta-model function, and theory of content.

Ontologies have been developed for a variety of domains, most of them being broad. The broadest of ontologies, are the upper-level ontologies that describe common sense-level knowledge. CYC, developed by Cycorp, is a commercial ontology containing over 200,000 terms and assertions. Its goal is to define high-level, common sense-type of concepts in a machine-interpretable manner. Potential applications for CYC include, online brokering of goods and services, enhanced virtual reality, improved machine translation, improved speech

recognition, data mining, true language processing, etc [Cycorp 2004]. However, since CYC is still a high-level ontology, it has not had a strong impact in the mechanical design domain. Nonetheless, in 1999, the National Institute of Standards and Technology (NIST) chose CYC as an ontology for further investigation in the manufacturing domain [Schlenoff 1999]. The results from this investigation lead to the development of Process Specification Language (PSL), which is a language that is generic enough to represent discrete manufacturing and construction process data [Gruninger 2003].

Narrower in scope than upper-level ontologies, enterprise-level ontologies attempt to formalize the practices and processes that occur within an organization. The level of concepts is enterprise specific and is meant to promote knowledge reuse with regard to business decisions and transactions. Enterprise Ontology [Uschold 1998] developed by the Artificial Intelligence Applications Institute (AIAI) at the University of Edinburgh, is an ontology intending to define the overall activities of an organization. While this ontology takes into account the business aspects of an organization, it does not in detail define engineering activities. Similar to the overall goal of Enterprise Ontology, TOVE [Fox 1992, 1998] is an ontology for enterprise knowledge. It is a composite of several smaller ontologies including ontologies defining activity, resource, organization, product requirements, quality, and costing. The results of TOVE particularly in the domain products and requirements, are closer to the knowledge-intensive tasks of engineering design than Enterprise Ontology, yet they still do not capture all detailed forms of

mechanical design knowledge. In the engineering domain, Lin et al. [Lin 1996] developed a Knowledge Aided Design (KAD) system to capture knowledge from engineering tasks, particularly those tasks related to engineering requirements. The ontology for their KAD system included many requirements including component structure, features, parameters, constraints, requirements, etc. Issues that they address for the motivation for their work include communication, traceability, completeness, consistency, document creation, and managing change. They used an object-oriented approach to implement their work. Some ontological research has been applied at both the conceptual and detailed design levels. Kitamura et al. [Kitamura 2002, 2003, 2004a, 2004b] successfully developed an ontology to represent functional design and deployed the ontology into industry. While their work captures the flow of something (e.g. the flow of fluids or parts in manufacturing), it has limitations on capturing complex mechanical phenomena. Horváth et al. [Horváth 1998] attempted to create an ontology for design features using ontology theory. They classify design concepts in terms of entities, phenomena, and situations.

To semantically capture design and functional knowledge, manufacturing industries have to concern the knowledge integration issue, which is that significant researchers (Lin et al. 1996; Horváth et al. 1998; Kitamura et al. 2004; Grosse et al. 2005) have attempted. While they are globalized and highly competitive, they are still struggling with the knowledge integration issue. In addition, the product development knowledge and rationale behind the product are infrequently captured

or retained in the industries because this knowledge is cumulative, creative, iterative, and evolutionary, specially, during product development processes. Ontology can handle the integration, sharing, and reuse issues of the cumulative and evolutionary product development knowledge if the roles of an ontology in product development, which are common vocabulary, data structure, explication of what is left implicit, semantic interoperability, explication of design rationale, systemization of knowledge, meta-model function, and theory of content, is realized.

3.6 Analysis in Product Development

The improvement of the complexity of a product in a hard concurrent marketing context encourages the managers to give more importance to the maintenance functions. The industrial monitoring, which is one of the most significant of them, is divided into two tasks: the fault detection, and the fault diagnosis [Wan 1999]. More the product is complex, more the monitoring is difficult. The heterogeneity of maintenance and product information is taken into account for the creation of the monitoring system. This information can be provided by various techniques, such as Failure Modes and Effects Analysis (FMEA Gilchrist 1993], Fault Tree (FT)[Vesely 1981], Functional Analysis (FA)[Hansen 2006], Production and Operations Management (POM)[Thierry 1993], Computerized Maintenance Management System (CMMS)[Niebel 1994], and Supervisory Control and Data Acquisition (SCADA)[Neville 1986], to name a few. Most of these techniques work appropriately in the product development problems, but these have many

limitations. For example, in FMEA, the most well adapted technique, a risk is measured in terms of Risk Priority Number (RPN) that is a product of occurrence, severity, and detection difficulty. Furthermore, measuring severity and detection difficulty is very subjective and with no universal scale. RPN is also a product of ordinal variables, which is not often meaningful as a proper measure. Generally, the enlisted techniques above inhibit the understanding of the true cause of failures and fault chains [Lee, 2000].

It seems currently available tools and techniques are not entirely suitable and effective enough to handle the challenges/pressures faced by product development processes. An interesting example demonstrating the inability of current methods to deal with product complexity is the amount of knowledge generated through failure analysis. The internal study of one of the US automakers reports that 35-40% of field failure issues are related to system interactions. Most of the manufacturing organizations use traditional FMEA technique for failure analysis in product development processes. However, the traditional FMEA is very tedious, painstakingly time consuming, and prone to errors of inconsistency and incompleteness, and hence unable to support development time reduction strategy beyond a limit. There are well established failure analysis (physics-of-failure) models for individual components, but when they are assembled together in a complex system, the failure behavior is often totally different. The FMEA fails to capture potential system interactions effects of complex products, dynamic behavior of the system, and its effects on system failure mechanisms. Moreover,

the product complexity leads to an emergence of unpredictable failure patterns and the FMEA is unable to anticipate these unpredictable failure patterns. Therefore, modeling of failure dependency and failure interactions among the components/modules of a system presents yet another interesting research opportunity. Specifically, there is need for a framework, which aims to understand and capture failure dependence/interactions and to develop better understanding of the product behavior from the perspective of the end-user, and to establish fast and efficient information/knowledge database for the failure and behavior.

The early identification of the few faulty components is an important research endeavor in that it allows an organization to take mitigating actions by optimally allocating testing resources or by redesigning components [Harrison 1988]. The early identification of faulty components is commonly achieved through a binary quality model that classifies components into either a faulty or not-faulty category [Briand 1993, Lanubile 1997]. Early detection and isolation of faults as other research motivations are critical factors for avoiding product deterioration, loss of production, poor plant economy, performance degradation, major damage to machinery, environmental pollution and damage to human health or even loss of life. These motivations generate a great attention in fault detection and isolation in dynamic processes. A wide variety of “model-based” approaches have been proposed to tackle this problem [Patton 1989]. The conceptual realization of these models can vary according to the following approaches: the parity space and state estimation, the fault detection filter, and non-linear techniques for parameter

identification. In each case, appropriate mathematical models are required, either in state space or in input-output form to guarantee that faults can be detected and isolated. Existing studies have shown that, only if certain modeling and design conditions are satisfied, model-based methods can be useful for the detection and isolation of multiple faults [Simani 2006].

CHAPTER 4

PRELIMINARY STUDY

The aim of this chapter is a preliminary study for causal product knowledge management. The preliminary study is for systematic knowledge elicitation and FCM-BBN constructor, which is co-work with chonnam national university in South Korea [Kim 2008].

4.1 Systematic Knowledge Elicitation and FCM-BBN Constructor

Managing design knowledge is an important concern for industry, including engineering. Engineering firms are facing pressures to increase the quality of their products, to have even shorter lead times and reduced costs. There is also a trend towards globalization resulting in complex supply chains and the need to manage teams that are not necessarily co-located. Design knowledge needs to be exchanged and accessed efficiently. Other motivations for managing design knowledge are to provide a trail for product liability legislation and to retain design knowledge and experience as engineering designers retire. Fuzzy Cognitive Map (FCM) is one of the main formalisms for modeling, representing and reasoning about causal knowledge. Despite the fact that FCM has been used extensively in causal knowledge engineering, there is a lack of methodology for the systematic construction of FCM. Although some techniques were used in the individual construction processes, these techniques were either not systematically documented or too specific to the problem at hand. FCM and Bayesian Belief Network (BBN) are two major frameworks for modeling, representing and

reasoning about causal design knowledge. Despite their extensive use in causal design knowledge engineering, there is no reported work which compares their respective roles. This research deals with three topics, which are systematic constructing FCM, a methodology for FCM-BBN conversion, and comparison FCM and BBN. BBN has a sound mathematical foundation and reasoning capabilities, also it has an efficient evidence propagation mechanism and a proven track record in industry-scale applications. However, BBN is less friendly and flexible, and often very time-consuming to generate appropriate conditional probabilities. Thus, FCM is used for the indirect knowledge acquisition, and the causal knowledge in FCM is systematically converted to BBN. Finally, we compare BBNs directly generated by domain experts and generated from FCM, with a realistic industrial example, a fuel nozzle for an aerospace engine.

4.1.1 Comparison of FCM and BBN

The roles of FCM and BBN in the knowledge engineering of causal reasoning systems have been compared. The knowledge engineering process includes knowledge acquisition, knowledge representation and causal reasoning. The comparison is done based on some inherent features of the frameworks which are independent of any specific applications. These features, such as usability, expressiveness, reasoning adequacy, formality and soundness, constitute the comparison criteria. The criteria are discrete because a framework is either having or not having a particular feature. Hence, the comparison is done in an objective and qualitative manner. Besides, a literature survey to compare the roles of the

frameworks in the knowledge engineering of some real applications (both research-based and industry-scaled) has been conducted with some conclusions related to the practicality of the frameworks.

The comparison results are summarized in Table 4-1. Overall, except for the modeling of dynamic system, BBN is, in general, more expressive and formal in representation as well as more powerful and sound in reasoning. The expressiveness in representation is attributed to the ability in handling uncertainty. The powerfulness in reasoning is attributed to the ability in performing backward diagnostic reasoning. The formality in semantics and soundness in inference is attributed to its solid foundation on probability theory. In addition, BBN is more superior because it has an efficient evidence propagation mechanism based on conditional independence and a proven track record in industry-scale applications. Unfortunately, BBN suffers from its complexity when used as a front-end modeling tool for capturing causal knowledge from the domain expert. Elicitation of causal knowledge from the domain expert, through the specification of CPTs is both unnatural and tedious. As a complement to it, FCM is an excellent front-end modeling tool. The visual graphical interface of FCM is both friendly and intuitive. It allows the domain expert to work at a higher level of abstraction as it hides the lower level details and focuses on the essentials.

Table 4-1 A Summary of Comparison Results

General Criterion	Specific Distinguishing Question	BBN	FCM	Remark
Usability in Modeling	What to construct essentially?	CPTs	Signed directed graph	
	What type of construction interface?	Tabular	Visual graphical	FCM is more user-friendly
	How to represent a causal relationship?	Probabilistic dependencies between variable states	A causal link between the variables	FCM is more direct in representation
	How to represent a causal strength?	Multiple conditional probability values in the CPT	Single value attached to the causal link	FCM is simpler in representation
	How obvious is the causal structure?	Implicitly represented in the CPTs	Explicitly represented on the graph	FCM is more intuitive
	What is the level of specification?	Variable states	Variables	FCM is more high-level
	How many values are required to specify a combination of causal effects?	The product of the number of possible states of the individual cause and effect variables	The number of cause variables or causal links	FCM is easier to handle
Expressiveness in Representation	Does it allow unequal likelihood of increase and decrease before any evidence?	Yes (user can decide and specify prior probabilities)	No (user has no control over initial likelihood)	BBN is more expressive
	Does it allow unrepresented causes?	Yes (effect of unrepresented causes is reflected in unequal prior probabilities)	No (assume all possible causes are represented)	BBN is more expressive
	Does it allow ignorance of individual causal effects of a combination?	Yes (it is only required to specify combination effect)	No (it is required to specify individual effects)	BBN is more expressive
	Does it allow ignorance of how individual causal effects are combined?	Yes (user estimates total effect if formula is unknown)	No (combination is only based on algebraic sum)	BBN is more expressive
	Does it allow feedback and causal loops?	No	Yes	FCM is more expressive
	Does it allow temporal representation?	No (it only supports static system)	Yes (it supports modeling of dynamic system)	FCM is more expressive
Adequacy in Reasoning	Does it support backward chaining?	Yes	No (it only supports forward chaining)	BBN is more powerful
	Does it support diagnostic reasoning?	Yes	No (it only supports predictive reasoning)	BBN is more powerful
	Does it have an efficient evidence propagation mechanism?	Yes (based on Pearl's conditional independence)	No	BBN is more practical
	Are there many commercially available powerful and efficient reasoning engines?	Yes (Netica, Hugin, etc.)	No	BBN is more practical
	Are there many industry-scale applications?	Yes (by Microsoft, Hewlett Packard, etc.)	No (restricted to research based applications)	BBN is more practical
Formality in Semantics & Soundness in Inference	Is it founded on sound mathematical theorems derivable from well-defined basic axioms?	Yes (founded on probability theory)	No	BBN has formal semantics
	Is the correctness of the inference mechanism provable?	Yes	No (Inference mechanism is rather ad hoc)	BBN has sound inference

From the comparison results, FCM has shown to be simpler, more intuitive, more high-level, and more user-friendly. These features make it very appropriate to be used at the front-end of knowledge engineering for the acquisition of causal knowledge from human experts. BBN, on the other hand, has shown to be more expressive, powerful, formal and sound. These features make it very appropriate to be used at the back-end of knowledge engineering for the representation and

automated reasoning by machine. The idea of integration is made possible by transforming FCM into BBN.

4.1.2 Systematic Construction of FCM

There are two basic components of a causal model constructed based on the knowledge elicited from a domain expert: domain variables which constitute factors to the problem at hand, and causal structure which describes the relationships between these variables. The elicitation of the domain variables and the causal structure are separately discussed in the following subsections.

4.1.2.1 Eliciting Variables Relevant to the Problem

The process is carried out through unstructured questions. These are exploratory and open ended questions, in which an expert is asked to list out all the domain variables relevant to the decision making. The elicitation process is carried out systematically as follow. First, the knowledge engineer is required to determine the first/main goal variable as the starting point for the elicitation process. Then, the domain expert is requested to enumerate factors contributing (either positively or negatively) to the first goal variable and these factors constitute the first-level variables. For each first-level variable, the expert is then requested to enumerate factors contributing to it and these are second-level variables. The probing process continues until the expert cannot think of any additional factors; or the expert feels that the additional factors are not significant to the problem at hand. The elicitation process is automatable and the role of a knowledge engineer can be significantly reduced or completely eliminated. The process can be implemented using queue

data structure.

At this stage, the domain expert will be notified for duplicate entry of the same variable. Hence, the interview process assimilates a breadth-first tree construction. The aim at this stage is to gather a complete set of relevant variables that make-up the problem domain, not their relationships. After having a complete set, V , of N domain variables, an $N \times N$ adjacency matrix, M , can be constructed for the representation of an expert's causal knowledge about the problem domain. The matrix can be represented as an $N \times N$ two dimensional array with M_{ij} be an entry at the intersection of i th row and j th column, where $i, j = \{1, 2, 3, \dots, N\}$. Each entry of the matrix is initialized with 0, that is $M_{ij} = 0$, representing no causal relationship has been assigned. An auxiliary 1-d array, A , is created to accompany the matrix M . It maps the indices of the matrix to their respective domain variables. Let V_i be the i th element of V , and A_j be the j th element of A , $A_i = V_j$ when $i = j$, where $i, j = \{1, 2, 3, \dots, N\}$.

4.2 Eliciting Causal Structure between Variables

After eliciting the domain variables, the next step is to request the domain expert to determine their mutual causal relationships. The causal effect can be positive (causal increase) or negative (causal decrease). The weight determines relative strength of the causal effect. It is easier for a human expert to specify discrete linguistic weights than continuous numerical weights. Hence, for each problem domain, a scheme for linguistic weights is to be determined by the

knowledge engineer before it is used for the elicitation of causal structures from the domain experts. After eliciting all the P positive linguistic variables, a one dimensional array, L , is created to store them. The array, L , can be considered as a one-to-one function which maps the positive integer causal values into their respective linguistic variables. L_i denotes the i th element of L , where $I = \{1,2,3,\dots,P\}$.

The causal structure elicitation process can be carried out through an interview using structured questions. These are closed questions with limited options for the answers. In the interview, the domain expert is requested to determine, for each variable, whether there is a causal link to the other variables. If a link exists, the domain expert is further requested to determine its sign and linguistic weight.

The causal structure elicited through structured interview or questionnaires is represented as a directed graph with feedback. As the causal relationships are added, the update is immediately reflected in the graph. This allows the human expert to observe and examine the growth of the causal structure. The cause-effect relationships elicited are also represented assigned integers into the appropriate entries of the adjacency matrix. Let W_{ij} be the causal weight, with a sign and a magnitude, for a link from variable i to variable j , elicited from a domain expert. The causal relationship can be assigned to the adjacency matrix as follow: $M_{ij}=W_{ij}$.

In the adjacency matrix elicited from a human expert, causal values are represented using integers drawn from a crisp set specific to the application at hand, such as $\{-5,-4,-3,-2,-1,0,1,2,3,4,5\}$. However, in FCM, it is a common practice to use real numbers for causal values drawn from a bipolar fuzzy interval, that is $[-1\dots0\dots1]$. A bipolar notation consists of a negative sub-interval $[-1\dots0)$, 0, and a positive sub-interval $(0\dots1]$. There are two advantages of using bipolar notation. First, it is more intuitive because it uses 0 for no causal effect, 1 for full or maximum causal effect, and real numbers in between 0 and 1 for causal effects with intermediate strength. Second, it captures more fine grain information and thus allows fuzzy functions to be used for defining causal strength.

It is often desirable to combine knowledge of multiple experts to obtain a collective view of a particular problem domain. Kosko, the author of FCM, has developed a mathematical method for combining the FCMs of multiple experts [Kosko 1988, 1995, 1997]. There are also some other works on knowledge fusion in FCMs [Taber 1987, 1991, 2007]. In our methodology, Kosko's mathematical formalism is used due to its simplicity. However, the methodology can be easily adapted for other formalisms by only changing the formula for computing the combination.

4.2.1 Defining the Augmented Matrix

In general, different FCMs specific to the same domain may consist of an unequal number of variables. This results in these FCM matrices having different sizes, hence, a need for an augmentation of the matrices to produce an augmented matrix which ensures conformity in addition. Suppose that in addition to

the FCM for the first expert mentioned above, there is a second expert's opinion captured in the form of a directed graph, and the corresponding adjacency matrix and auxiliary array. The set of domain variables proposed by the second expert is almost the same as those proposed by the first expert, except there is an additional variable called Productivity. The first expert proposed 10 variables and the second proposed 11, and there are 10 overlaps. Hence, the augmented matrix has 11 rows by 11 columns as there is a total of 11 (=10+11-10) distinct domain variables. The auxiliary array contains all of these 11 domain variables. The augmented matrix for the previous 2 matrices is shown in Figure 4-1.

		M3[i, j]												
		1	2	3	4	5	6	7	8	9	10	11		
1	0	0	0	0	0	0	0	0	0	0	0	0	1	Market Share
2	0	0	0	0	0	0	0	0	0	0	0	0	2	Competitiveness
3	0	0	0	0	0	0	0	0	0	0	0	0	3	Market Demand
4	0	0	0	0	0	0	0	0	0	0	0	0	4	Competitor's Advertisements
5	0	0	0	0	0	0	0	0	0	0	0	0	5	Sales Price
6	0	0	0	0	0	0	0	0	0	0	0	0	6	Assembly Quality
7	0	0	0	0	0	0	0	0	0	0	0	0	7	Quality Control
8	0	0	0	0	0	0	0	0	0	0	0	0	8	Economic Conditions
9	0	0	0	0	0	0	0	0	0	0	0	0	9	Assembly Design
10	0	0	0	0	0	0	0	0	0	0	0	0	10	Assembly Cost
11	0	0	0	0	0	0	0	0	0	0	0	0	11	Productivity

Figure 4-1 Augmented matrix for previous two matrices

4.2.2 Constructing the Additive Matrix

The augmented matrix M_3 is the super structure of the individual matrices, M_1 and M_2 , and it has zero content. The additive matrix for M_1 and M_2 is the augmented matrix M_3 , after it is added with the causal weights from M_1 and M_2 . It represents the combination of causal knowledge $K_3 (=K_1+K_2)$. Two related entries of M_1 and M_2 for the same cause and effect are averaged, and the result is recorded in the related entry of M_3 . It represents the average of the values proposed by the two experts. For example, if both experts say that a particular causal effect is 1, the resulting causal effect is also $1=(1+1)/2$. If one says that the causal effect is 1 and the other says that it is -1 , the resulting causal effect is $0=(1-1)/2$. If one says that a particular causal effect is 1 and the other says that it is 0, or without saying anything about it, the resulting causal effect is $0.5=(1+0)/2$.

The combination of causal knowledge of multiple experts can be done incrementally such that matrices are added two at a time. This approach allows the accumulation of new causal knowledge once it is elicited from a domain expert. The additive matrix after combining the causal knowledge of two experts is shown in Figure 4-2.

		M3[i, j]										A3[i]		
		1	2	3	4	5	6	7	8	9	10	11		
1		0	0	0	0.3	0	0	0	0	0	0	0	1	Market Share
2		0.7	0	0	0	0	0	0	0	0	0	0	2	Competitiveness
3		0.3	0	0	0	0	0	0	0	0	0	0	3	Market Demand
4		-0.9	0	0.15	0	0	0	0	0	0	0	0	4	Competitor's Advertisements
5		-0.5	-0.1	-0.1	0	0	0	0	0	0.15	0	0	5	Sales Price
6		0	0.7	0	0	0.5	0	-0.45	0	0	0.25	0	6	Assembly Quality
7		0	0.5	0.05	0.25	0	0	0	0	0.45	0	0	7	Quality Control
8		0	0	0.5	0	0	0	0	0	0	0.35	0	8	Economic Conditions
9		0	0.7	0.9	0	0.3	0	-0.15	0	0	0	0	9	Assembly Design
10		0	-0.9	0	0	0.9	0	0	0	0	0	0	10	Assembly Cost
11		0	0	0	0	-0.45	0	0	0	0	-2.5	0	11	Productivity

Figure 4-2 Additive matrix after combining causal knowledge of two experts

4.2.3 Converting Bipolar Values into Linguistic Weights

There are three possible ways to output the bipolar causal values of an additive matrix, depending on the target application at hand. First, a bipolar causal value can be returned as it is, to a higher level client, for subsequent causal reasoning or further computation because it is easy for a machine to manipulate real numerical causal values. Second, it can be returned as a crisp linguistic weight using an appropriate linguistic variable derived from a predefined set. Third, it can be returned as a fuzzy linguistic weight using a linguistic variable accompanied by a membership value. The second and third forms are normally targeted to human users because they appreciate qualitative weights better than quantitative values. Fuzzy linguistic weights are used when high precision is needed. Otherwise, crisp linguistic weights should be used.

4.3 Systematic Generation of BBN from FCM

Nadkarni et al. proposed a systematic approach for capturing causal knowledge from domain experts [Nadkarni 2004]. It includes a method for the elicitation of unstructured knowledge, with a set of open ended interview questions. It also includes a procedure for the subsequent derivation of environmental factors and initial causal structure. Figure 4-3 shows an example of an initial Cognitive Map (CM) for the assembly design decision (ADD) and environmental factors, elicited from the domain experts based on the FCM approach. This example will be used to explain the FCM and the method for mitigating FCM to BBN.

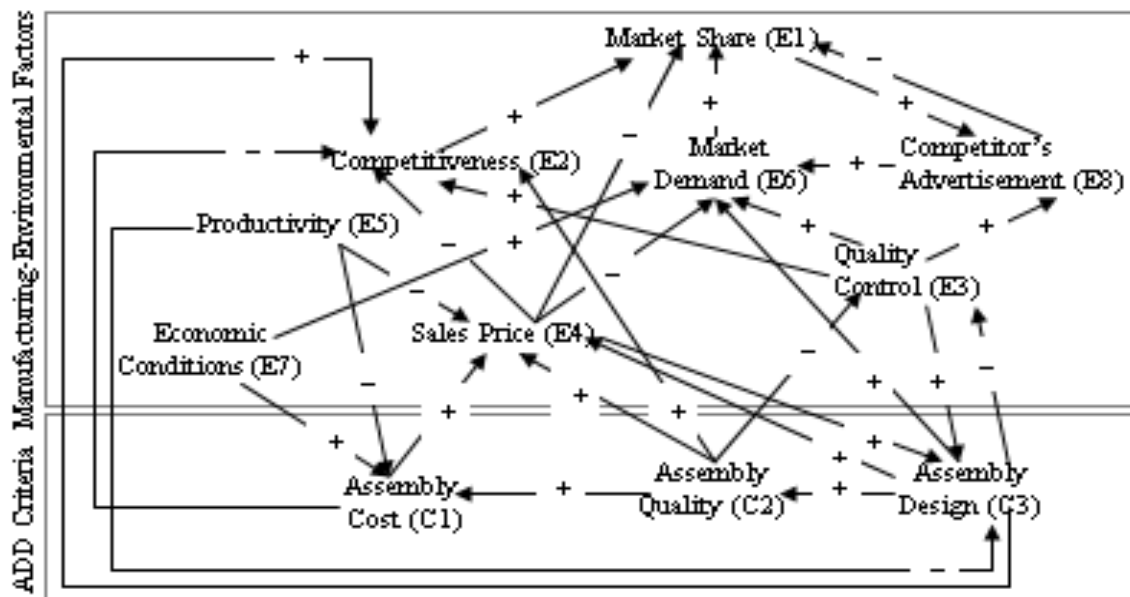


Figure 4-3 Causal Structure

In this methodology, causal weights or values are elicited from the experts and represented using an FCM. Conditional probability distributions can be derived directly from the causal values in the FCM. In general, an FCM with n nodes can be described by an $n \times n$ adjacency matrix, M_{ij} , whose elements, e_{ij} , are the causal value (representing causal strength) of the link directed out of V_i into V_j . Table 4-2 shows an adjacency matrix

for the FCM. In the table, the causal value for the link from C_2 to E_2 is 0.8, which indicates a strong positive causality from C_2 to E_2 .

Table 4-2 Adjacency Matrix for FCM

	E1	E2	E3	E4	E5	E6	E7	E8	C1	C2	C3	Notations:
E1	0	0	0	0	0	0	0	0.3	0	0	0	E1: Market Share
E2	0.7	0	0	0	0	0	0	0	0	0	0	E2: Competitiveness
E3	0	0.6	0	0	0	0.1	0	0.4	0	0	0.7	E3: Quality Control
E4	-0.5	-0.2	0	0	0	-0.2	0	0	0	0	0.3	E4: Sales Price
E5	0	0	0	-0.8	0	0	0	0	-0.3	0	-0.8	E5: Productivity
E6	0.4	0	0	0	0	0	0	0	0	0	0	E6: Market Demand
E7	0	0	0	0	0	0.6	0	0	0.4	0	0	E7: Economic Conditions
E8	-1	0	0	0	0	0.2	0	0	0	0	0	E8: Competitor's Advertisements
C1	0	-0.9	0	1	0	0	0	0	0	0	0	C1: Assembly Cost
C2	0	0.8	-1	0.5	0	0	0	0	0.6	0	0	C2: Assembly Quality
C3	0	0.8	-0.3	0.4	0	0.9	0	0	0	0.7	0	C3: Assembly Design

Table 4-3 BBN Compatible Adjacency Matrix

	E1	E2	E3	E4	E5	E6	E7	E8	C1	C2	C3	Notations:
E1	0	0	0	0	0	0	0	0	0	0	0	E1: Market Share
E2	0.7	0	0	0	0	0	0	0	0	0	0	E2: Competitiveness
E3	0	0	0	0	0	0	0	0	0.6	1	0.7	E3: Quality Control
E4	0	-0.2	0	0	0	-0.2	0	0	0	0	0	E4: Sales Price
E5	0	0	0	0	0	0	0	0	-0.3	0	0	E5: Productivity
E6	0.4	0	0	0	0	0	0	0	0	0	0	E6: Market Demand
E7	0	0	0	0	0	0.6	0	0	0.4	0	0	E7: Economic Conditions
E8	-1	0	0	0	0	0.2	0	0	0	0	0	E8: Competitor's Advertisements
C1	0	0	0	1	0	0	0	0	0	0	0	C1: Assembly Cost
C2	0	0.8	0	0	0	0	0	0	0	0	0	C2: Assembly Quality
C3	0	0.8	0	0	-0.8	0.9	0	0	0	0.7	0	C3: Assembly Design

4.3.1 Constructing BBN form FCM

The migration involves two stages, qualitative and quantitative. Qualitative migration involves the transformation of the qualitative structure of FCM. Quantitative migration involves the transformation of “fuzzified” causal weights or causal values into the conditional probability distributions in BBN. For each variable or node in the BBN compatible qualitative causal structure, there will be CPT

associated with it.

4.3.1.1 Building BBN Compatible Causal Structure form CM

The initial CM is less structured due to the way knowledge is elicited from the experts. The initial CM requires modification to make it compatible with BBN by performing four operations: 1) ensuring conditional independency; 2) removing indirect relationships; 3) converting abducted links to deductive; and 4) eliminating circular relations [Taber 1987]. The operations are elaborated below and the result is shown in Figure 4-4. In BBN, all the dependent nodes are to be linked with an arrow, so that when there is no link between two nodes, we can conclude that the nodes are conditionally independent. Only variables with direct causal relationships are linked with an arrow directly. Hence, the links between variables which are indirectly related are to be removed. The indirectly related variables are to be separated as conditionally independent variables. The direct links between the following pairs of variables were removed: (E3, E2), (E3, E6), (E3, E8), (E4, E1), (E5, E4), (C1, E2), (C2, C1), (C2, E4), (C3, E4). Each of them is substituted by one or more indirect links which indicate the propagation of causal effects.

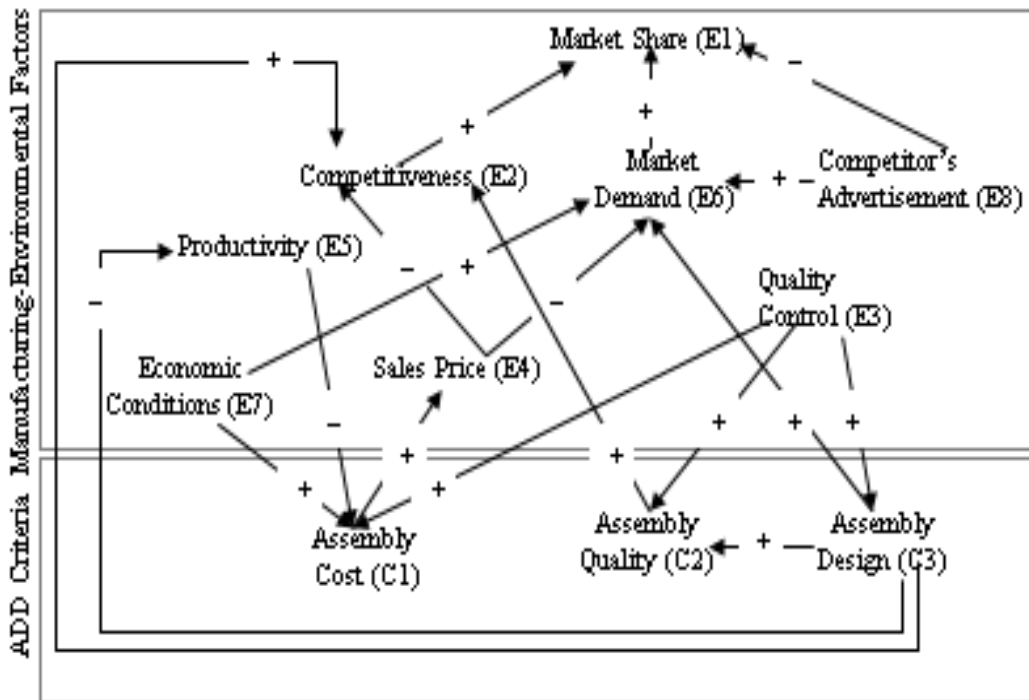


Figure 4-4 BBN Compatible Causal Structure

Causal statements involving abducted reasoning are often represented by a link from effect to cause in CM. These causal links are to be converted as links from cause to effect (i.e., in the direction of causation). The reverse, effects to cause, relationships will be inferred by using the probabilistic inference mechanism of BBN. In Figure 4-3, there is a negative link from productivity to assembly design. The link is abducted and it is removed. Instead, a negative deductive link from assembly design to productivity is added in Figure 4-4. The rationale is that a better (often more complex) assembly design usually requires more time and effort, hence lower productivity. In CM, circular relations violate the acyclic graphical structure required in BBN, hence, they are to be removed. In this work, we confine our model to the current time frame of the decision being modeled, and we remove

the link from market share to competitor's advertisement. There is a loop between the three variables: quality control, assembly design, and assembly quality. The negative link from assembly quality to quality control represents the fact that high quality assembly will require less control and managing in the future; though, currently, the high quality assembly is the result of the high quality control. Since the link pertains to the future time frame, it is removed.

4.3.1.2 Constructing CPTs of BBN from Causal Weights of FCM

The above four operations have modified the qualitative structure of the FCM making it compatible with BBN. The conversion has also changed the adjacency matrix and the result is shown in Table 4-4. This updated matrix is useful for constructing CPTs for the BBN. There are three steps involved in the construction of CPTs from causal weights: 1) summing the causal effects; 2) normalizing the tables; and 3) assigning probability to the opposite state.

Table 4-4 Three-step Construction of CPT for Assembly Cost (C1)

CPT for C1				
E3	E5	E7	C1	
			+	-
+	+	+	0.70	
+	+	-		0.10
+	-	+	1.30	
+	-	-	0.50	
-	+	+		0.50
-	+	-		1.30
-	-	+	0.10	
-	-	-		0.70

(a)

CPT for C1				
E3	E5	E7	C1	
			+	-
+	+	+	0.54	
+	+	-		0.08
+	-	+	1.00	
+	-	-	0.38	
-	+	+		0.38
-	+	-		1.00
-	-	+	0.08	
-	-	-		0.54

(b)

CPT for C1				
E3	E5	E7	C1	
			+	-
+	+	+	0.77	0.23
+	+	-	0.46	0.54
+	-	+	1.00	0.00
+	-	-	0.69	0.31
-	+	+	0.31	0.69
-	+	-	0.00	1.00
-	-	+	0.54	0.46
-	-	-	0.23	0.77

(c)

Table 4-4(a) shows a result of summing the causal effects from three sources: E3, E5, and E7. It shows the probability distributions for C1, one for each configuration of states of its parents. After summing the causal effects, the value for the '+' state of C1 is greater than 1 (i.e., 1.3), when both E3 and E7 increase but

E5 decreases. The value for the '–' state of C1 is also 1.3, when both E3 and E7 decrease but E5 increases. A probability value greater than 1 is not acceptable in probability theory. Therefore, a normalization process is necessary. Since we only want the relative strength, it is fine to modify the values, as long as their ratios remain unchanged. We normalize the probabilistic values by dividing each of them using the maximum value, which is greater than 1.

In Table 4-4(b), when both E3 and E5 increase, E7 decreases. The three factors, collectively, produce a causal effect of 0.08, to the '–' state of C1, which is the state of interest. We have no knowledge about the causal effect to the counterpart (i.e., the '+' state of C1). However, in BBN, the '+' state has to be assigned 0.92 ($1 - 0.08$). This causes a semantic problem because it implies that the collective effect from the three factors is more likely to cause an increase to C1 (0.92) than a decrease (0.08). This is commonly recognized as a limitation of the classical probability theory. We propose a simple and practical method which ensures the assigned probability is always smaller than the probability of the state of interest, though it does not eliminate the assignment of probability to the counterpart state.

Without any knowledge, we assume a prior probability of 0.5 for both '+' and '–' states of a variable. The value of 0.5 indicates absolute uncertainty of their likelihood. Once concrete evidence (complete certainty) is acquired for a particular state of interest, its probability immediately increases to 1, and the counterpart state immediately decreases to 0. Hence, the probability range of the state of

interest is 0.5. The minimum probability is 0.5 and the maximum probability is 1. The counterpart state, on the other hand, stays within 0 and 0.5; hence, it is always less than the state of interest. Suppose we are 50% sure (0.5 initial probability) that a variable will increase (i.e., in between absolute uncertainty and absolute certainty). Based on our proposed method, the moderated probability should fall exactly in between 0.5 and 1, which is 0.75. It can be computed using a simple proportionality formula, as follow: Moderated Probability = (Initial Probability × Probability Range) + Minimum Probability.

4.4 Case study: Fault Diagnosis for Fuel Nozzle

This case study for FCM-BBN conversion is based on a fuel nozzle of an aerospace jet engine. Two networks are created based on the domain expert for the fault diagnosis. These networks include ten design aspects and twenty different maintenance aspects of the fuel nozzle. Figure 4-5 illustrates FCM-BBN (BBN generated from FCM) and the other network is a traditional BBN (Figure 4-6). The both networks are showing the design stages, which are ten nodes located in left side in the networks, and the maintenance stages are others. Also, the Graphical Network Interface (GeNIe) is used to compare the performance of both networks. The GeNIe software package, which is developed by the University of Pittsburgh, can be used to create decision theoretic models intuitively using the graphical click-and-drop interface.

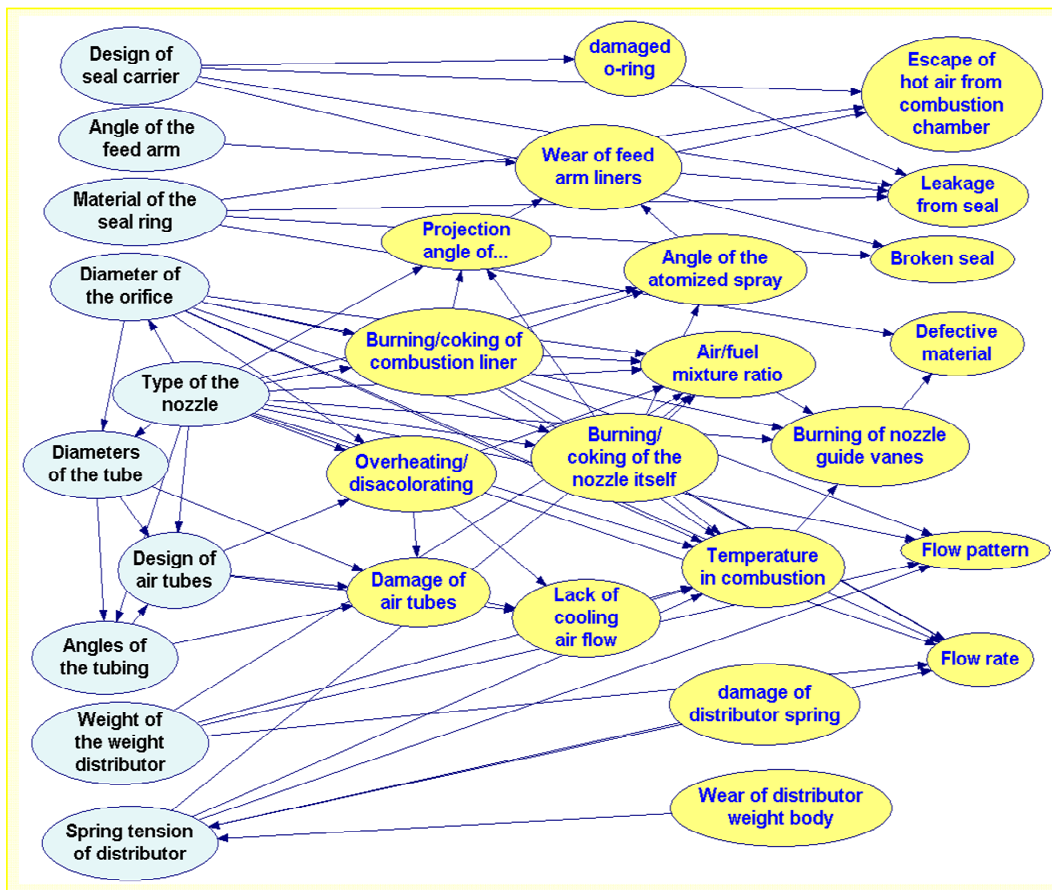


Figure 4-5 Network from FCM-BBN

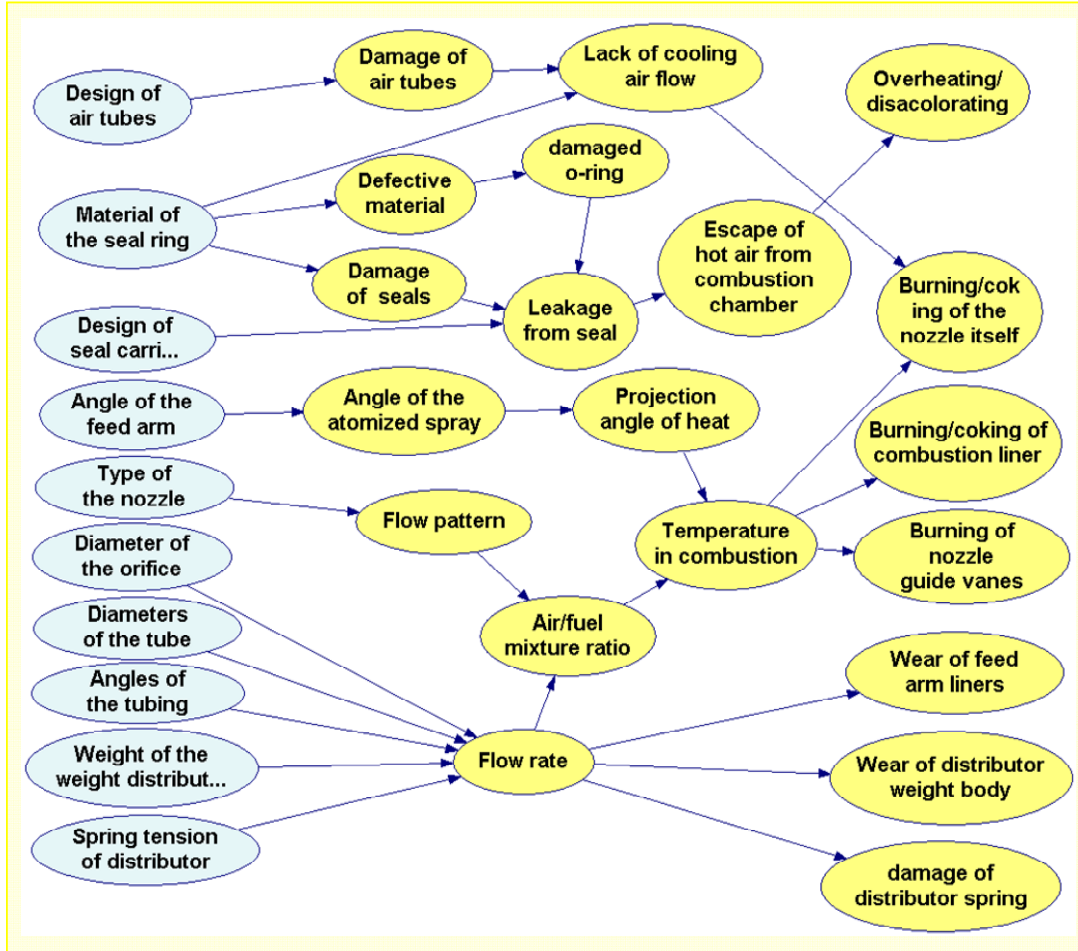


Figure 4-6 Network from BBN

For test these networks, ten scenarios are used. One example scenario is that temperature in combustion is 'Yes' and leakage form seal is 'Yes'. The meaning of this particular scenario is that the maintenance department observed that the temperature in combustion is not normal and the leakage from seal is occurred in the fuel nozzle. To compare the result of the scenario with both networks, the testing diagnosis module, which is one of GeNIe modules, and two criteria are used in this work. One is the number of matching in the top five ranked

targets (N), which is shown left and upper side of figures. The other is the number of matching order (O) of the matching ranked targets (identified in N). In O, a flipped order is considered as one matching count in a consecutive order. These two criteria are combined as a comparison measure (R) as a weighted sum as shown below:

$$R = C_1 \times W_1 + C_2 \times W_2, \quad C_1 = N / 5, \text{ and } C_2 = O / N,$$

where C_1 and C_2 are scaled measures for N and O, and W is a weight for each criteria.

The sum of W_1 and W_2 equals to 1. The R (Table 5) with this example case was 0.87, which means that the both networks performed 87% similar each other, where average N is 3.8 and O is 3.7. It concludes that the method, FCM-BBN, shows similar performance with the BBN in this fuel nozzle case. In the experiments, the probabilities of FCM-BBN were greater than ones in BBN. It seems that this difference is due to the normalization process.

Table 4-5 The Test Results with Fuel Nozzle

No.	Scenarios	N	O	C1	C2	R
1	Temperature in combustion	5	4	1	0.8	0.9
	Leakage from seal					
2	Overheating/discolorating	3	3	0.6	1	0.8
	Burning/coking of the nozzle itself					
3	Burning of nozzle guide vanes	5	5	1	1	1
	Air/fuel mixture ratio					
4	damaged o-ring	3	3	0.6	1	0.8
	damage of distributor spring					
5	Damage of seals	4	4	0.8	1	0.9
	Air/fuel mixture ratio					
6	Flow pattern	3	3	0.6	1	0.8
	Temperature in combustion					
	Flow rate					
7	Lack of cooling air flow	4	4	0.8	1	0.9
	Wear of distributor weight body					
	Defective material					
8	Leakage from seal	3	3	0.6	1	0.8
	Temperature in combustion					
	Escape of hot air from combustion chamber					
9	Burning of nozzle guide vanes	4	4	0.8	1	0.9
	Projection angle of heat					
	Defective material					
10	Projection angle of heat	4	4	0.8	1	0.9
	Wear of distributor weight body					
	damaged o-ring					
	Burning/coking of the nozzle itself					
	Average	3.8	3.7	0.76	0.98	0.87

4.5 Conclusion

This chapter presents three topics. First, a methodology for the systematic construction of FCM is presented. Our methodology is general and independent of

any specific application. It covers the entire process of constructing FCM. The methodology is systematically and formally described to avoid potential ambiguities. Second, this research have compared the roles of FCM and BBN in the knowledge engineering of causal reasoning systems. The comparison is done based on some inherent features of the frameworks which are independent of any specific applications. Third, a methodology of FCM-BBN conversion is presented. BBNs are used for the representation and reasoning of the assembly design decision and environmental factors. Also, FCM is used for the indirect knowledge acquisition, and the causal knowledge in FCM is converted to BBN. In case study, we compared the networks' accuracy between BBNs directly generated by domain experts and generated from FCM, with a realistic industrial example, a fuel nozzle for an aerospace engine. The result of comparison concludes that FCM-BBN is similar performance with BBN in this fuel nozzle case.

In this chapter, only one case study for fuel nozzle is tested. Other future work planned is to get a group of knowledge engineers and domain experts working on a number of real applications, with different nature, using FCM and BBN separately. A comparison of the frameworks can be done based on the statistics of the subjective opinion from the knowledge engineers and the domain experts. Also, this research is conducting more comprehensive testing for the FCM-BBN method and the result will be reported in a separate article. An automatic FCM-BBN generating system will be implemented to reduce the manual construction of BBN, which requires time-consuming processes.

CHAPTER 5

PRODUCT DESIGN KNOWLEDGE REPRESENTATION AND TRANSFORMATION

The aim of this chapter is to conduct a mathematical representation and comparison of procedural knowledge and causal knowledge from the perspective of representing product development knowledge. Product development knowledge is seldom documented since in typical product development processes, the knowledge evaporates or erodes after the product design is completed. Product development knowledge is exponentially exploding because the Information Technology (IT) can be providing various data/information/knowledge from various sources, such as Internet, books, other domain experts, communities, and more. Nowadays, to realize a truly collaborative product development environment, therefore, product development knowledge should be managed, which means it is properly captured, represented, stored, and reused. The first topic is knowledge capture or elicitation from domain experts. This issue is presented in preliminary study in chapter 4. In this chapter, the representation of product development knowledge, which is one of the significant functions in the product development knowledge management, is addressed. This chapter discusses the comparison of two knowledge representations (i.e., procedural knowledge representation and causal knowledge representation) for properly using the representation of product development knowledge. Also, it discusses how procedural knowledge can be transformed as causal knowledge, which represents the relationship between

cause and effect.

5.1 Definitions and Relationships in Knowledge

Most product development knowledge is represented by procedural knowledge, since the procedural knowledge includes both declarative and contextual knowledge. Product design knowledge can be represented by procedural knowledge. However, procedural knowledge is broad and requires unwieldy processes to define itself discretely. Furthermore, during product development processes, procedure knowledge cannot fully represent product design knowledge [Kim 2008]. Because procedural knowledge is static, and cumbersome processes are needed to define the procedural knowledge individually. Causal knowledge, which utilizes causal reasoning, is particularly useful for overcoming these challenges. By modeling causal relationships, causes of certain events are diagnosed and their effects are predicted [Gopnik 2002, Gopnik 2004, Liu 2001]. The causal network (e.g., Bayesian belief network) has a sound mathematical foundation and reasoning capabilities; it also has an efficient evidence propagation mechanism and a proven track record in industry-scale applications.

The probabilistic causal network (e.g., Bayesian belief network) represents causal relationship, which is quantified by the computation of the probabilities of any subset of variables given evidence about any other subset. These relationships, which are updated by probabilistic beliefs, represent informational or causal dependencies in the causal network. To utilize causal knowledge (CK), first,

CK must be defined. One of the possible methods is set theory, which is the branch of mathematics that studies collections of objects. Although any type of object can be collected into a set, set theory is applied most often to objects that are relevant to mathematics. The following are the definitions of CK and procedural knowledge (PK) models by set theory. In this paper, we follow a naïve set theory by Halmos [Halmos 1960]. As described in the previous chapter, CK has the ability to explain why a particular conclusion is made; via causal reasoning, the causality can be diagnosed and their effects can be predicted even under incomplete situation.

Definition 5-1 illustrates CK, which is a network that is composed with vertices, edges, and probability of each vertex. Vertices include input, intermediate, and output vertices. Input vertices represent a set of input objects (number of inputs) and a set of conditions (CoK). Intermediate vertices are a set of knowledge system, for example, joining. Output vertices are a set of methods for the knowledge system (e.g., welding, riveting, fastening, adhesive bonding). Edges represent connections between vertices. If vertex v_1 is connected with vertex v_2 , e_{12} is 1. Otherwise, e_{12} is 0. The probability of each vertex represents its causal effects.

Definition 5-1 Causal Knowledge

Causal knowledge is a network that is composed with vertex, edge, and probability of each vertex.

CK : Causal knowledge

$CK = \{V, E, Pa\}$,

n is the total number of vertices in CK

V is a set of all vertices in CK .

$V = \{V^{in}, V^{int}, V^{out}\}$, where V has input, intermediate, and output vertices.

$V^{in} = \{i_m, c_{hg}\}$, $V^{int} = \{s_o\}$, $V^{out} = \{m_e\}$,

$I = \{i_m; m = 1, \dots, n_m\}$

$C = \{c_{hg}; h = 1, \dots, n_h, g = 1, \dots, n_g\}$,

$M = \{m_e; e = 1, \dots, n_e\}$

$S = \{s_o; o = 1, \dots, n_o\}$,

where I is a set of input objects, C is a set of conditions, M is a set of methods, and S is a set of names of knowledge system

E is a set of connected edges in CK .

$E = \{e_{jk}; j, k = 1, \dots, n, j \neq k\}$, where $e_{jk} = 1$ if the edge jk is existed, otherwise, $e_{jk} = 0$.

Pa is a set of probabilities of nodes in CK .

$Pa = \{Pa_i; i = 1, \dots, n_i\}$

Definition 5-2 illustrates PK, which is knowledge with specific pre-defined conditions. PK includes DK, CoK, and knowledge system. As defined in Chapter 2.1, DK is knowledge of facts or is knowledge that answers the question of “what”. CoK addresses “when” and “why” to use the DK [Yoo 2006]. Knowledge system is a pre-defined system for a specific knowledge model (e.g., welding system). All three components of PK are pre-defined. It means that this knowledge is *static*.

Definition 5-2 Procedural Knowledge

In the procedural knowledge model, a knowledge system represents procedural knowledge with specific pre-defined conditions

PK : Procedural Knowledge

PK = {*DK*, *CoK*, *KS*}, where *DK* is declarative knowledge; *CoK* is contextual knowledge; *KS* is knowledge system.

DK = {*I*, *M*} = {Input object, Method, Output}, *DK* is pre-defined.

CoK = {*C*} = {Conditions}, *CoK* is pre-defined.

KS = {*S*} = {Names of the knowledge models}, *KS* is pre-defined.

$I = \{i_m; m = 1, \dots, n_m\}$,

$C = \{c_{hg}; h = 1, \dots, n_h, g = 1, \dots, n_g\}$,

$M = \{m_e; e = 1, \dots, n_e\}$

$S = \{s_o; o = 1, \dots, n_o\}$,

where *I* is a set of input objects, *C* is a set of conditions, *M* is a set of methods, and *S* is a set of names of knowledge system

The relationship between PK and CK is addressed with simple product development knowledge (definitions 5-1 and 5-2). The PK includes two input objects (A, B), one method (welding \textcircled{W}), two conditions (when, why), and output object (A \textcircled{W} B) for this specific two-object-welding knowledge. This knowledge means two input objects are welded to get an output object when the conditions are occurred. This PK does include CoK, which is a condition of the method. However, the conditions are predefined, which means this knowledge is *static* for

the specific conditions.

Most product development knowledge can be represented by PK. However, that knowledge is rarely represented by CK in product development. To use CK requires a knowledge transformation from PK to CK. Definition 5-3 represents knowledge transformation from PK to CK (Figure 5-1). PK's DK, CoK, and KS are transformed to vertices in CK ($DK: \rightarrow V^{in}, V^{out}$, $COK: \rightarrow V^{in}$, $KS: \rightarrow V^{int}$). However, there is a limitation to obtain Pa from PK because a single PK cannot represent the probability of event. If the cases of the same PK are existed, Pa can be calculated.

Definition 5-3 Knowledge transformation

Knowledge transformation is a transformation process from procedural knowledge to causal knowledge.

When PK is transformed to CK ,

$PK: \rightarrow CK$,

$DK: \rightarrow V^{in}, V^{out}$,

$COK: \rightarrow V^{in}$,

$KS: \rightarrow V^{int}$

$E = \{ V^{in} \rightarrow V^{int}, V^{int} \rightarrow V^{out} \}$, where $V^{in} \rightarrow V^{int} = 1$, $V^{int} \rightarrow V^{out} = 1$ if the edge $V^{in} \rightarrow V^{int}$, $V^{int} \rightarrow V^{out}$ is existed, otherwise, $V^{in} \rightarrow V^{int} = 0$, $V^{int} \rightarrow V^{out} = 0$.

Therefore, $CK = \{DK, CoK, KS, E, Pa\}$, but Pa not defined by a single PK .

CK represents a necessary relationship between one event and another event (cause to effect). The PK is transformed to a CK. The CK has three input nodes -

one join node, and two output method nodes. Each has more than two stages with probability, such as two cases of input object (e.g., two objects with 0.6, three objects with 0.4) for input object node. This CK can represent exactly the same knowledge with PK if number of object is two (A, B) and the condition of method (when, why) is defined, the method 'welding' is occurring and can represent more knowledge (e.g., 'riveting,' 'fastening,' 'adhesive bonding' for the methods with different inputs).

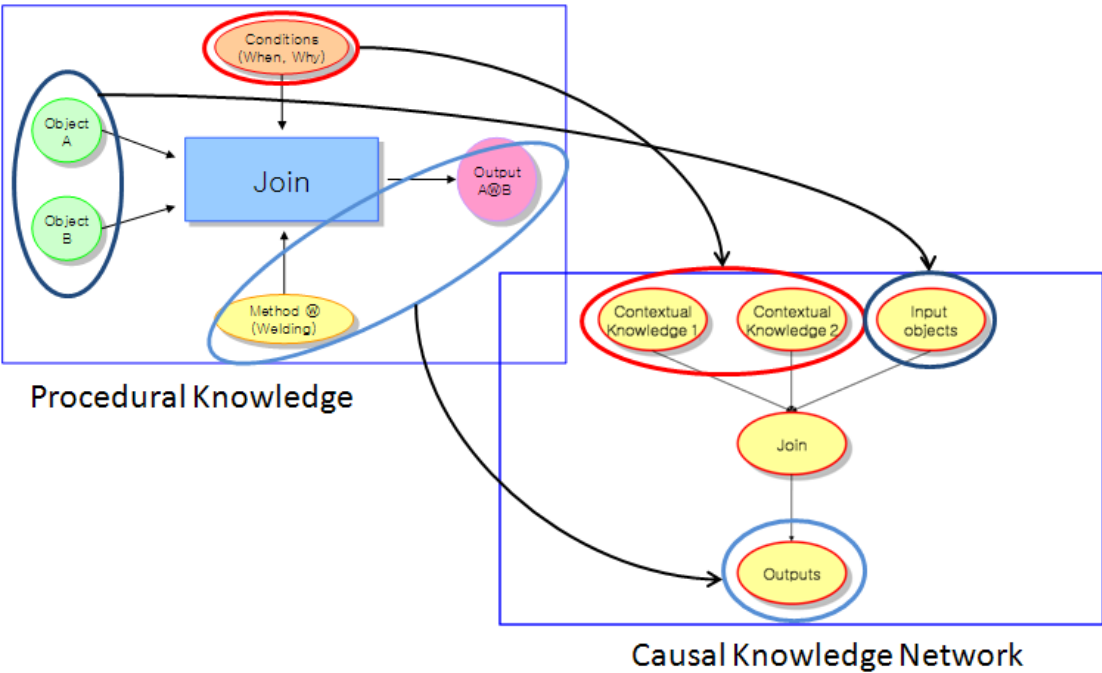


Figure 5-1 Knowledge transformation

5.2 Mathematical Comparison of PK and CK

In the previous chapter, CK and PK using a set theory and the knowledge transformation method from PK to CK, is addressed. In this chapter, to show that CK is superior to PK, I compare CK and PK from four perspectives: knowledge

expression ability, decision alternative representation ability, reasoning capability, and knowledge cultivation ability. To explain the different characteristics of two knowledge methods, I use a two-object-welding knowledge example (Figure 5-2). In this example, all information for the PK is already given. The number of input objects must be *two*. The CoK (when) is *the objects are weldable* and CoK (why) is *the objects have to be firmly joined*. The method of this joining knowledge is *welding*. PK represented by Definition 5-2 is following.

$DK_1 = \{i_1, i_2\} = \{\text{object 1, object 2}\},$
 $CoK = \{c_1, c_2\} = \{\text{'the objects are weldable', 'the objects have to be firmly joined'}\},$
 $DK_2 = \{m_1\} = \{\text{welding}\},$
 $KS = \{s_1\} = \{\text{the name of the knowledge system}\} = \{\text{join}\}.$

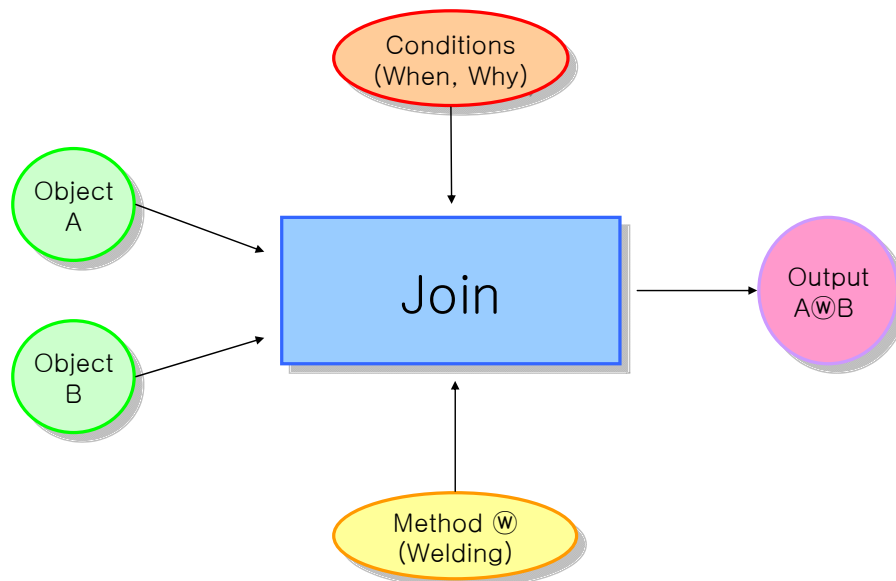


Figure 5-2 Procedural knowledge representation for two-object-welding knowledge

For CK (Figure 5-3), *join* knowledge is used. The input object number is determined by the probability of each input instance, such as two cases of input object (e.g., two objects with 0.6, three objects with 0.4). The CoK (conditions of join) is defined by the combination of cases from four instances where CoK (when) is *'the objects are weldable or the objects have holes,'* CoK (why) is *'the objects have to be firmly joined or the objects are joined'*. Also, the output (method) is determined by all inputs. It can be one of four methods: welding, riveting, fastening, adhesive bonding. CK as represented by Definition 5-1 as shown below:

$$\begin{aligned}
 V &= \{V^{in}, V^{int}, V^{out}\}, \\
 V^{in} &= \{i_m, c_{hg}\}, \\
 V^{int} &= \{s_g\}, \\
 V^{out} &= \{m_e\} \\
 E &= \{e_{pq}; p = 1, \dots, n_p, q = 1, \dots, n_q\}, \\
 Pa &= \{pa_h; h = 1, \dots, n_h\}, \\
 I &= \{i_m; m = 1, \dots, n_m\} = \{i_1, i_2\} = \{\text{object 1, object 2}\}, \\
 C &= \{c_{hg}; h = 1, \dots, n_h, g = 1, \dots, n_g\} = \{c_{11}, c_{12}, c_{21}, c_{22}\} = \{\text{'the objects are weldable'}, \text{'the objects have holes'}, \text{'the objects have to be firmly joined'}, \text{'the objects are joined'}\} \\
 M &= \{m_e; e = 1, \dots, n_e\} = \{m_1, m_2, m_3, m_4\} = \{\text{'welding'}, \text{'riveting'}, \text{'fastening'}, \text{'adhesive bonding'}\} \\
 S &= \{s_g; g = 1, \dots, n_g\} = \{s_1\} = \{\text{'join'}\}
 \end{aligned}$$

where I is a set of input objects, C is a set of conditions, M is a set of methods, and S is a set of names of knowledge model; $|I_K|$, $|C_K|$, $|M_K|$, $|S_K|$ are a number of cases for each set.

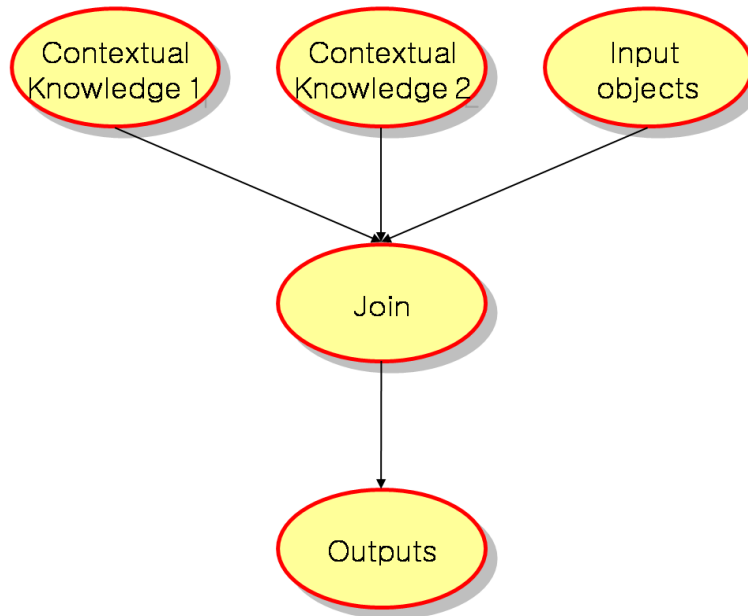


Figure 5-3 Causal knowledge representation for joining knowledge

To compare PK and CK, we need appropriate comparison perspectives. The perspectives should define the core characteristics of knowledge such as knowledge expression, decision alternatives suggestion, reasoning capability, and dynamic information processing. Therefore, in this study four perspectives are defined: knowledge expression ability; decision alternative representation ability; reasoning capability; and knowledge cultivation ability. Knowledge expression ability means how much knowledge can be represented by a single knowledge model. The decision alternative representation ability measures how many decision alternatives can be included and can be used for decision support in a single knowledge model. Reasoning capability, which includes prediction and diagnosis, is the ability to extract hidden knowledge, new knowledge, and rules from any given knowledge based on mathematical theory. Knowledge cultivation ability

represents how much additional knowledge can be obtained by the insertion of information instance.

The first perspective, knowledge expression ability, is defined in Definition 5-4. Knowledge expression ability is about the generality of knowledge expression; a single knowledge model is able to represent how much knowledge. This ability is measured with η . η is a combined measurement of knowledge cases and is calculated by case multiplication with cases in input, contextual, and method nodes. In the two-object-welding example, for PK, two pre-defined input objects, two conditions, and one method *welding* provides only one case for each set: $|I_{PK}| = 1$, $|C_{PK}| = 1$, $|M_{PK}| = 1$. Then, η_{PK} is $|I_{PK}| \cdot |C_{PK}| \cdot |M_{PK}| = 1 \times 1 \times 1 = 1$. However, CK is more general than PK. Three input nodes can have multiple instances; the number of the input objects can be two and more; two of contextual knowledge also can be multiple. All instances of nodes have probability for the specific cases, and based on these input nodes with probability, the output (method) will be determined. For example, the *welding* output is determined with the same input information with PK. If the input information for CK is changed from *the objects are weldable* to *the objects have holes*, the output will be riveting or fastening. For this comparison, let's assume the number of input objects is two. Then, the conditions of the CK are either *when = weldable or objects have hole* or *why = have to be firmly joined or have to be joined*. Since CK methods are *welding, riveting, fastening, and adhesive bonding* with probabilities, $|I_{CK}| = 1$, $|C_{CK}| = 4$, $|M_{CK}| = 4$, and $\eta_{CK} = |I_{CK}| \cdot |C_{CK}| \cdot |M_{CK}| = 1 \times 4 \times 4 = 16$. A single knowledge model of CK can represent 16 different

knowledge.

Definition 5-4 Knowledge expression ability

Knowledge expression ability means how much knowledge can be represented by a single knowledge model. η is a measurement of knowledge expression and is written in the number of represented knowledge in a single knowledge model.

$$\begin{aligned}\eta_K &= \text{total number of expression with knowledge } K \\ &= |I_K| \cdot |C_K| \cdot |M_K|,\end{aligned}$$

where \cdot is case multiplication symbol; $|I_K|$, $|C_K|$, $|M_K|$ are a number of cases for each set.

Decision alternative representation ability is the next perspective (Definition 5-5). This ability represents how many decision alternatives are provided for decision support from a single knowledge model. ρ is represented by the total number of the order of decision alternatives (O_{MK}). O_{MK} is the order of decision alternative (M_K). M_K is determined by a probability based on input nodes (I_k , C_k). PK is only for a specific task and is limited in the task. PK can provide only welding with two objects and CoKs (when and why). For the PK, $|M_{PK}| = 1$, and $\rho_{PK} = O_{MPK} = 1$ because it has only one knowledge, which means it can only support two object welding tasks. However, CK can have multiple alternatives for any given input information and it can provide ranked alternatives. Two input objects, the combination of CoKs (when and why) cause all CK methods, which include welding,

riveting, fastening, and adhesive bonding with probabilistic order (e.g., Welding (45%), Riveting (25%), Fastening (20%), and Adhesive bonding (10%)). Therefore, $|M_{CK}| = 4$, and $\rho_{CK} = O_{MCK} = 4$. Comparing between PK and CK, CK has a more powerful decision supporting function than PK.

Definition 5-5 Decision alternative representation ability

A decision alternative representation ability measures how many decision alternatives can be included and be used for decision support in a single knowledge model. ρ is represented by the total number of the order of decision alternatives (O_{MK}).

$$\rho_K = O_{MK},$$

where $P(M_K) \rightarrow O_{MK}$, in other words O_{MK} is determined by $P(M_K)$; $P(M_K)$ is a probability based on input nodes (I_K, C_K); O_{MK} is the order of M_K .

Definition 5-6 Reasoning capability

Reasoning capability, which includes induction, deduction, and abduction, is the ability to extract hidden knowledge, new knowledge, or a rule from any given knowledge based on mathematical theory. i is a measurement of reasoning capability, which is represented with a number of inferred knowledge from existing knowledge.

$$I_K = I_K^i + I_K^d + I_K^a,$$

Induction

$$i_K = |I_K, C_K \rightarrow M_K| = |I_K| \cdot |C_K|,$$

Deduction

$$i_K^d = 2 \times (|I_K| \cdot |C_K| \cdot |M_K|),$$

Abduction

$$i_K^a = |I_K| \cdot |C_K| \cdot |M_K|$$

where \cdot is a case multiplication symbol; $|I_K|$, $|C_K|$, $|M_K|$ are a number of cases for each set.

The next perspective is reasoning capability. Definition 5-6 illustrates the causation of knowledge with induction, deduction, and abduction to extract a hidden knowledge, a new knowledge, and a rule from any given knowledge based on mathematical theory. i is a measurement of reasoning capability, which is represented with a number of inferred knowledge from existing knowledge. Based on the Free On-line Dictionary of Computing (FOLDOC), these induction, deduction, abduction knowledge causation are defined as: 1) induction is a type of reasoning which involves moving from a set of specific facts to a general conclusion; 2) deduction is a type of reasoning which constructs or evaluates deductive arguments; 3) abduction is a method of logical inference which comes prior to induction and deduction. Induction is the process of inferring probable antecedents as a result of observing multiple consequents. For example, the statement "It is snowing outside" is invalid until one looks or goes outside to see whether it is true or not. Induction requires sense experience. Deduction is the process of deriving the consequences of what is assumed. Given the truth of the

assumptions, a valid deduction guarantees the truth of the conclusion. For example, if it is true (given) that the sum of the angles is 180° in all triangles, and if a certain triangle has angles of 90° and 30° , then it can be deduced that the third angle is 60° . Abduction allows inferring a precondition as an explanation of a consequence. Because of this, abduction allows the precondition to be inferred from the consequence—for example, “The window’s getting wet”; then, it may be raining outside.

PK cannot support any reasoning unless only for a specific task. For PK, $|I_{PK}| = 1$, $|C_{PK}| = 1$, $|M_{PK}| = 1$. Induction (I^i_{PK}) is $|I_{PK}| \cdot |C_{PK}| = 1 \times 1 = 1$, deduction (I^d_{PK}) is $2 \times (|I_{PK}| \cdot |C_{PK}| \cdot |M_{PK}|) = 2 \times (1 \times 1 \times 1) = 2$, and abduction (I^a_{PK}) is $|I_{PK}| \cdot |C_{PK}| \cdot |M_{PK}| = 1 \times 1 \times 1 = 1$. Therefore, $I_{PK} = I^i_{PK} + I^d_{PK} + I^a_{PK} = 1 + 2 + 1 = 4$. However, CK has a powerful reasoning capacity based on mathematical theory. With reasoning capability, CK can reason the causation of knowledge (i.e., induction, deduction, and abduction). For example, if welding is selected, the node constraint for when becomes the most effective node. The input objects are the second effective nodes since welding needs more than two objects. Also, with reasoning capability, CK can handle incomplete information. As an example, say the number of objects is 2 and the CoK (why) is *objects have to be joined firmly*. CoK (when) is missing. In this case, CK can provide best alternatives to decision, such as welding, fastening, adhesive bonding. For CK, $|I_{CK}| = 2$, $|C_{CK}| = 4$, $|M_{CK}| = 4$. Induction (I^i_{CK}) is $|I_{CK}| \cdot |C_{CK}| = 2 \times 4 = 8$, deduction (I^d_{CK}) is $2 \times (|I_{CK}| \cdot |C_{CK}| \cdot |M_{CK}|) = 2 \times (2 \times 4 \times 4) = 64$, and abduction (I^a_{CK}) is $|I_{CK}| \cdot |C_{CK}| \cdot |M_{CK}| = 2 \times 4 \times 4 = 32$. Therefore, $I_{CK} = I^i_{CK} + I^d_{CK} + I^a_{CK}$

$+ I_{CK}^a = 8 + 64 + 32 = 104$. So, CK has more powerful reasoning capability than PK has.

Definition 5-7 Knowledge cultivation ability

Knowledge cultivation ability represents how much additional knowledge can be obtained by the insertion of information instance. ψ is the total number of cultivated information and is increased knowledge by the insertion of information instance ($\Delta I_K, \Delta C_K, \Delta M_K$)

$$\psi_K = |I_K + \Delta I_K| \cdot |C_K + \Delta C_K| \cdot |M_K + \Delta M_K| - \eta_K,$$

where \cdot is a case multiplication symbol; $|I_K|, |C_K|, |M_K|$ are a number of cases for each set; $\Delta I_K, \Delta C_K, \Delta M_K$ are increased information instance in $I_K, C_K,$ and M_K .

The last perspective is knowledge cultivation ability (Definition 5-7). Knowledge cultivation means knowledge extension, representing how much additional knowledge can be obtained by the insertion of an information instance. ψ is the total number of cultivated information and is increased knowledge by the insertion of information instance ($\Delta I_K, \Delta C_K, \Delta M_K$). ψ is calculated by the total number of cultivated and existing information, minus the number of the existing information (η_{PK}), which is from knowledge expression ability. PK is static. All information is given, already defined for a specific task. For the PK, $|I_{PK} + \Delta I_{PK}| = 1$, $|C_{PK} + \Delta C_{PK}| = 1$, $|M_{PK} + \Delta M_{PK}| = 1$, $\eta_{PK} = 1$. $\psi_{PK} = |I_{PK} + \Delta I_{PK}| \cdot |C_{PK} + \Delta C_{PK}| \cdot |M_{PK} + \Delta M_{PK}| - \eta_{PK} = 1 \times 1 \times 1 - 1 = 0$. This means that PK does not support any

knowledge extension, because it has static. However, CK can handle dynamic information; the number of objects can be more than two. CoKs (when and why) and outputs also can be multiple. It is easy to extend the instance of each node. For the CK, the number of input objects, conditions, and methods are multiples—2 and more. In this case, assume the number of input objects is 2 or 3 plus 4, which is dynamically increased. The conditions of the CK are CoK (when) is *the objects are weldable or the objects have holes* and CoK (why) is *the objects have to be firmly joined or the objects are joined*. The method of the CK is *welding, riveting, fastening, and adhesive bonding*. $|I_{PK} + \Delta I_{PK}| = 1 + 2$, $|C_{PK} + \Delta C_{PK}| = 4 + 0$, $|M_{PK} + \Delta M_{PK}| = 4 + 0$, $\eta_{PK} = 16$. $\psi_{PK} = |I_{PK} + \Delta I_{PK}| \cdot |C_{PK} + \Delta C_{PK}| \cdot |M_{PK} + \Delta M_{PK}| - \eta_{PK} = 3 \times 4 \times 4 - 16 = 32$. This means CK can represent 32 more knowledge cases only adding two more input object cases. Therefore, CK is more capable to handle dynamic information.

In summary, after comparison between PK and CK with four perspectives, CK is superior to PK in terms of knowledge expression, reasoning, decision alternative representation, knowledge cultivation ability. In addition, CK has sound mathematical theorem and knowledge integration by structure and belief integration. Table 5-1 shows more comparison result.

Table 5-1 Comparison result between procedural knowledge and causal knowledge

Perspective	Criterion	Procedural Knowledge	Causal Knowledge	
Knowledge expression ability	Generality	Specific with conditions	General for similar cases	
Reasoning ability	Reasoning	No reasoning	Possible	
	Incompleteness for input data	Not supported	Supported	
	Causation	Induction	One knowledge	Inductable
		Deduction	One knowledge	Deductable
		Abduction	One knowledge	Abductable
Decision alternative representation ability	Decision Supporting	One knowledge	Support alternatives	
Knowledge cultivation ability	Dynamic information processing	Static information	Dynamic information	
	Extension	Limited	Unlimited	
Others	Mathematical representation	Symbolic	Sound mathematical theorems	
	Integration	Not supported	Structural and belief integration	

5.3 Implementation: Knowledge Modeling with SysML

This chapter shows knowledge modeling with Systems Modeling Language (SysML), including an explanation of SysML followed by knowledge modeling implementation. SysML is a general-purpose modeling language for systems engineering applications (<http://www.wikipedia.com> and http://www.omg.sysml.org/#What-Is_SysML). It supports the specification, analysis, design, verification, and validation of a broad range of systems and systems-of-systems. SysML was developed by an open source specification project and is defined as an extension of the subset of Unified Modeling Language (UML) using UML's profile mechanism.

There are three advantages to use SysML as follows: 1) SysML's semantics are more flexible and expressive than UML. SysML reduces UML's software-centric restrictions and adds two new diagram types (i.e., requirement and parametric diagrams) to model hardware, software, information, processes, personnel, and facilities; 2) SysML is a smaller language since it removes many of UML's software-centric constructs. SysML has a total of nine diagram type, which includes reuses seven of UML's thirteen diagrams and adds two diagrams (requirements and parametric diagrams); 3) the SysML model management constructs support models, views, and viewpoints. These constructs extend UML's capabilities and are architecturally aligned with IEEE-Std-1471-2000 (IEEE's Recommended Practice for Architectural Description of Software Intensive Systems). Figure 5-4 illustrates the four pillars of SysML. Block is the basic unit of structure in SysML and can be

used to represent hardware, software, facility, personnel, or any other system element. The system structure is represented by block definition diagrams and internal block diagrams. The behavior diagrams include use case diagram, activity diagram, sequence diagram, and state machine diagram. The activity diagram represents the flow of data and control between activities. A sequence diagram represents the interaction between collaborating parts of a system. The state machine diagram describes the state transitions and actions that a system or its parts perform in response to events. The requirements diagram captures requirements hierarchies and requirements derivation, and the “satisfy and verify” relationships allow a modeler to relate a requirement to a model element that satisfies or verifies the requirements. The parametric diagram represents constraints on system property values such as performance, reliability, and mass properties, and serves as a means to integrate the specifications and design models with engineering analysis models.

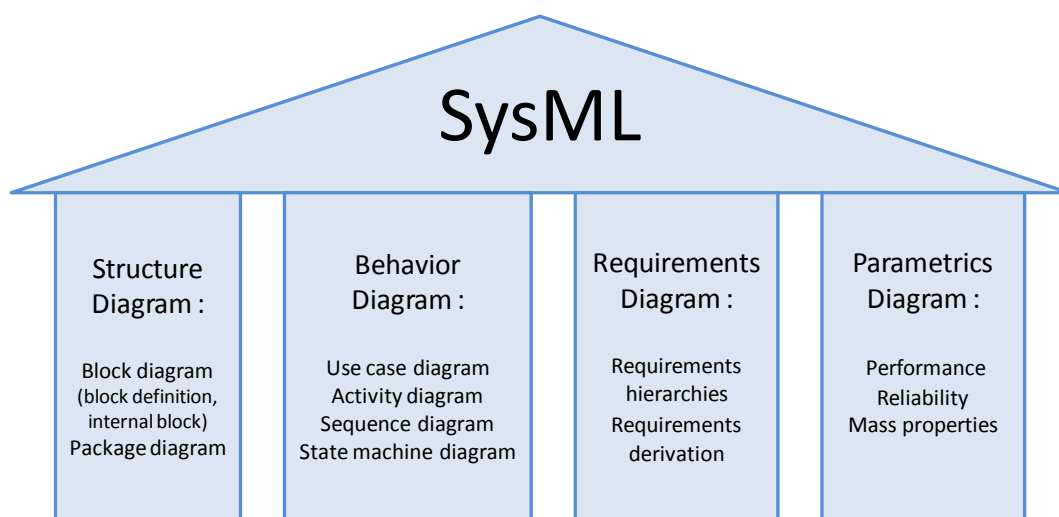


Figure 5-4 The Four Pillars of SysML

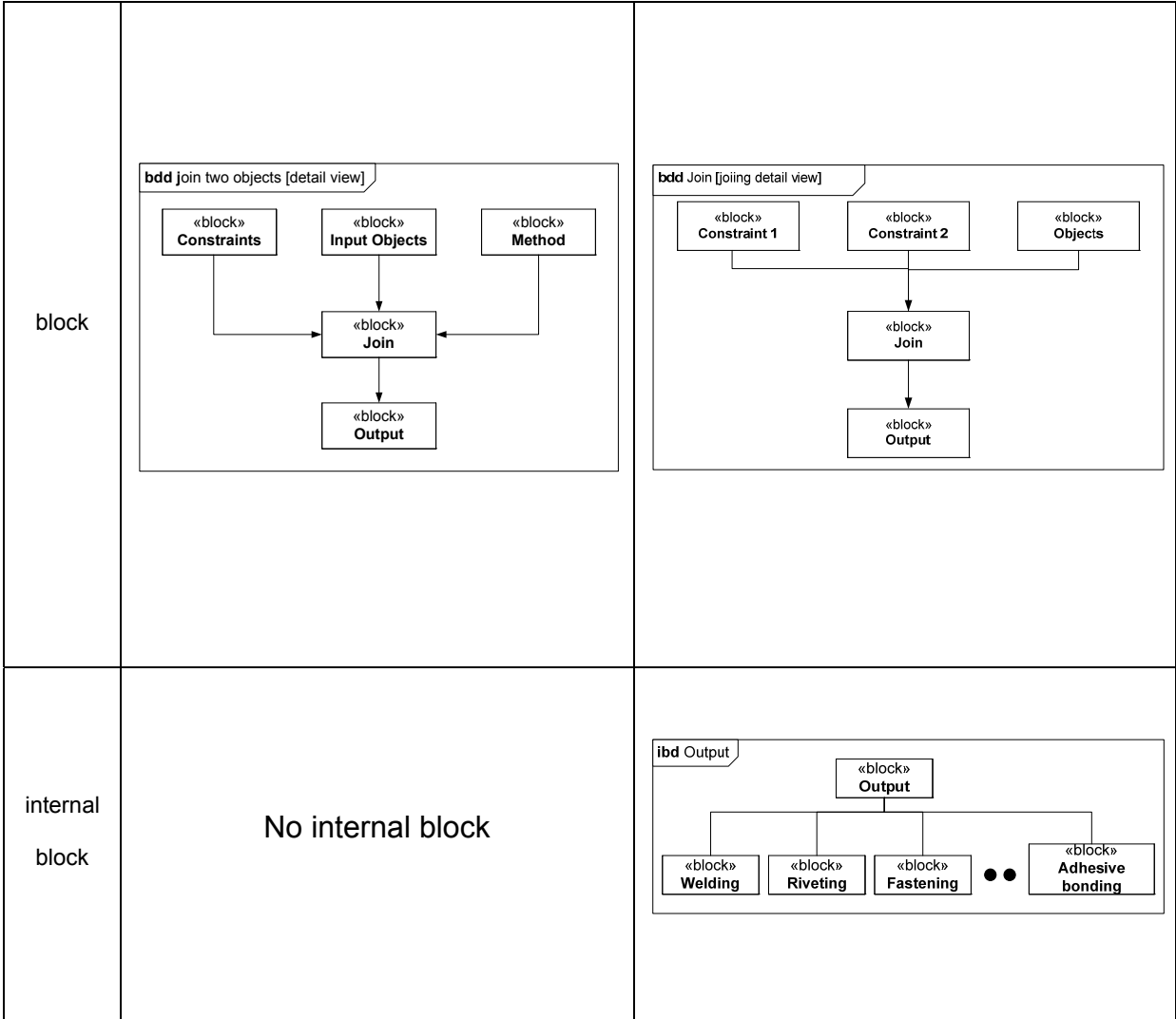
Using the two-object-welding examples in Chapter 5-2, the knowledge modeling for PK and CK is implemented with SysML. Each knowledge model is composed with package, block, internal block, activity, sequence, state machine, and requirements diagrams. Table 5-2 includes the detail implementations of SysML for PK and CK. Comparing these detail implementations of both knowledge models we note: 1) the CK and PK package diagrams are similar; however, PK's inputs, conditions, and method are pre-defined, 2) CK's block diagram is similar to PK, 3) only an internal block diagram of CK exists, because CK's output can be decomposed to sub-block. This means that CK can support multiple alternatives for the method. 4) An activity diagram of CK can represent probabilities for the outputs, which means that CK is able to infer knowledge, 5) sequence diagrams are identical with both knowledge, 6) state machine diagram of CK can represent dynamic information processing ability because the information of the CK is handled by information manager, finder, or recommender, which are linked with dynamic information blocks, 7) requirement diagram of CK has more requirements than PK has because CK can represent more knowledge.

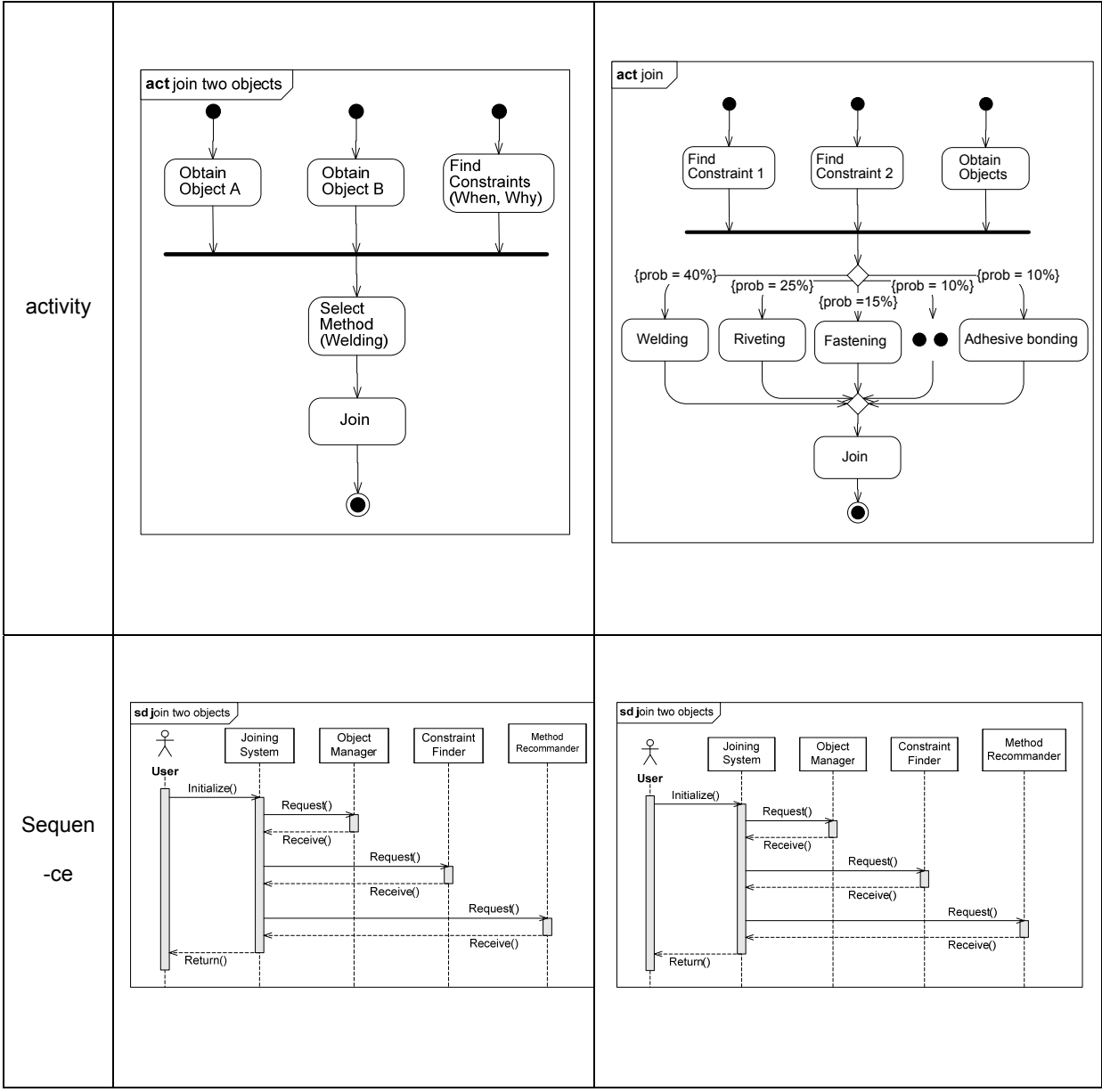
In summary, this SysML analysis indicates that PK's knowledge model is inferior to CK's knowledge model because CK can represent more knowledge than PK as shown in the state-machine and requirement diagrams. PK's knowledge is static, which means all information is predefined. CK's block diagram includes more detailed knowledge with internal block diagrams. Also, more requirements are necessary to represent CK than PK. The state machine diagram for CK can

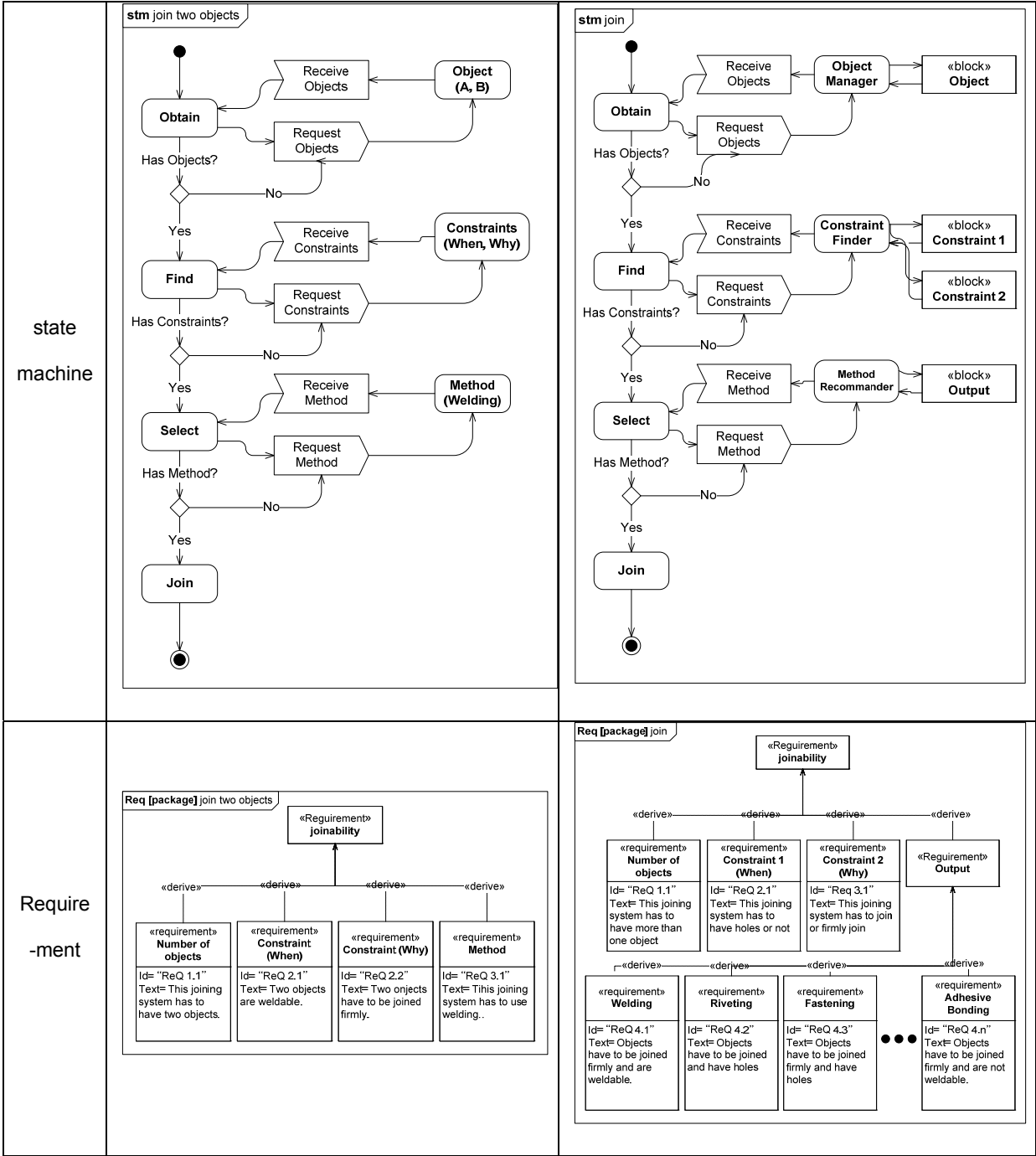
handle dynamical increased/decreased information. Therefore, we can reconfirm the result with Table 1 in Chapter 5-4.

Table 5-2 SysML implementation for PK and CK

SysML Diagram	PK	CK
package	<p>bdd [package] join two objects</p> <p>«block» {encapsulated} Join</p> <hr/> <p>Constraints When, Why</p> <hr/> <p>Input Object A, Object B</p> <hr/> <p>Method Welding</p> <hr/> <p>Output Weldment (A⊕B)</p>	<p>bdd [package] join</p> <p>«block» {encapsulated} Join</p> <hr/> <p>Inputs Constraint 1, Constraint 2, Objects</p> <hr/> <p>Join Mapping</p> <hr/> <p>Output Welding , Riveting, Fastening, Adhesive bonding, etc.</p>







5.4 Demonstration: Representation and Reasoning Capability of Causal Knowledge

This chapter includes a demonstration of CK to represent product development knowledge, particularly decision support and reasoning capability. This demonstration is based on a fuel nozzle of the aerospace jet engine. The network is created based on the domain expert for the CK representation of the fuel nozzle as shown in Figure 5-5. This network shows the design stages, which are ten nodes located in left side in the networks; the others are the maintenance stages. GeNIe (Graphical Network Interface) is used to represent and test the performance of network.

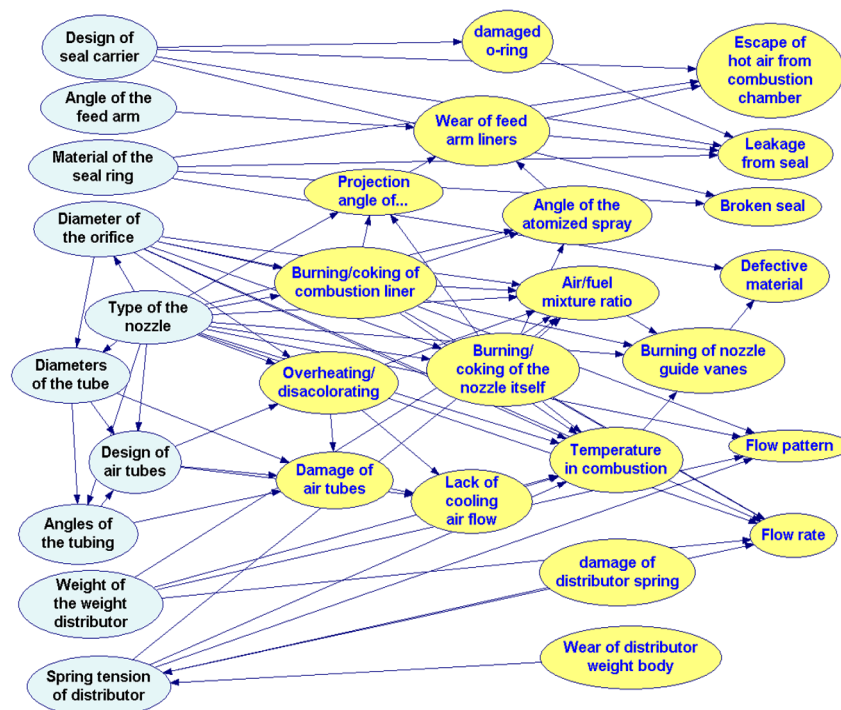
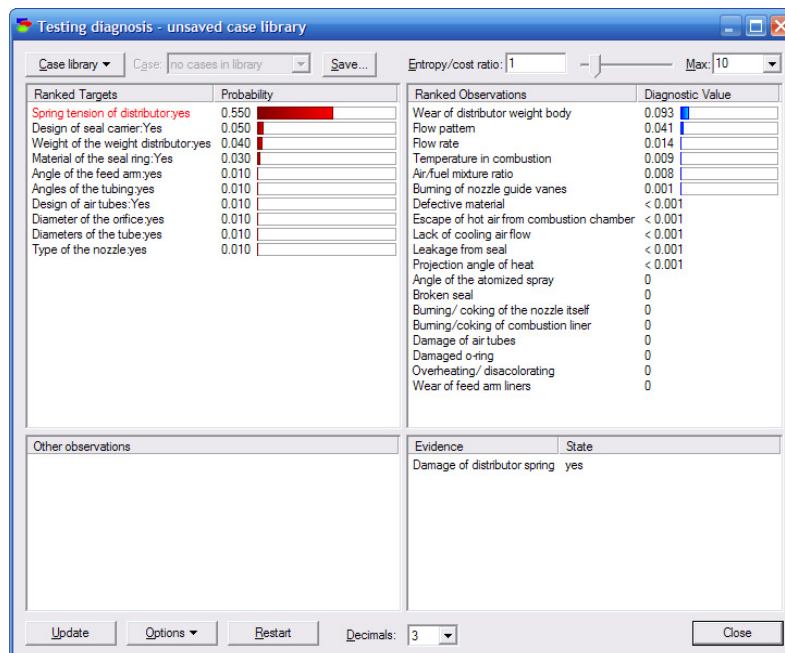
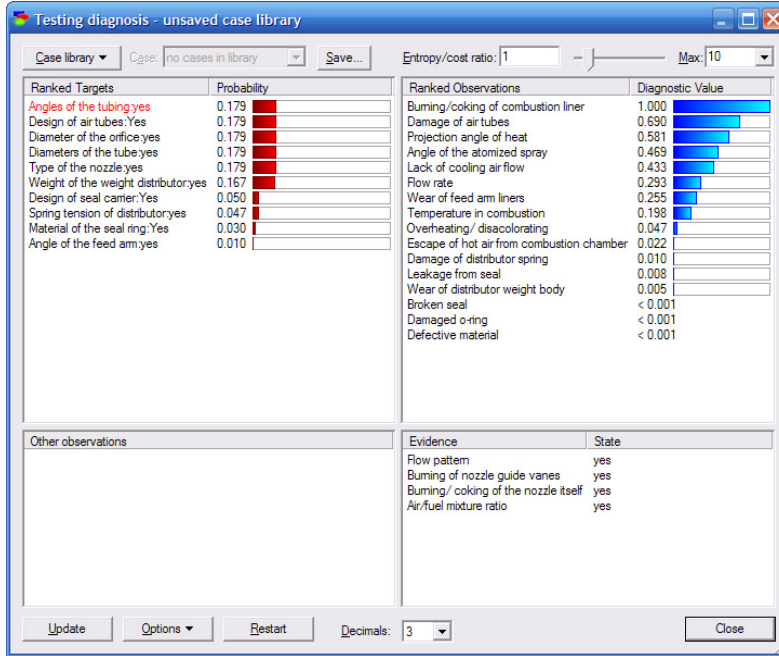


Figure 5-5 Causal network for the fuel nozzle knowledge

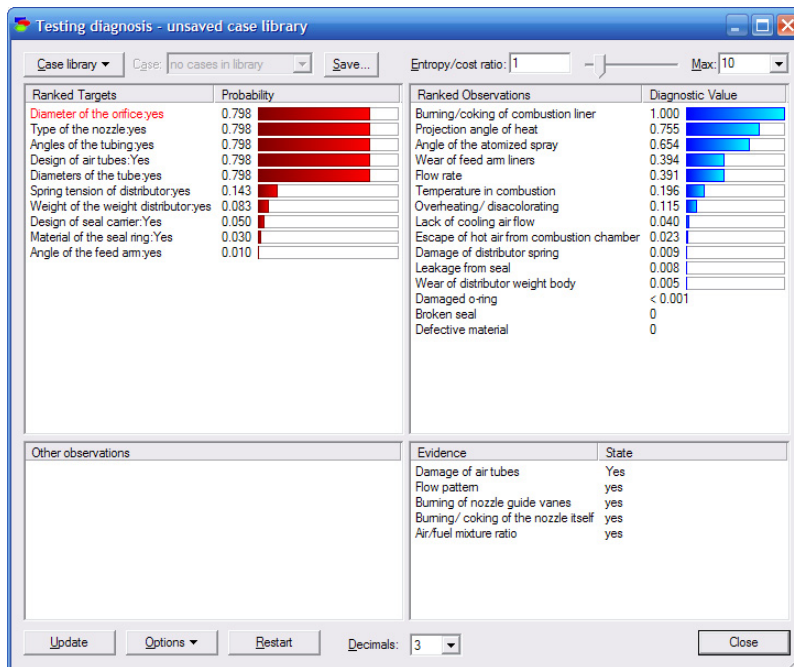
One of the features for causal product development knowledge is decision support, which provides decision alternatives in order to make appropriate and better decisions in different scenarios. Figure 5-6 (a) illustrates a diagnosis reasoning for decision support. The spring tension of a distributor is the most affected design issue when the distributor spring fails or is damaged. Figure 5-6 (a)'s right upper box shows the ordered list of affected design causes. Figure 5-6 (b) shows the ordered list of affected design causes when the failures of flow pattern, burning of nozzle guide vanes, burning/coking of the nozzle itself, and air/fuel mixture ratio issues occur. Figure 5-6(c) shows another case of failure and the ordered list of affected design causes. Therefore, these ordered lists can be used to support a decision in product design stage.



(a) Decision alternative with one observation



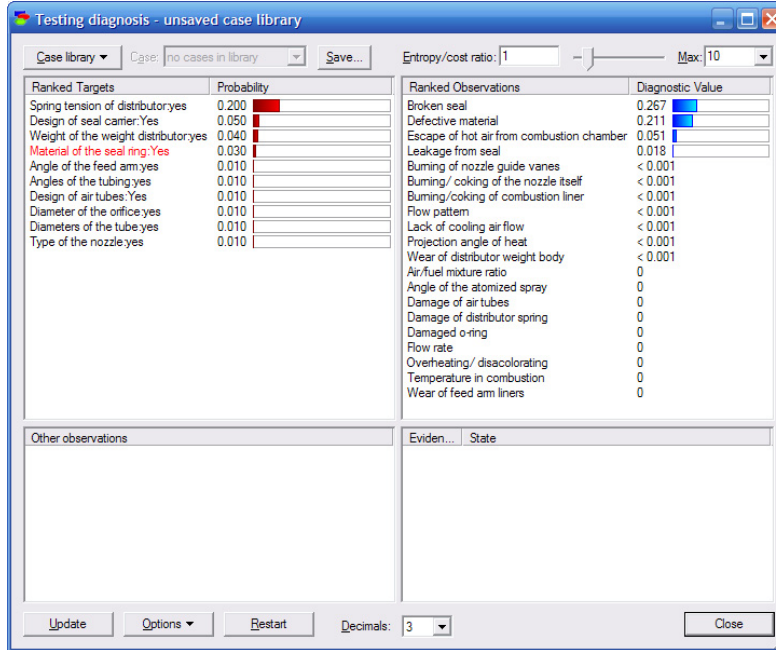
(b) Decision alternative with four observations



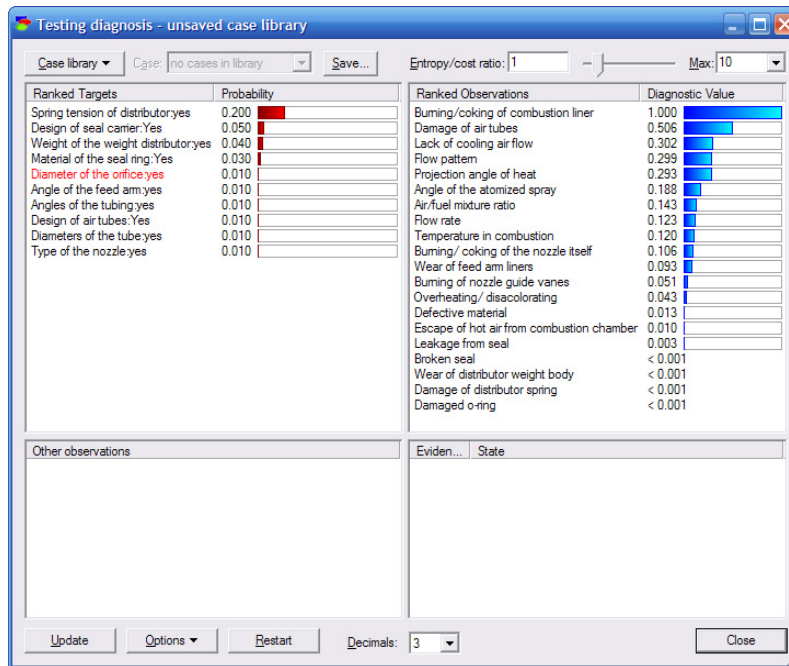
(c) Decision alternative with five observations

Figure 5-6 Examples of decision alternative using causal knowledge diagnosis

The other important feature of causal product development knowledge is reasoning, which includes prediction and diagnosis. Figure 5-7 illustrates two examples of predictions. If a designer modifies the material of the seal ring, it will provide a list of potential failures (see Figure 5-7 (a)'s left upper box). The list shows an ordered effect from the modification by designer. Figure 5-7 (b) shows another case of prediction. For instance, when an aerospace engine designer designs a new fuel nozzle for the engine, the designer has to consider multiple factors, which affect the performance of the engine. The designer indicates the material of seal ring has a problem and the material of seal ring should be replaced with a different material. However, the replacement will affect other designs of fuel nozzle. The designer needs to check what are the effects of the replacement to other parts. This reasoning feature can warn to designer for the effects of the replacement. The order of effects is listed (i.e., broken seal, defective material, escape of hot air from combustion chamber, leakage from seal, etc.) in Figure 5-7 (a). The diagnosis feature is presented in Figure 5-6 with an explanation of the decision support feature.



(a) The effects of material modification



(b) The effects of diameter modification

Figure 5-7 The examples of the effects of the causal knowledge prediction

5.5 Conclusion

Most knowledge is represented as procedural knowledge because procedural knowledge includes declarative knowledge and contextual knowledge. Product development knowledge can be represented by procedural knowledge but unwieldy processes are required to define procedural knowledge individually. Furthermore, during the product development processes, this procedure knowledge cannot fully represent evolutionary and dynamic product development knowledge. Therefore, this research presents mathematical definitions of procedural product development knowledge, causal product development knowledge, and the knowledge transformation by set theory. This research develops a set-theory-based knowledge transformation method to match the components of knowledge models between procedural and causal knowledge and mathematically defines the relationship between procedural and causal product development knowledge. Based on the comparison with four perspectives, this research concludes that causal knowledge represents more knowledge expression, reasoning, decision alternative representation, and knowledge cultivation ability than procedural knowledge. Furthermore, causal knowledge has sound mathematical theorem and knowledge integration by structure and belief integration. This research implements knowledge models by SysML. With this demonstration, causal knowledge (CK) represents more product development knowledge than procedural knowledge (PK). CK's block diagram includes more detail knowledge with internal block diagrams; more requirements were required to

represent CK than PK. The state machine diagram for CK can handle dynamic, increased information, but PK cannot support dynamic information. This research also demonstrates the features of the causal knowledge with a real industrial case, a fuel nozzle of an aerospace engine in product development. This research also concludes that CK's characteristics are more beneficial to represent product development knowledge than PK and provide more functions of knowledge practices.

These results of the knowledge comparison and transformation method can be used to represent, store, retrieval, and reuse the product development knowledge since the knowledge is formally defined. In the product development processes, the amount of knowledge is often difficult to count and imagine. A CK model can represent similar multiple PK models; therefore CK has a potential to generate knowledge compression. Furthermore, the knowledge expression ability of the model is increased and the possibility of the knowledge reuse increases in product development. Un-captured knowledge will be decreased in product development. CK's knowledge inference can increase the ratio of the knowledge reuse in the product development, since knowledge inference provides several advantages to reuse knowledge: 1) more decision alternatives; 2) predictive reasoning to advise design decision; 3) diagnostic reasoning to acquire design faults in current product design; 4) dynamic knowledge allowance to give flexible product design; and 5) knowledge integration to keep the product development knowledge in a team, a department, and a company. The author has developed

new causal CK evaluation method to evaluate different CK that can be obtained in overall product development processes. The results will be reported in the next Chapter.

CHAPTER 6

DEGREE OF CAUSAL REPRESENTATION

The aim of this chapter is to present a new causal design knowledge evaluation for product development knowledge management. Current product development processes still include unintended feedback due to insufficient product design knowledge—a problem that a causal design knowledge evaluation and support system and its reasoning capability is designed to overcome. This chapter presents a new method and system for causal design knowledge evaluation and support to appropriately, easily, and quickly design a new product and to prevent a future potential failure. This research develops a degree of a causal representation-based causal knowledge evaluation method as one of the main functions of the product design knowledge support system. Finally, the implementation of a causal product design knowledge support system is presented with a new valve design case scenario.

6.1 Causal Design Knowledge Evaluation and Support

A framework of the causal design knowledge evaluation and support system is composed with a design application, causal knowledge support system, and causal knowledge-bases as shown in Figure 6-1. A designer utilizes a design application for designing a new product. The designer searches an existing design of the product and loads that design into the design application. The new design is slightly modified from the existing design one. When the designer modifies the existing design, he or she is able to obtain the effects of the modification in the design. The causal

knowledge support system provides the design analysis and evaluation toward the product development life cycle from design modification at the design stage. The provided design analysis and evaluation includes a degree of causal representation (DCR), an ordered list of the effects of the design modification, and causal design knowledge. One of the core functions in the causal design knowledge evaluation and support system is a causal knowledge support system, which utilizes a DCR-based method. The causal knowledge support system is composed with causal knowledge representation, evaluation, and integration. Causal knowledge representation is a knowledge representation method for design knowledge using a cause-and-effect relationship. Causal knowledge evaluation is a DCR-based method for measuring a causal representation within the causal knowledge. Causal knowledge integration is a knowledge-combining method that allows us to integrate different areas of knowledge within the broader product design.

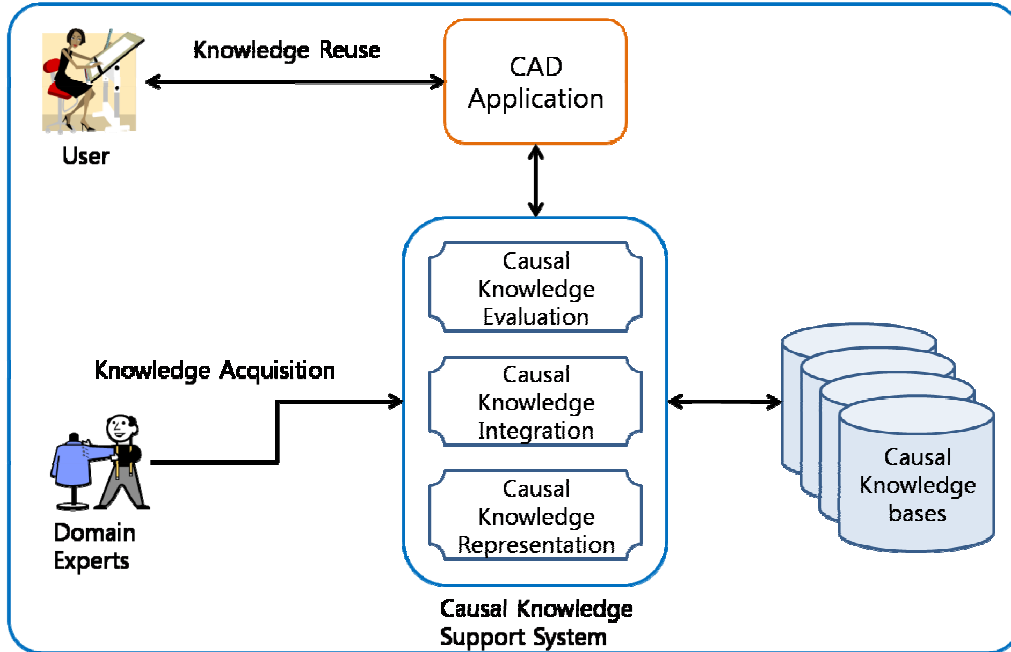


Figure 6-1 Framework of the causal design knowledge evaluation and support system

6.2 Mathematical Representation of Causal Knowledge

A mathematical definition is required to use the causal knowledge network (CKN) for causal knowledge evaluation. There are two types of the CKN (Figure 6-2): definition 6-1, CKN with weighted edges (e.g., fuzzy cognitive map); definition 6-2, CKN with weighted vertices (e.g., Bayesian belief network). Definition 6-1 represents CKN with weighted vertices. The CKN is composed with the sets of vertices (V), edges (E), and weights (W) of the knowledge network. V is a set of all vertices in the knowledge network. E is a set of connected edges in the knowledge network, and W is a set of weights in the knowledge network.

Definition 6-1 Causal knowledge network with weighted vertices

The causal knowledge network is represented by $CKN = \{V, E, W\}$ as a weighted-directed graph, where:

n is the total number of vertices in CKN;

V is a set of all vertices in CKN and $V = \{v_i; i = 1, \dots, n\}$;

E is a set of connected edge in CKN and $E = \{e_{jk}; j, k = 1, \dots, n, j \neq k\}$,

where $e_{ij} = 1$ if the edge ij is existed, otherwise, $e_{ij} = 0$;

W is a set of weights in CKN (e.g., the weight can be probability of V) and $W = \{w_l; l = 1, \dots, n\}$.

Figure 6-2's Network 1 is a CKN with weighted vertices and can be represented as below.

$$CKN = \{V, E, W\},$$

$$V = \{v_1, v_2, v_3, v_4\},$$

$$E = \{e_{12}, e_{21}, e_{23}, e_{31}, e_{41}, e_{42}\},$$

$$W = \{0.6, 0.2, 0.7, 0.9\}$$

A CKN with weighted edges can be represented based on definition 6-1 with modified weight (W). In this definition, the weights of the CKN are modified from w_l to w_{jk} , which means the weights are on the edges.

Definition 2 Causal knowledge with weighted edges

The causal knowledge network is represented by $CKN = \{V, E, W\}$ as a weighted directed graph where:

n is the total number of vertices in CKN;

V is a set of all vertices in CKN and $V = \{v_i ; i = 1, \dots, n\}$;
 E is a set of connected edge in CKN and $E = \{e_{jk} ; j, k = 1, \dots, n, j \neq k\}$,
 where $e_{ij} = 1$ if the edge ij is existed, otherwise, $e_{ij} = 0$;
 W is a set of weights in CKN (i.e., the weight can be probability of
 connected edge (V)) and
 $W = \{w_{jk} ; j, k = 1, \dots, n, j \neq k\}$.

Figure 6-2's Network 2 is a CKN with weighted edges, which is represented by definition 6-2.

$$CKN = \{V, E, W\},$$

$$V = \{v_1, v_2, v_3, v_4\},$$

$$E = \{e_{12}, e_{13}, e_{21}, e_{23}, e_{24}, e_{31}, e_{34}, e_{41}, e_{42}, e_{43}\},$$

$$W = \{-0.6, 0.5, 0.5, 0.2, 0.7, 0.8, 0.3, 0.7, -0.7, 0.7\}$$

Figure 6-2 Examples of causal knowledge networks

To evaluate them, first, we defined CKNs (definition 6-1 and 6-2) as illustrated in the previous section. Next, as in this chapter, we define the degree of causal representation (DCR) as a causal representation measure. The DCR is a combined measure with causality (C) and network connectivity (NC) for a CKN with weighted vertices, and is a weighted network connectivity (WNC) for a CKN with weighted edges.

6.3 Evaluation of Causal Knowledge Network with Weighted Vertices

The DCR of a CKN with weighted vertices is decomposed with Causality (C), which represents the effects of each vertex, and Network Connectivity (NC), which measures the ratio of the connection. Causality is a measure how a CKN represents a causal relationship with the consideration of incoming and outgoing edges of each vertex. Definition 6-3 represents causality. NP is the number of parent vertices, which is the same as the number of incoming edges. P is the measure of the distributed of weights. The weight of each vertex is distributed based on the number of states (S) in the vertex. For instance, if the number of states is two, the weights are distributed based on 0.5 (e.g. (0.2, 0.8), (0.3, 0.7), (0.1, 0.9), ...). In this case, P is the variance of this distribution.

Definition 6-3 Causality

Causality (C) is based on a number of affected vertices and the measure of the distributed of weights in a vertex.

$$C = \sum (NP_i \times P_i),$$

where n is the total number of vertices in CKN;

NP_i is a number of parent vertices or is a number of incoming edges and

$$NP_i = \sum e_{ki}; k = 1, \dots, n;$$

P_i is the measure of the distributed of weights and $P_i = \sum (P_{i,q} - (1/S_i))^2 / (n - 1)$,

where $q = 1, \dots, |S| \times 2^{NP_i}$, S_i is a number of the state in each vertex and n is the total number of V .

Definition 6-4 explains Network Connectivity. NC represents the connection of the network with the ratio of total connections in CKN. The ratio of total connections is an accumulation of each edge's ratio of connections based on the connection, which includes direct and indirect connection. A direct connection means that an edge exists between vertex 1 (v_1) to vertex 2 (v_2). An indirect connection means that edges exist from vertex 1 (v_1) to vertex 2 (v_2) through another vertex.

Definition 6-4 Network Connectivity

Network Connectivity (NC) represents the connection of the network with the ratio of total connections in causal knowledge network.

$$NC = \sum u_{ij},$$

where n is the total number of vertices in CKN;

u_{ij} is the ratio of total connections and $u_{ij} = r_{ij} / t_i$;

r_{ij} is the relation of the connections, which includes direct and indirect

connections and $r_{ij} = (d_{ij} + ind_{ij})$;

t_i is the total number of connections, which includes direct and indirect connection, from vertex i to vertex k and $t_i = \sum t_{ik} = \sum (d_{ik} + ind_{ik})$, $k = 1, \dots, n$;

d_{ij} is the number of direct link from V_i to V_j ;

ind_{ij} is the number of indirect link form V_i to V_j .

In the proposed analysis method for the CKN with weighted vertices, $DCR = NP_i \times P_i \times \sum u_{ij}$ is used. To illustrate this method, the network 1 in Figure 6-3 is used. We assumed each vertex has only two states (Yes/No) for the simplicity of illustration. For the network 1, C is calculated by NP and P and NC is calculated by the relation of the connection.

The numbers of parent vertices are $NP_1 = e_{k1}$, $e_{k1} = 3$; $NP_2 = 2$; $NP_3 = 1$; and $NP_4 = 0$. For the vertex v_1 , the weights of states are $P(\text{Yes}) = 0.6$, $P(\text{No}) = 0.4$. The measures of the distributed of weights are $P_1 = ((0.6 - 0.5)^2 + (0.4 - 0.5)^2) / (4 - 1) = 0.0067$, $P_2 = 0.06$, $P_3 = 0.0267$, $P_4 = 0.1067$. Therefore, the C is $6 \times 0.2 = 1.2$. The relations of connections are $r_{12} = 1 + 0 = 1$, $r_{13} = 1$, $r_{21} = 2$, $r_{23} = 1$, $r_{13} = 1$, $r_{32} = 1$, $r_{41} = 3$, $r_{42} = 2$, $r_{43} = 2$. The total numbers of connection are $t_1 = \sum t_{1k} = \sum (d_{1k} + ind_{1k}) = 3 + 12 = 15$, $k = 1, \dots, n$, $t_1 = t_2 = t_3 = t_4 = 15$. The ratios of the connection are $u_{12} = 1/15 = 0.0667$, $u_{13} = 0.0667$, $u_{21} = 0.1333$, $u_{23} = 0.0667$, $u_{13} = 0.0667$, $u_{32} = 0.0667$, $u_{41} = 0.2$, $u_{42} = 0.1333$, and $u_{43} = 0.1333$. Therefore, NP is 6, P is 0.6, C is 3.6, NC is 2.6, and DCR is 9.36. Using the same calculation for network 2, NP is 9, P is 0.36, C is 3.24, NC is 6.6, and DCR is 21.384. Based on this calculation, network 2's DCR is higher than network 1. It means network 2 represents approximately two

times more causality and network connectivity than network 1. More detailed discussion on this is in section 6.4.

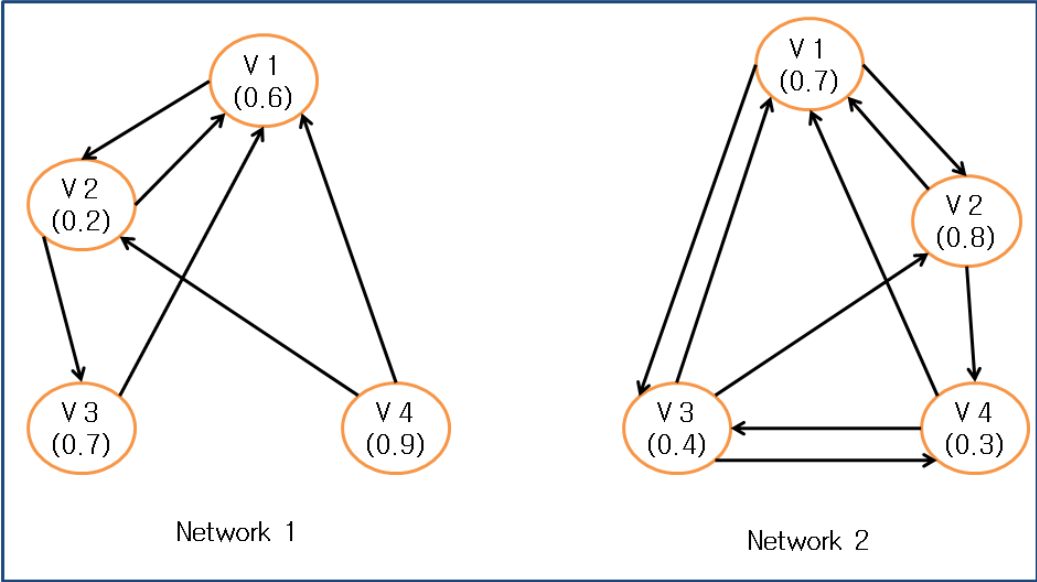


Figure 6-3 Examples of knowledge network with weighted vertices

6.4 Evaluation of Causal Knowledge Network with Weighted Edges

The DCR of CKN with weighted edges uses a weighted network connectivity (*WNC*) as shown in Definition 6-5. *WNC* is composed with network connectivity and the normalized edge weights of vertices. Network connectivity is used the same definition of Definition 6-4. The normalized edge weights of the vertices represent the effects of the edges from a normalized zero line (in this research 0.5 is used.). If the normalized edge weight is higher than normalized zero line, it represents a positive effect of original weight. If the normalized edge weight is lower than a normalized zero line, it represents a negative effect of original weight.

The reason of normalization is because the negative effect of original weight cannot appropriately be calculated for *WNC*: even if the effect of weight is negative, this effect should increase *WNC*, however, the original weight of negative effect decreases *WNC*. Therefore, the normalization of weights can adjust this problem. *WNC* is the sum of the direct edge and the indirect edges. The indirect of the edges in *WNC* uses a special function (Π), which is a multiplication function of connections. For instance, the indirect of edges from vertex 1 to vertex 2 can be calculated with three parts; 1) vertex 1 to vertex k_1 , 2) vertex k_1 to vertex k_2 , 3) vertex k_2 to vertex 2, where k_1 and k_2 are in $V = \{v_i ; i = 1, \dots, n\}$ and cannot be vertices 1 and 2. If k_1 and k_2 are the same, this means that there is only one intermediate vertex (e.g., vertex 1 to vertex k and vertex k to vertex 2). If k_1 and k_2 are different, the multiplication function of connections can calculate this indirect connections (e.g., if $k_1 = 3$ and $k_2 = 4$, the connection is vertex 1 to vertex 3, vertex 3 to vertex 4, vertex 4 to vertex 2. Also, k_1 and k_2 can be multiple).

Definition 6-5 Weighted network connectivity

Weighted network connectivity (*WNC*) represents the connection of the network with the weights of edges in causal knowledge network.

WNC is composed by the ratio of total connections (u) and the normalized weight (nw) distribution of each edge.

$$\begin{aligned}
 WNC_{ij} &= \text{sum of } WNC \text{ for direct edge and indirect edges} \\
 &= u_{ij} \times e_{ij} \times p_{ij}
 \end{aligned}$$

$$+ \sum_{k=1}^N (u_{ik1} \times e_{ik1} \times p_{ik1}) \times (u_{k2j} \times e_{k2j} \times p_{k2j}) \times \prod (u_{k1k2} \times e_{k1k2} \times p_{k1k2}); k_1, k_2 = 1, \dots, n; k_1, k_2 \neq i, j,$$

where u_{ij} is the ratio of total connection and $u_{ij} = r_{ij} / t_i$;

r_{ij} is the relation of the connection, which includes direct and indirect connection and $r_{ij} = (d_{ij} + ind_{ij})$;

t_i is the total number of connection, which includes direct and indirect connection, from vertex i to vertex k and $t_i = \sum t_{ik} = \sum (d_{ik} + ind_{ik})$, $k = 1, \dots, n$;

d_{ij} is the number of direct link from V_i to V_j ;

ind_{ij} is the number of indirect link form V_i to V_j ;

nw_{ij} is a set of the normalized weights and $nw_{ij} = (w_{ij} / 2 + 0.5) \times e_{ij}$;

p_{ij} is the measure of the distributed of weights of the edges and $p_{ij} = \sum (nw_{ij} - 0.5)^2 \times e_{ij}$;

and $0 < nw_{ij} < 0.5$ if the weight of the edge has negative, $0.5 < nw_{ij} < 1$ if the weight of the edge has positive.

In this research, a CKN with weighted edges is represented by $DCR = \sum_{i=1}^N (\sum_{j=1}^N WNC_{ij}) \times C_{ij}$. For example, Figure 6-4 shows two knowledge networks with weighted edges. For the network 1, the normalized weights and network connectivity are calculated for WNC .

The normalized weights are $nw_{12} = -0.6/2 + 0.5 = 0.2$, $nw_{13} = 0.75$, $nw_{21} = 0.75$, $nw_{24} = 0.85$, $nw_{31} = 0.9$, $nw_{34} = 0.65$, $nw_{42} = 0.15$, and $nw_{43} = 0.85$. The measure of the distributed of weights is $p_{12} = (0.2 - 0.5)^2 \times 1 / (4 - 1) = 0.03$, $p_{13} = 0.02083$, $p_{21} = 0.02083$, $p_{24} = 0.04083$, $p_{31} = 0.05333$, $p_{34} = 0.0075$, $p_{42} = 0.04083$, $p_{43} = 0.04083$. The relations of

connection are $r_{12}=1+1=2$, $r_{13}=2$, $r_{14}=2$, $r_{21}=2$, $r_{23}=2$, $r_{24}=2$, $r_{31}=2$, $r_{32}=2$, $r_{34}=2$, $r_{41}=2$, $r_{42}=2$, and $r_{43}=2$. The total numbers of connection are $t_1=3+(2+2) \times 3=15$, $t_1=t_2=t_3=t_4=15$. The WNC are $wnc_{12}=0.1333 \times 0.03 \times 1 + 0.1333 \times 0.02083 \times 1 + 0.1333 \times 0.0075 \times 1 + 0.1333 \times 0.04083 \times 1 = 0.004$, $wnc_{13}=0.00279$, $wnc_{14}=0.00025$, $wnc_{21}=0.0028$, $wnc_{23}=0.00037$, $wnc_{24}=0.00545$, $wnc_{31}=0.00711$, $wnc_{32}=0.00033$, $wnc_{34}=0.00101$, $wnc_{41}=0.00054$, $wnc_{42}=0.00546$, and $wnc_{43}=0.05045$. Based on this calculation, nw is 5.1, p is 2.55, WNC is 0.03557, and DCR is 0.4626. Using the same calculation for the network 2, nw is 6.55, p is 2.9917, c is 0.07524, and DCR is 1.474376. Network 2's DCR is higher than network 1. It means network 2 represents approximately three times more weighted network connectivity than network 1. Detailed discussion about this interpretation is shown in the next sections.

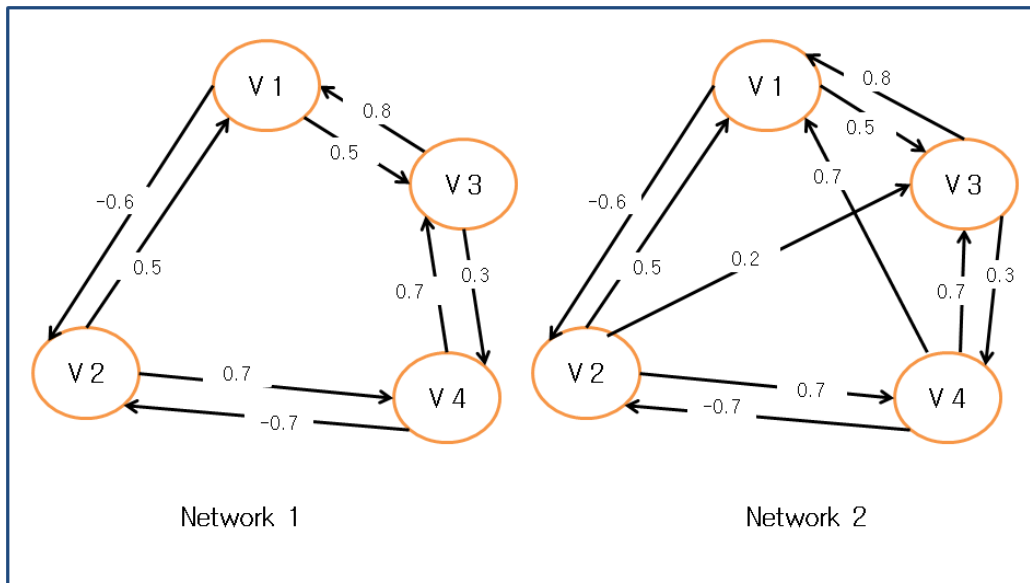


Figure 6-4 Examples of knowledge network with weighted edges

Based on causal knowledge representation and evaluation method, the causal knowledge evaluation is composed with network analysis interface, network analysis manager, optimality evaluation engines (for network connectivity, causality, and weighted network connectivity), and a knowledge-base. Network analysis interface provides a visual presentation of a knowledge network, which is selected for analysis by a user. A network analysis manager coordinates the network analysis interface, optimality evaluation engines, and knowledge-base. This manager receives a request for the network analysis from the user. Via the manager, the received request is sent to the optimality evaluation engines for the optimality calculation, and the requested network is displayed in the network analysis interface. The network manager communicates with the knowledge-base for finding the requested knowledge. Optimality evaluation engines include three sub-engines: network connectivity, causality, and weighted network connectivity. The network connectivity engine calculates the ratio of the connection for the network with weighted vertices in the knowledge network, and the weighted network connectivity engine is for the network with weighted edges. Figure 3 shows the examples of the networks with weighted vertices and edges. A causality engine calculates the ratio of the causal relationship between the knowledge network components. After the analysis of the network, the result of the network analysis can be displayed in the network analysis interface based on the user's request.

The process in the causal knowledge network analysis system is illustrated in Figure 6-5. The number in the figure indicates the sequence of analysis processes. A user analyzes a causal knowledge network. The user selects a network and requests the network analysis results in DCR (degree of causal representation). The selected network and the request are sent by the network analysis interface to the network analysis manager. The network analysis manager finds the network from the knowledge-base and distinguishes the characteristics of the network. There are two different causal knowledge networks in knowledge-base: a knowledge network with weighted vertices and one with weighted edges. If the selected network is the knowledge network with weighted vertices, a DCR with weighted vertices is generated with network connectivity and causality. Similarly, a DCR with weighted edges with weighted network connectivity is generated for the knowledge network with weighted edges. The generated DCR is displayed to the user by the network analysis interface.

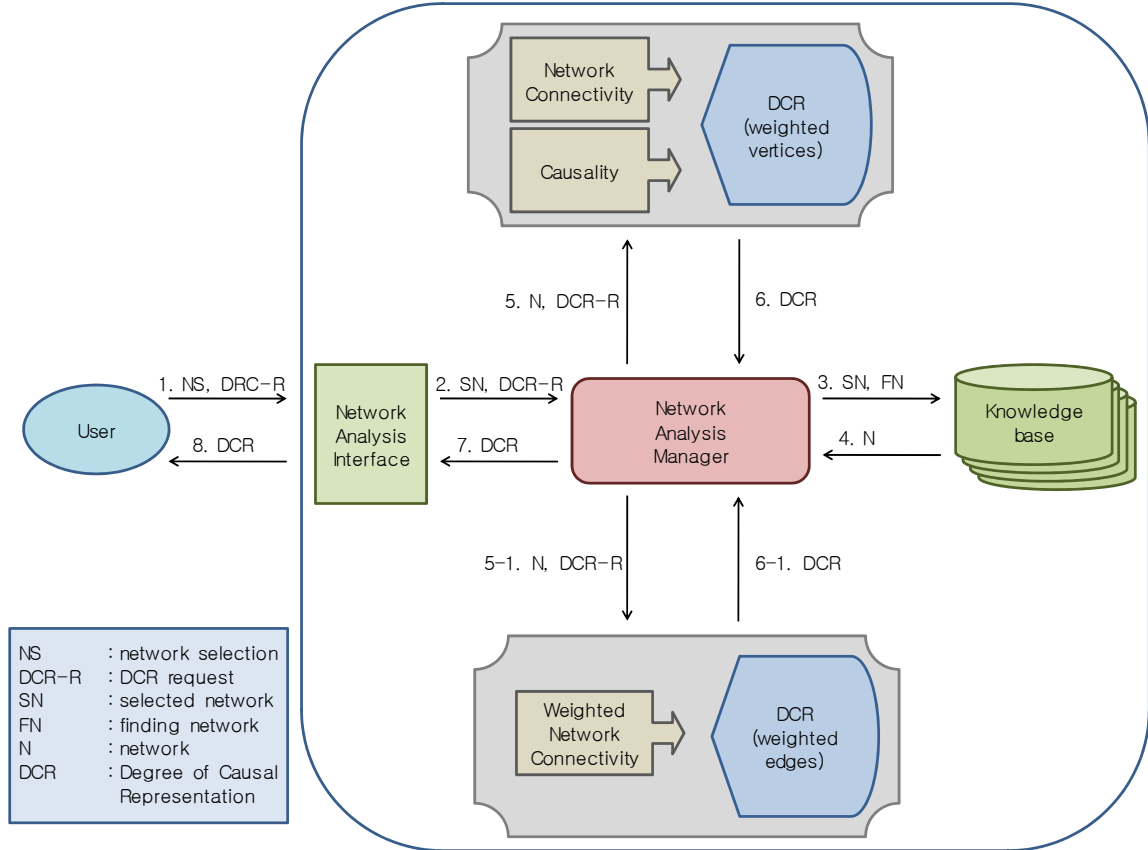


Figure 6-5 Processes of the causal knowledge evaluation

6.5 Validation of DCR-based Causal Knowledge Evaluation

6.5.1 Comparison of Different Causal Knowledge

In the previous chapter, we defined the knowledge evaluation method for CKN with weighted vertices and edges. The result of DCR for both networks show network 2 has a higher degree of DCR than network 1 (Figure 6-3 and Figure 6-4). These results are used to compare connectivity and weight of CKN. In this chapter, the effects of weights and network connectivity are compared. The DCR results of the CKN with weighted edges are 0.0916, 0.2964, and 0.627 for the network 1, 2,

and 3, as in Figure 6-8. The result indicates the complicated network model has a higher DCR. To validate this result, for the network model, three types of models are tested as shown in Figure 6-6: 1) a model that has only direct edges (Network 1); 2) a model that has only direct edges with minimum indirect edges (Network 2); and 3) a model that has more indirect edges (Network 3). For the weight of edges, three weights are used: 0.1, 0.5, and 0.9. If the weight of each edge is 0.1, every edge's weight is 0.1—e.g., the edge from vertex 1 to vertex 2 is 0.1 instead of -0.6 in Figure 6-6's Network 1.

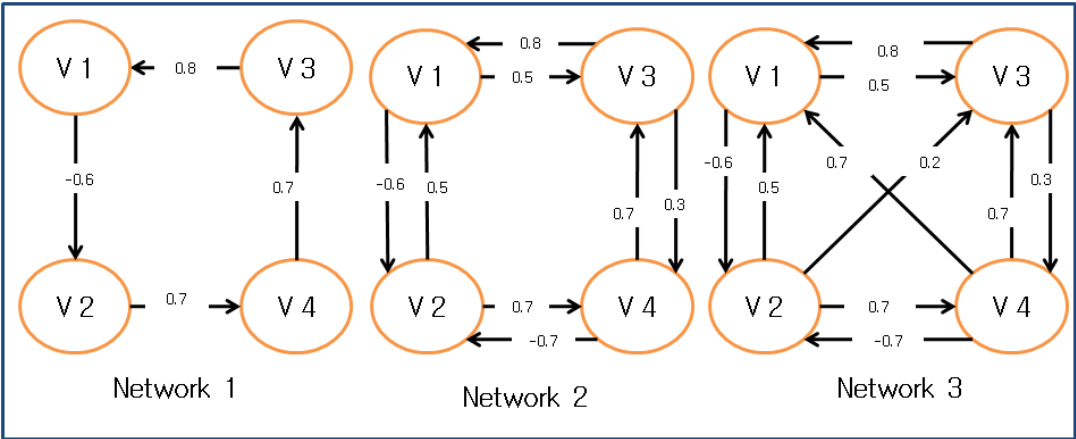
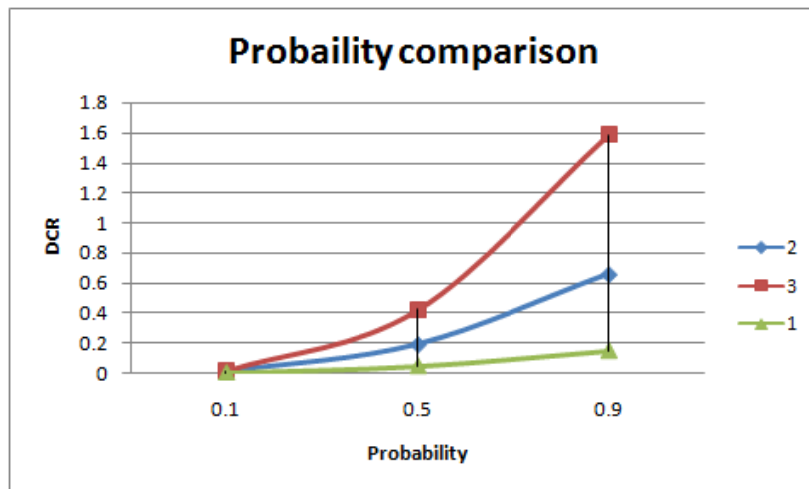


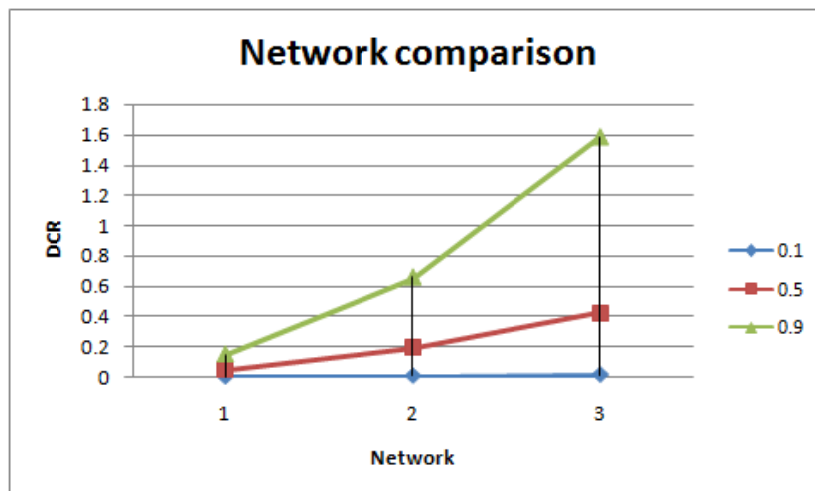
Figure 6-6 Examples of causal knowledge network with weighted edges for comparison

The results of these comparisons are shown in Figure 6-7. There is an assumption, which is that the number of vertices is the same in networks. In the network comparison, network models 2 and 3 have a higher DCR than network model 1. Network model 2 is approximately 4.17 times, and network model 3 is approximately 9.4 times higher. These results indicate that the DCR is higher if the network model has more connection (number of edges). In the probability

comparison, the networks with 0.5 and 0.9 have higher DCR than network with 0.1. The network with 0.5 is 25.8 times higher, and the one with 0.9 is 90.1 times higher. These results indicate that the DCR is higher if the network has higher probability. Also, Figure 6-7 shows the probability effect is exponentially increased with a complex network model.



(a) Probability comparison



(b) Network comparison

Figure 6-7 Comparison results for causal knowledge network with weighted edges

The similar comparison for the CKN with weighted vertices is tested. The DCR results of the CKN with weighted vertices are 0.2133, 2.3467, and 5.3333 for networks 1, 2, and 3, as in Figure 6-8. The similar conclusion is indicated with the previous comparison with CKN weighted edges. The comparison test is similar with the previous comparison with three types of network model and five different vertices weights (0.1, 0.4, 0.5, 0.75, 0.9) with a assumption, which is that the number of states (S) is 2 for all vertices, and the number of vertices is the same in networks. After comparison, we indicate the weights 0.1 and 0.9 are the same and 0.5 is no effect as we expected. The results of network comparison are that network 2 is 11 time higher and network 3 is 25 times higher than network 1. For a probability comparison, network with 0.75 is 6.25 times higher and network with 0.1 is 16 times higher than network with 0.4. Figure 6-9's probability comparison shows the distribution is symmetric at 0.5 and a network comparison shows the probability effect is exponentially increased, similar to the network with weighted edges. If the number of states (S) is changed from 2 to 3, the probability comparison distribution will be changed to asymmetric at 0.33.

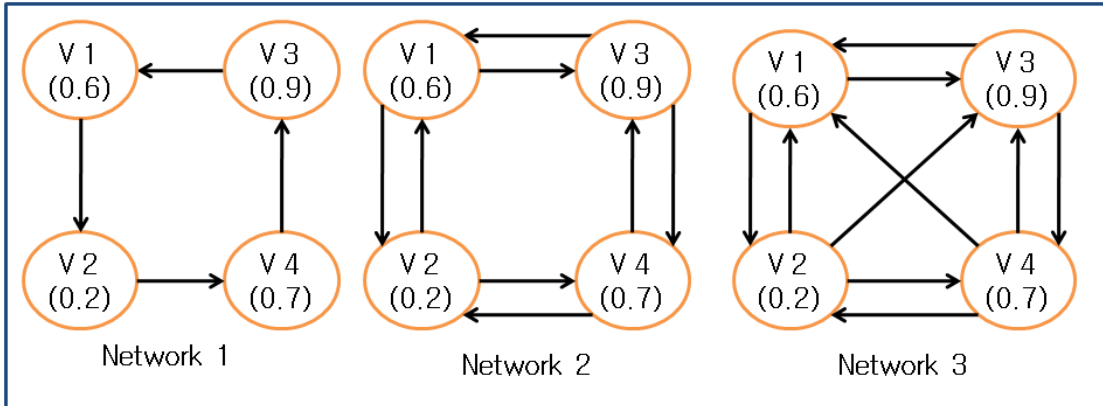
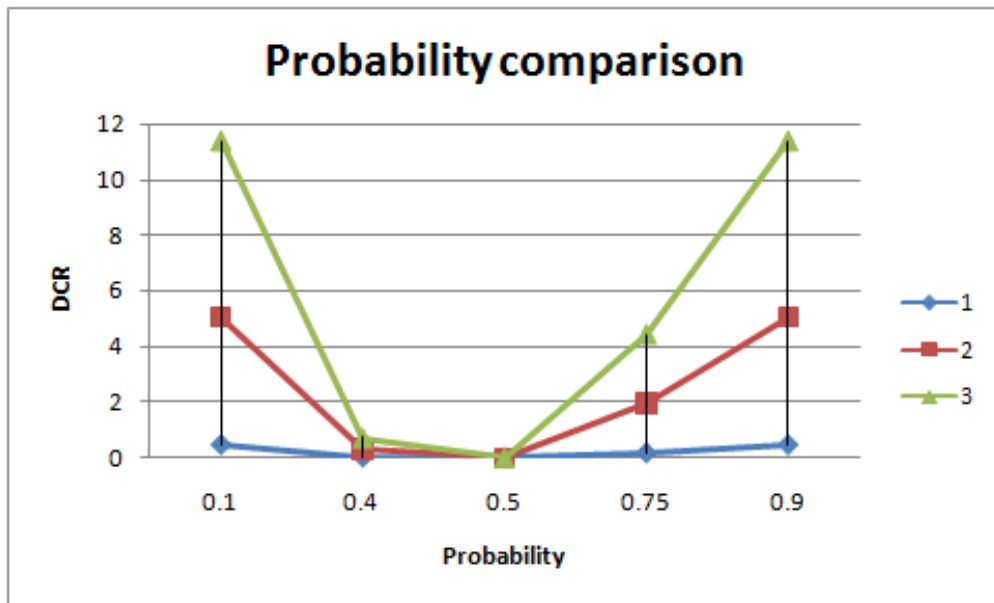
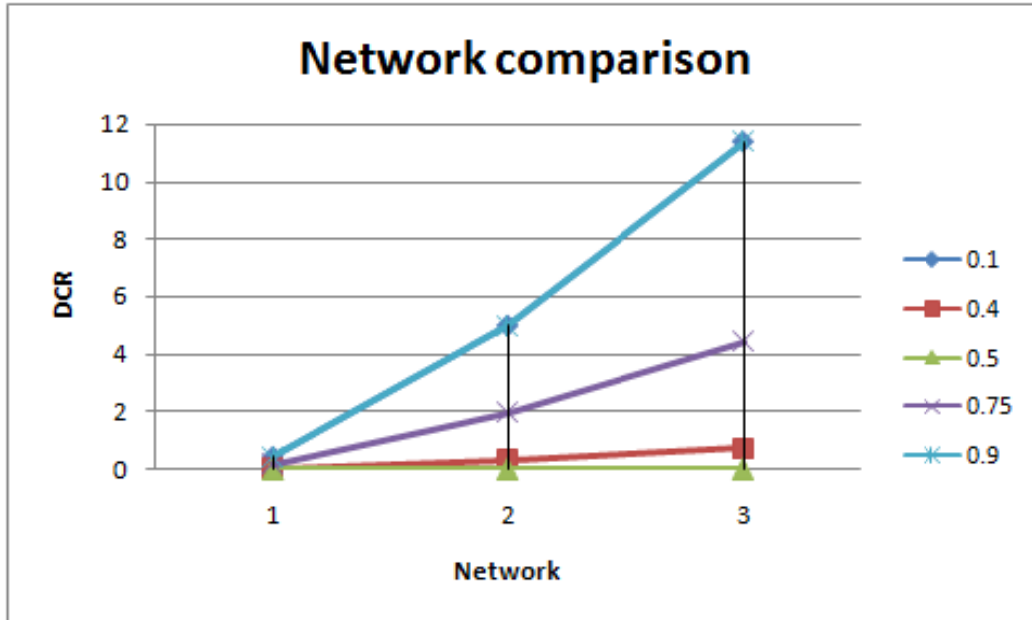


Figure 6-8 Examples of causal knowledge network with weighted vertices for comparison



(a) Probability comparison



(b) Network comparison

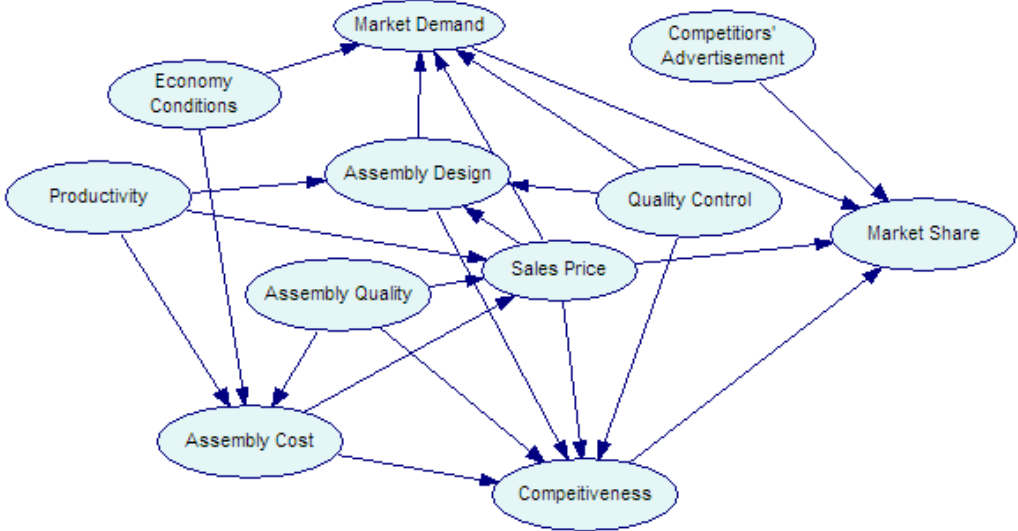
Figure 6-9 Comparison results for causal knowledge network with weighted vertices

The comparison results are as follows: 1) the more complex network model has higher DCR, 2) the network with higher weight has higher DCR, 3) the effect of weight increases with the more complex network. The next chapter will show more detail implementation for the functions in this network analysis method.

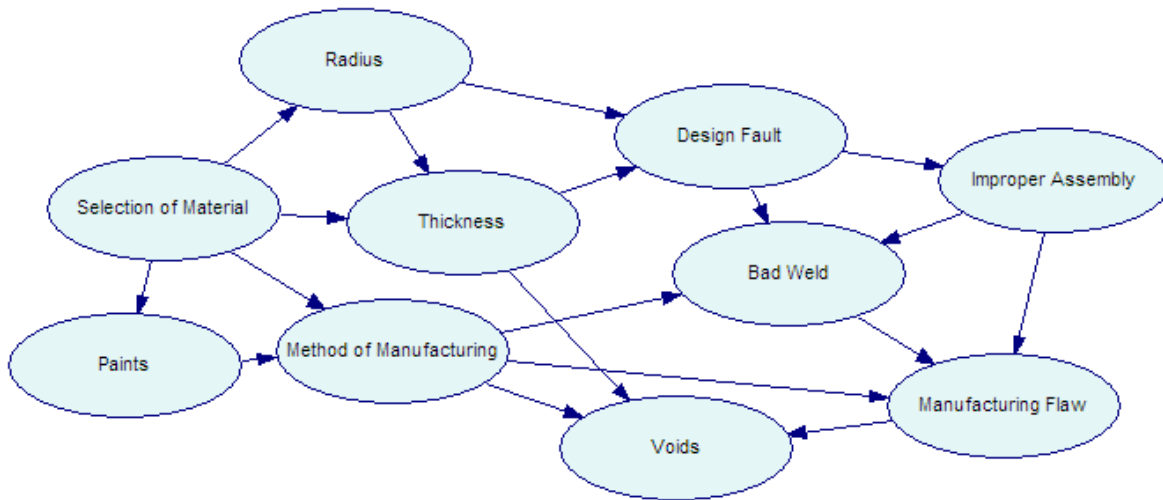
6.5.2 Case Study

In this chapter we present and discuss three case studies—assembly design, wheel, and fuel nozzle—to evaluate knowledge representation and evaluation using the proposed DCR-based method. All use a causal knowledge network. The assembly design case represents knowledge, which is the relationship between assembly design and decision environment (Figure 6-10 a). The wheel and fuel

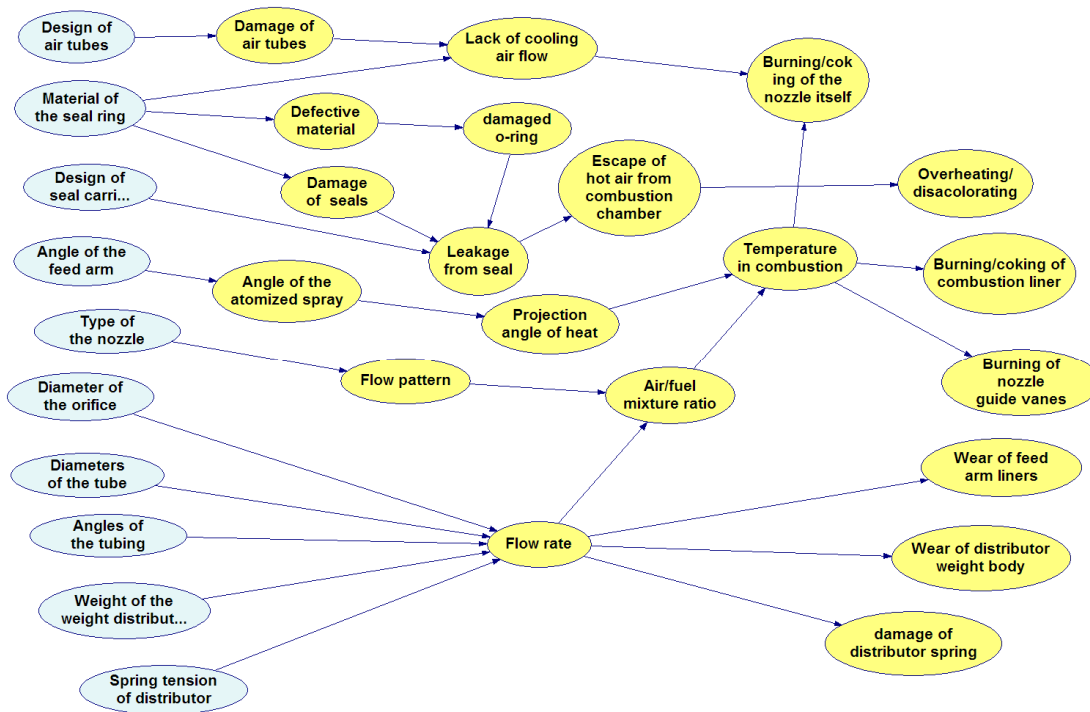
nozzle cases illustrate the relationship of the design and maintenance aspects of each case (Figure 6-10 c). These three cases are conducted with three different causal knowledge: a Bayesian belief network (BBN), which is directly generated by the domain experts; a Bayesian belief network from fuzzy cognitive map (FCM-BBN), which is a knowledge network converted from FCM using the FCM-BBN method from our previous research [Cheah 2007]; and a modified Bayesian belief network from fuzzy cognitive map (FCM-BBN-M) [Kim 2008], which is a FCM-BBN without direct edge if any indirect edge exists. Finally, this research tests these cases with my developed CK evaluation method.



(a) Causal knowledge for assembly design



(b) Causal knowledge for wheel design and maintenance



(C) Causal knowledge for fuel nozzle design and maintenance

Figure 6-10 Examples of causal knowledge

The CK evaluation method analyzes causality of vertices, edges, and network structure. In the assembly design case, BBN is the best CKN and is analyzed with causality (14.44), network connectivity (7.595), and DCR (109.672). However, very little differences among the three DCR were seen. BBN and FCM-BBN is similar DCR in the wheel case, and FCM-BBN-M is different from them. In the fuel nozzle case, FCM-BBN is the best CKN with significant difference. Table 6-1 shows more test results of CK evaluation.

Table 6-1 The results of causal knowledge network analysis

Case	Criterion	BBN	FCM-BBN	FCM-BBN-M
Market-share	No. of Nodes	11	11	11
	No. of Parent	20	22	15
	Weight	0.76	0.567	2.241
	Causality	14.44	12.481	33.614
	Network Connectivity	7.595	6.683	2.437
	DCR	109.672	83.414	81.933
Wheel	No. of Nodes	10	10	10
	No. of Parent	16	18	11
	Weight	4.592	2.762	1.843
	Causality	73.479	49.716	20.28
	Network Connectivity	4.693	7.254	2.901
	DCR	344.877	360.621	58.841
Fuel nozzle	No. of Nodes	30	30	30

	No. of Parent	30	73	48
	Weight	6.545	10.646	10.121
	Causality	196.35	777.158	511.008
	Network Connectivity	3.121	12.485	5.706
	DCR	612.853	9702.75	2916.061

6.5.3 Implementation: Knowledge Network Optimality Evaluation System (KNOES)

This work presents a new knowledge support system, called Knowledge Network Optimality Evaluation System (KNOES), for future CAD applications in product development. The implementation of this system will be presented with a valve design case scenario in this chapter. The system is developed with C++, C#, IIS, and MS SQL. C++ conducts the main function, C# is used for the web application, IIS is Internet information service, and MS SQL is for database. Currently, KNOES operates as a stand-alone web application. CAD and KNOES can communicate through a common knowledge network interface. Currently an .xdsI format is used. To fully use this system, these different applications (i.e., CAD system, KNOES, and GeNIe) need to be set up.

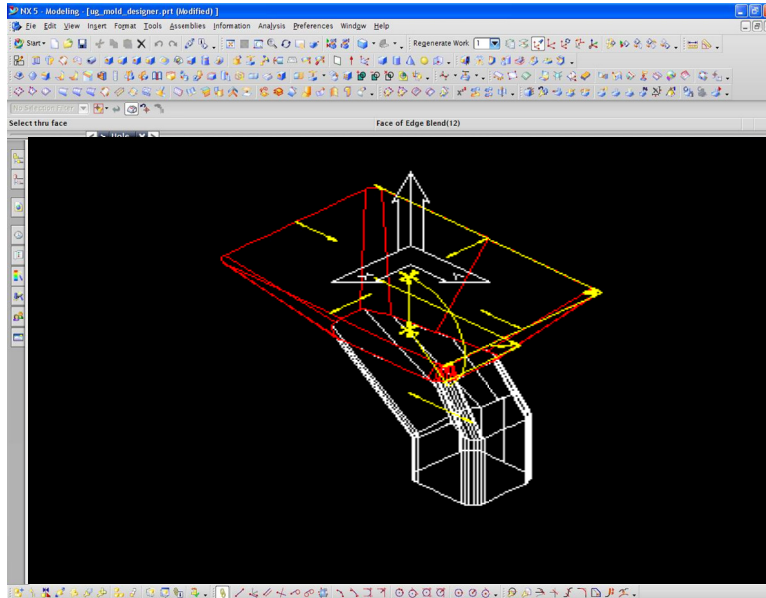


Figure 6-11 Example of UGS NX5 valve design

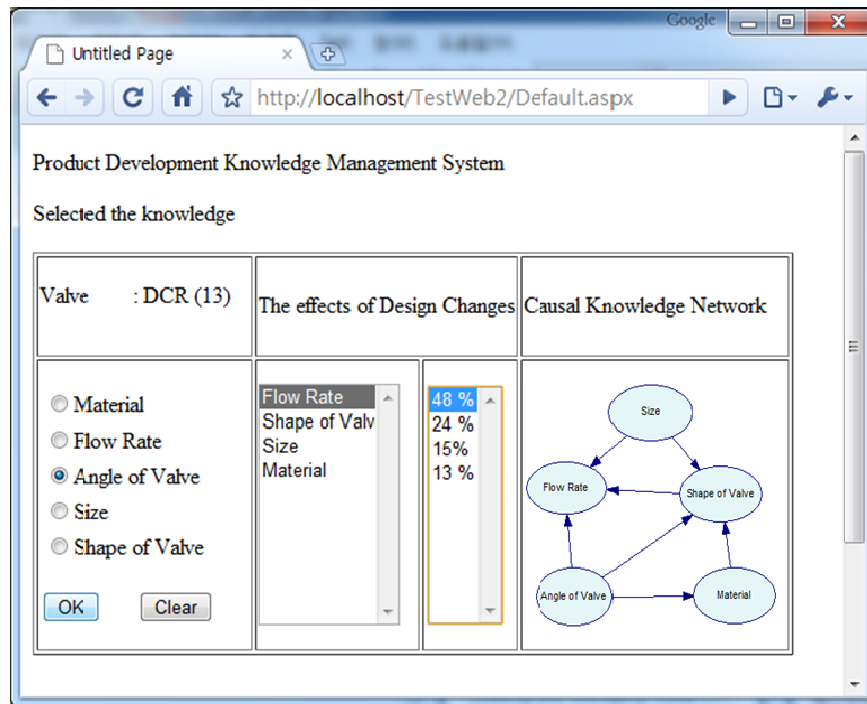


Figure 6-12 Web-based causal design knowledge evaluation and support system

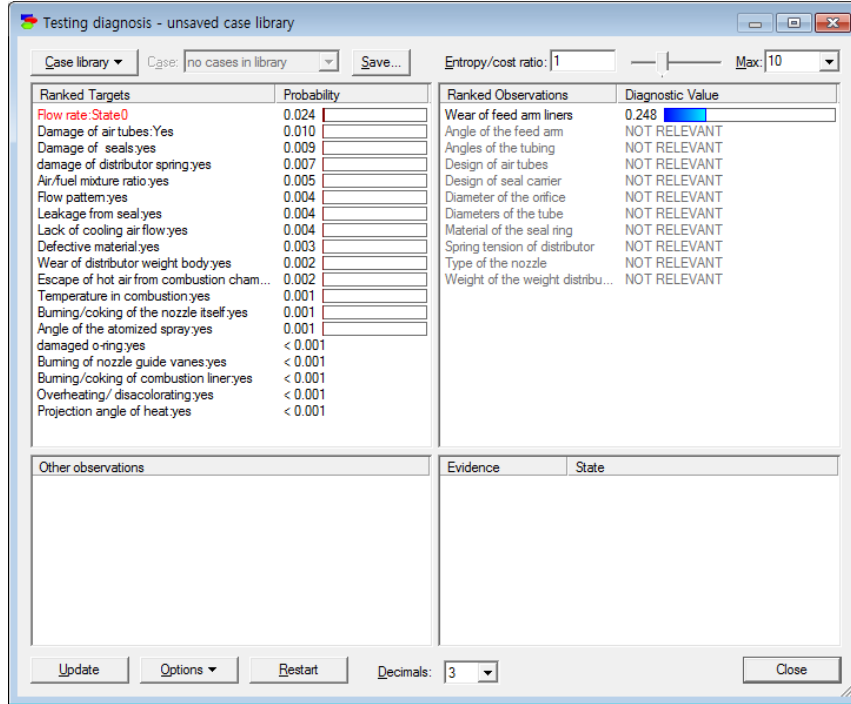


Figure 6-13 Example of the effects of the design modification

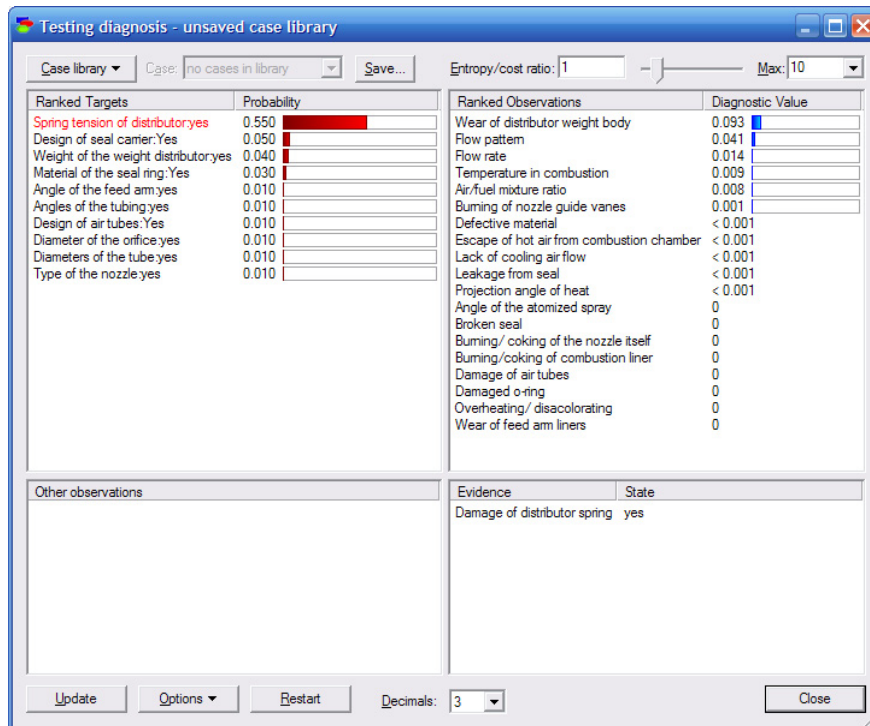


Figure 6-14 Example of the design factors from maintenance issues

In this case example, a scenario is presented between the design state of a new valve and its maintenance stage. A designer wants to create the new valve based on an existing valve design in the product design knowledge-base. First, the designer opens a CAD application, such as UGS NX 5, which is used for this scenario, and searches for a valve design from product design knowledge-base. The designer loads a valve design on the NX 5 as shown in Figure 6-11. KNOES provides the design analysis and evaluation results for this existing design as shown in Figure 6-12. The designer is easily able to understand the existing design and plans the modification, because the provided result has comprehensive causal knowledge of the design, such as DCR, knowledge network, the effects of factor's changes, and more. Second, the designer wants to modify this design for an appropriate design of the new valve. The designer increases the angle of the valve. KNOES notices the effects of this modification as shown in Figure 6-13. The effects of the modification include not only design factors but also maintenance factors for preventing future potential failures. Therefore, the designer is able to imagine and understand any impacts of potential changes made to the current design. Third, the designer completes the new valve design and sends it to a manufacturer to make the product. The manufacturer makes the new valves, which is very popular at the market. However, this valve has some maintenance issues when it gets to customers. The designer and manufacturer want to identify what are the problems. In this situation, KNOES provides the design factors relating to maintenance issues as shown in Figure 6-14. The inputs of maintenance factors will generate the

outputs, which are design factors using causal knowledge reasoning in KNOES. Therefore, the designer is able to redesign the valve appropriately. The new valve design is stored in product design-base and the knowledge of these design and maintenance issues are stored in product design knowledge-base for the future reuse.

As the iteration of the product development is increased, the knowledge from the product development is accumulated and improved. The knowledge accumulation is an important issue, but the quality of knowledge is more significant for KNOES. The knowledge in the evaluation system can be thoroughly obtained from the domain expert's knowledge acquisition to the DCR calculation. To obtain quality DCR-analysis results, the quality of the domain expert's knowledge must be maintained. The collective knowledge is a significant issue in terms of knowledge quality. Recently, the expended definition of knowledge is required in an advanced Internet environment. Berger and Luckman defined that knowledge is all ideas that are included when a society or social group believes that a thing exists [Berger 1996]. The definition of knowledge has evolved to the whole of the collected experiences in the society. Not only the knowledge that some experts can understand or generate is knowledge, but also, one, which is understood and generated by a variety member of the society in their daily experience, is knowledge [McCarty 1996]. Objectified knowledge is a formal and advanced process to collect and share the distributed knowledge and intelligence [Nahapiet 1998]. Objectified knowledge is growing in cyberspace. The spread of Web 2.0 is

leading the changes of society and economy in order to collect and share that objectified knowledge. The biggest change is that a user becomes a generator of knowledge. Wikipedia is a symbolic service of the collective intelligence to collaborate and share knowledge, and it is exponentially growing. Users of this service expect that the trusted knowledge is generated, shared, and utilized. There is a question about this knowledge quality in collective knowledge because the collective knowledge can be edited by anyone. However, *Nature* noted that Wikipedia comes close to encyclopedia Britannica in terms of the accuracy of its science entries [Wales 2005]. This supports the quality of collective knowledge. Furthermore, the methods of keeping knowledge quality in collective knowledge are utilized, such as using real name, reviewing by experts, opening editing processes, and levels of members. In the future, the quality effect of collective knowledge in causal knowledge management will be investigated.

6.5 Conclusion

In this chapter, this research presents a system of causal design knowledge evaluation and support, and a new causal knowledge evaluation method is developed and implemented. This new causal knowledge evaluation method compares design knowledge using degree of causal representation. The results show that: 1) the more complex network model has higher DCR, 2) the network with higher weight has higher DCR, 3) the effect of weight increases with the more complex network. Next, we presented a causal knowledge evaluation system and its validation by comparing causal knowledge through three realistic cases:

assembly design, wheel, and fuel nozzle. Finally, this research presents an implementation of a causal design knowledge evaluation support system, called KNOES, with a new valve design case, and reviewed the advantages and disadvantages of the new product design knowledge support system. Using KNOES, a designer is able to obtain knowledge analysis and evaluation, the effects of any design change, and sensitivity analysis for future potential failures. For future research, an extension of the new causal knowledge evaluation method is required for the more complex knowledge models.

CHAPTER 7

DCR INDEX AND KNOWLEDGE INTEGRATION

The aim of this chapter represents DCR index and knowledge integration for causal product design knowledge. First, DCR index is for comparison of multiple causal knowledge with different number of vertices. DCR is strongly dependant to number of vertices. The more number of vertices have the more DCR. DCR index utilizes a normalization method for comparison of causal product knowledge with different number of vertices. Second, knowledge integration is required for obtaining a new knowledge from existing knowledge. For example, a user requests knowledge for the heating cup. However, the knowledge base only has knowledge for heating and cup, not heating cup. At his situation, knowledge integration can generate a new heating cup knowledge from existing heating and cup knowledge. Therefore, the user can obtain the knowledge for heating cup.

7.1 DCR Index

This chapter represents how to utilize DCR to compare multiple causal knowledge. The DCR is strongly dependant to the number of vertices in the causal knowledge network. DCR is conducted with two parts, connectivity and probability, as presented in Chapter 6. The connectivity has more effect than the probability to calculate DCR. The number of vertices in the causal knowledge network is most effected parameter for calculating DCR. For example, comparing three knowledge networks with different number of vertices (3, 6, 10). The connectivity is maximum and probability is 0.99 (Table 7-1). Depending on the number of vertices, DCR is

34.5744, 12074.7334, and 115732072.2365. Since the DCRs are significantly different, multiple causal knowledge is not able to compare with DCR. Therefore, there is need for DCR index to compare multiple causal knowledge.

To develop DCR index, two cases should be defined (Figure 7-1), minimum and maximum of each number of vertex. Each case has number of vertices, connectivity, and probability. For the minimum case, connectivity is minimum (only one connection between vertices) and probability is 0.51, which is lowest because there is no effect on 0.5 and 1 is full effect. For the maximum case, connectivity is maximum (every vertices are connected) and probability is 0.99, which is highest in this research.

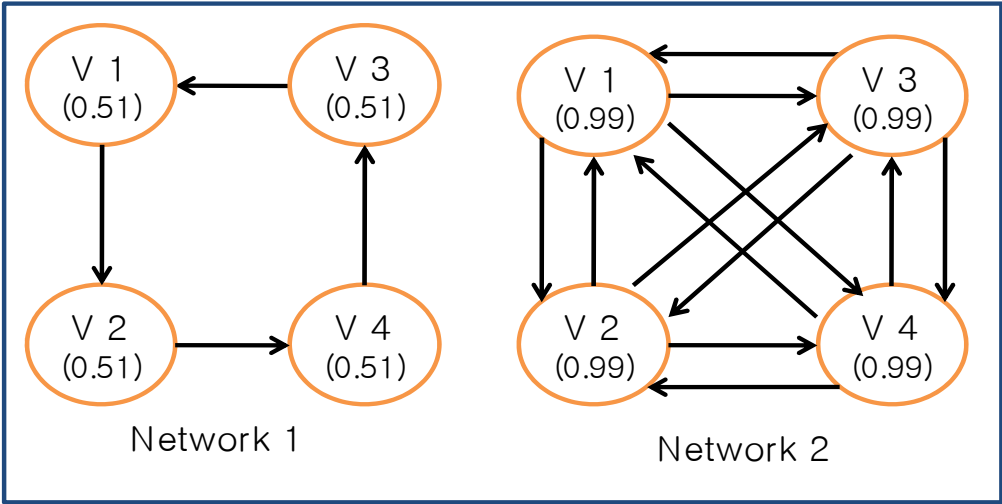


Figure 7-1 Examples of network for DCR

The DCR index is generated with a normalization method as show in below.

Definition DCR index

$$\text{Index } (I) = (\bar{X} - \bar{M}) \times 100 / (X - \bar{M})$$

Where \mathcal{A} is actual DCR, \mathcal{M} is min. DCR, \mathcal{X} is max. DCR.

Figure 7-2 shows the meaning of DCR index. DCR index normalizes multiple causal knowledge to single index for comparison. Each of causal knowledge has minimum DCR, middle DCR, and maximum DCR. However, increasing number of vertices, the middle DCR and maximum DCR are exponentially increased. We cannot compare the knowledge with different numbers of vertices (e.g., numbers of vertices are 4 and 6) because DCR is strongly depended on number of vertices as shown in Figure 7-2. After DCR indexing, one single DCR index can represent multiple knowledge's DCR levels. Using this DCR index, this research can compare the knowledge with different numbers of vertices. Currently, DCR index is conducted for the knowledge with numbers of vertices from three to eleven. The detail result is showing in Table 7-1. Each case has minimum DCR (index is 0) and maximum DCR (index is 100). The maximum DCR is confirmed the limitation of DCR, which is strongly depended on numbers of vertices.

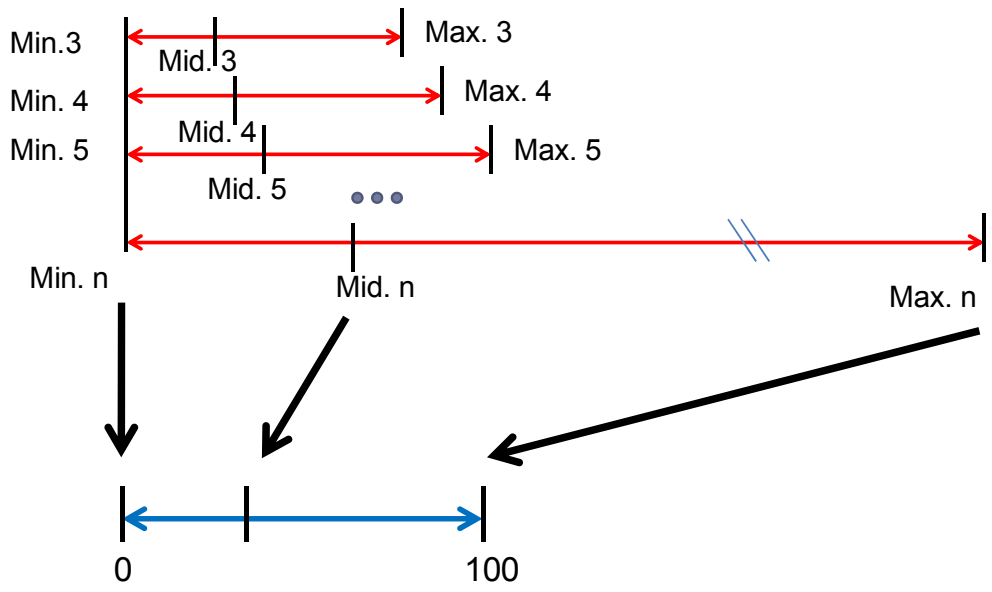


Figure 7-2 DCR indexing process

Table 7-1 Result of DCR index with vertices 3 to 11

ID	# of Nodes	Connectivity	Probability	DCR	Index
1	3	Minimum	0.51	0.0036	0
	3	Maximum	0.99	34.5744	100
2	4	Minimum	0.51	0.0077	0
	4	Maximum	0.99	262.7654	100
3	5	Minimum	0.51	0.0154	0
	5	Maximum	0.99	1728.7200	100
4	6	Minimum	0.51	0.0282	0
	6	Maximum	0.99	12074.7334	100
5	7	Minimum	0.51	0.0473	0
	7	Maximum	0.99	98049.6898	100
6	8	Minimum	0.51	0.0738	0
	8	Maximum	0.99	981411.6411	100
7	9	Minimum	0.51	0.1088	0
	9	Maximum	0.99	10184059.6366	100
8	10	Minimum	0.51	0.1536	0
	10	Maximum	0.99	115732072.2365	100
9	11	Minimum	0.51	0.2093	0
	11	Maximum	0.99	1427091751.6989	100

The test of DCR index is conducted with two networks as shown in Figure 7-3. The test is conducted with the knowledge with five vertices network. First, evaluate five vertices network with DCR. The first three row of Table 7-2 shows the result of evaluation with three different conditions based on connectivity and probability. After this evaluation, increase one vertex and one edge on the five vertices network and evaluate it. Then, increase one edge each time with the same condition of probability (e.g., 0.51, 0.75, 0.99). The detail result is shown in Table 7-2. One interesting finding is confirmed that the DCR normalization is conducted correctly. The remark 1 and 2 shows the same DCR index (26.030) with different original DCR (450, 3143.1522). If we compare the five vertices knowledge and six vertices knowledge with maximum connectivity and 0.75 probability, the DCR index is the same, which means these two knowledge represent the same level of causal representation.

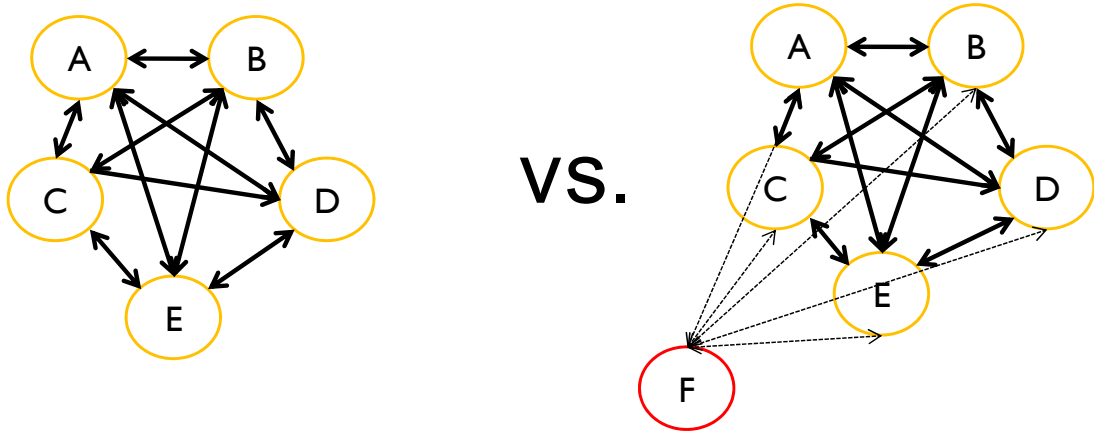


Figure 7-3 Example of DCR index test

Table 7-2 Result of DCR index test

# of Nodes	Connectivity	Probability	DCR	Index	Remark
5	Minimum	0.51	0.0154	0	
5	Maximum	0.75	450.0000	26.030	1
5	Maximum	0.99	1728.7200	100	
6	Max.+One	0.51	3.4591	0.028	
6	Max.+Two	0.51	3.8246	0.031	
6	Max.+Three	0.51	4.2029	0.035	
6	Max.+Four	0.51	4.6072	0.039	
6	Max.+One	0.75	2161.9565	17.905	
6	Max.+Two	0.75	2390.3804	19.796	
6	Max.+Three	0.75	2626.8261	21.755	
6	Max.+Four	0.75	2879.5109	23.847	
6	Max.+Five	0.75	3143.1522	26.030	2
6	Max.+One	0.99	8305.3722	68.783	
6	Max.+Two	0.99	9182.8855	76.050	
6	Max.+Three	0.99	10091.2151	83.573	
6	Max.+Four	0.99	11061.9290	91.612	

7.2 Knowledge Integration

To fully understand knowledge integration, the integration environments should be defined, such as knowledge framework, integration models, and other considerable factors. The first factor is knowledge framework. Current product development knowledge in product development processes is showing in Figure 7-4. Current product development knowledge framework cannot handle recursive product development knowledge since there is not enough method to capture knowledge in product development processes. To overcome this problem, new knowledge framework is required in order to handle recursive knowledge during the product development processes. Inter-relational product development

knowledge framework is proposed and it can handle recursive knowledge using causal network integration.

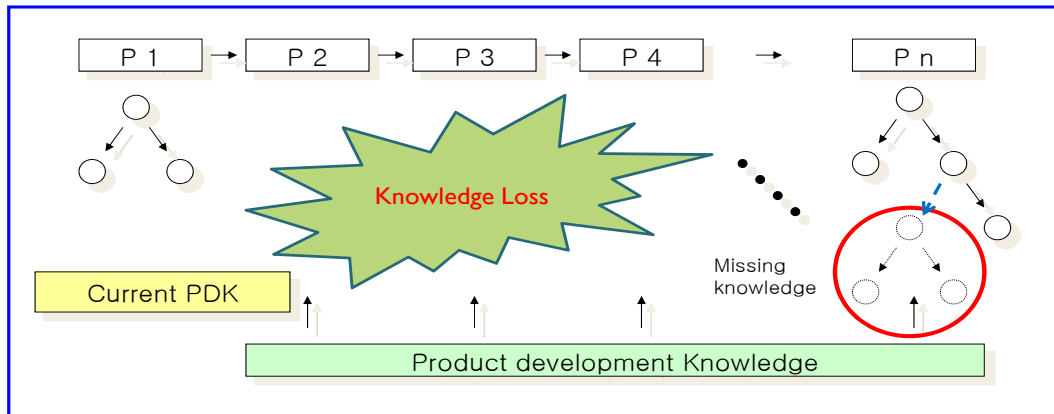


Figure 7-4 Current product development knowledge acquisition and loss

The product development knowledge can be represented with three different knowledge models, which are based on perspective of knowledge relationships. Knowledge relationships include three categories for inter-relational knowledge framework: inter-process knowledge, inter-actor knowledge, and inter-product knowledge (Figure 7-5). First, inter-actor knowledge acquires and reuses the same domain knowledge with different actors (designers, systems, and etc.) using causal network belief integration. Second, inter-process knowledge acquires and reuses different domains knowledge, which has different constraints for each domain, using causal network structure integration method during the product development processes. Third, inter-product knowledge acquires and reuses different domains knowledge and different products knowledge using causal network belief integration between different structures.

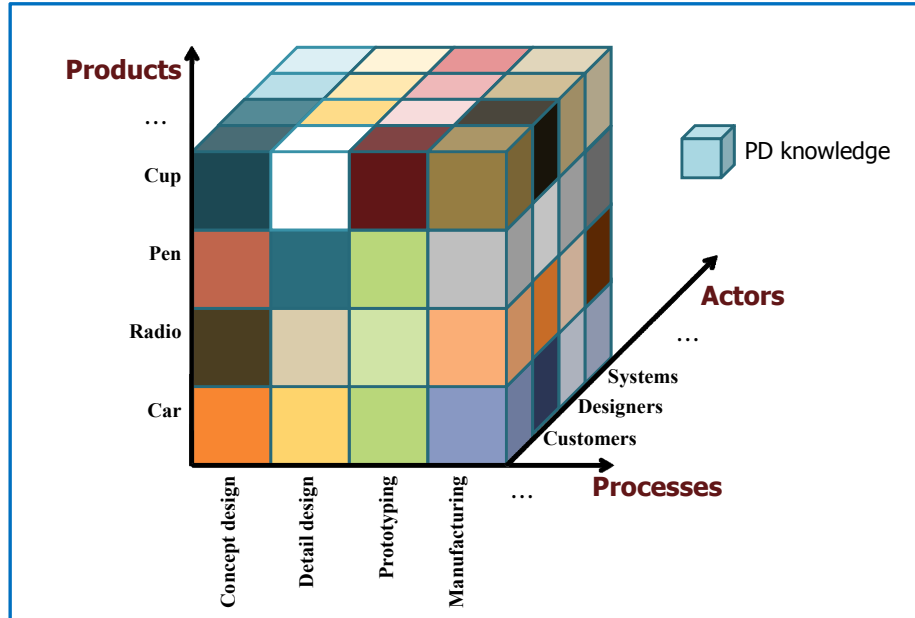


Figure 7-5 Knowledge relationship for product development.

7.2.1 Inter-actor Knowledge

Inter-actor knowledge acquires and reuses the same domain knowledge with different actors (designers, systems, and etc.) using causal network integration (Figure 7-6). In this case, one basic assumption is that a causal network structure is the same in each domain. Inter-actor knowledge framework integrates actors' knowledge in the same domain, because different actors have different knowledge with the same process and product in domain. Main function of this framework is that the accuracy of the knowledge is improved by knowledge integration with weights, which are based on experiences, positions, number of same project completion, and other considerable factors.

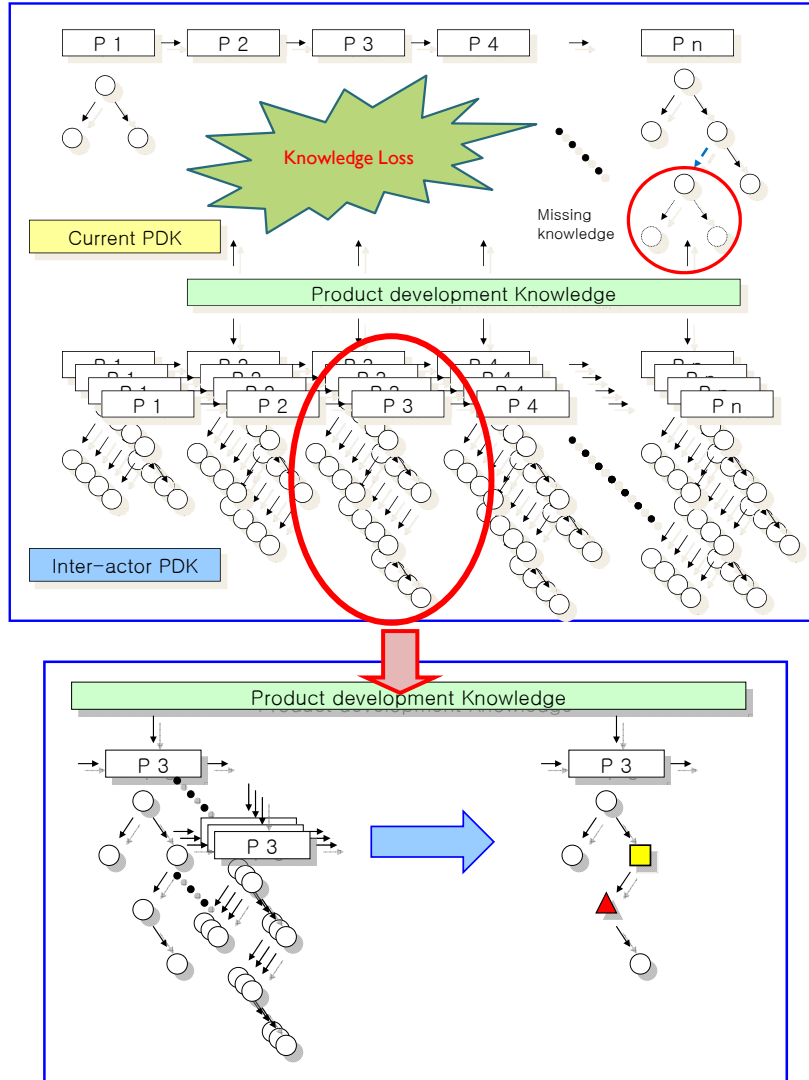


Figure 7-6 Inter-actor knowledge integration for product development

In the Figure 7-7, inter-actor knowledge framework is integrated one single framework, which is the same of the inter-process knowledge framework. Also, this single framework will be able to use for inter-process knowledge framework.

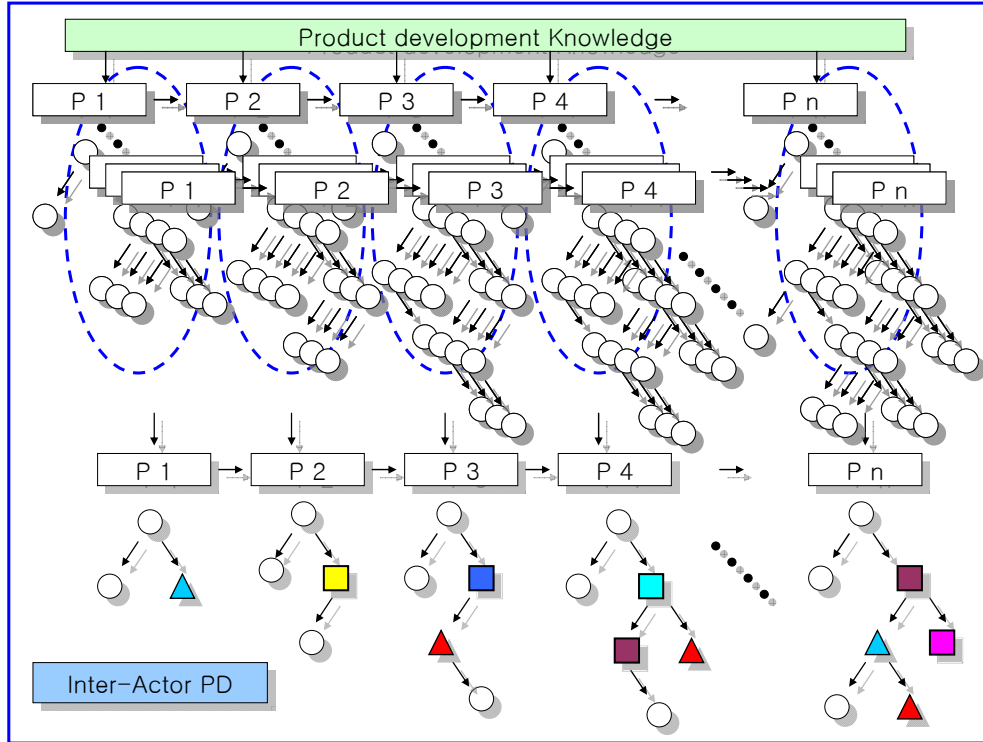


Figure 7-7 Inter-actor knowledge framework for product development

The inter-actor knowledge framework can acquire and reuse the same domain knowledge with different actors (designers, systems, etc.) using causal network integration during the product development processes. Finally, I conduct inter-actor knowledge framework and product development knowledge for this framework for single product. The next step is applying this result to inter-process knowledge framework, which include heterogeneous domains.

7.2.2 Inter-process Knowledge

Inter-process knowledge framework acquires and reuses different domains knowledge, which has different constraints for each domain, using causal network structure integration method during the product development processes. Inter-

process knowledge framework is showing in Figure 7-8. Compared with current product development knowledge, inter-process knowledge framework is evolutionary increasing the knowledge process-by-process. Current knowledge framework has loosed product development knowledge during the product development processes (Figure 7-9).

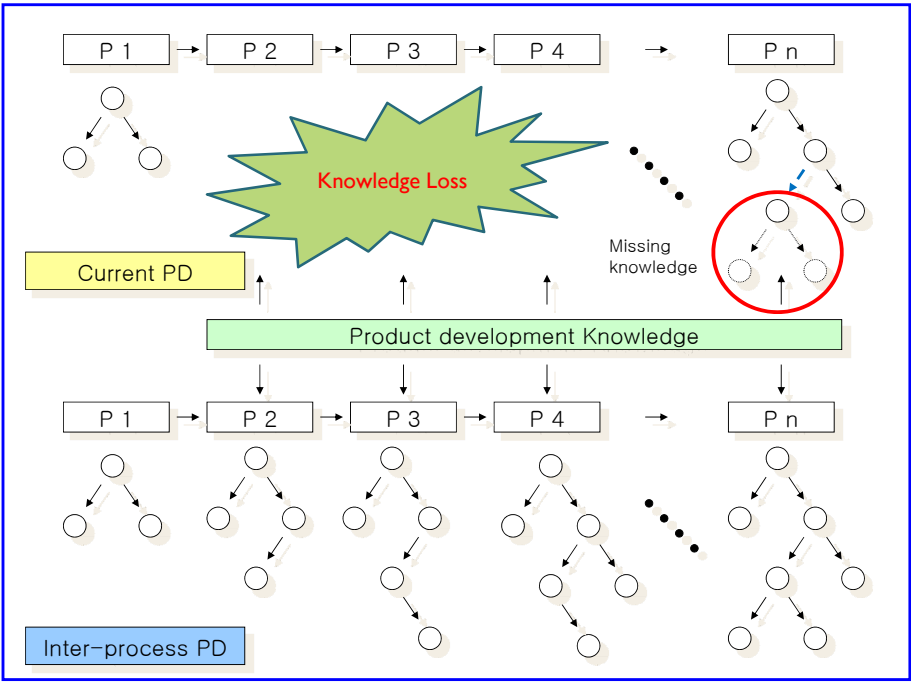


Figure 7-8 Inter-process knowledge framework vs. current knowledge framework for product development

For example, a product development has six processes: Detailed Requirements, Conceptual Development, System-level Design, Detail Design, Testing and Refinement, and Production Ramp-up. Each of process has different constraints. These constraints affect product development knowledge to add or delete knowledge to apply constraints. Because conceptual development process

does not require a detail design aspect, the causal network structure of conceptual development process is slightly smaller than detail design process's one. Process is moved from system-level design process to detail design process. The knowledge from system-level design to detail design will be added, deleted, or updated, which means causal network structure will be added, deleted, or updated. Finally, production ramp-up process will have more informative causal network structure than any other processes. Via current framework, production ramp-up process has almost the same causal network structure with other processes. Therefore, the result of production in inter-process knowledge framework will be significantly improved.

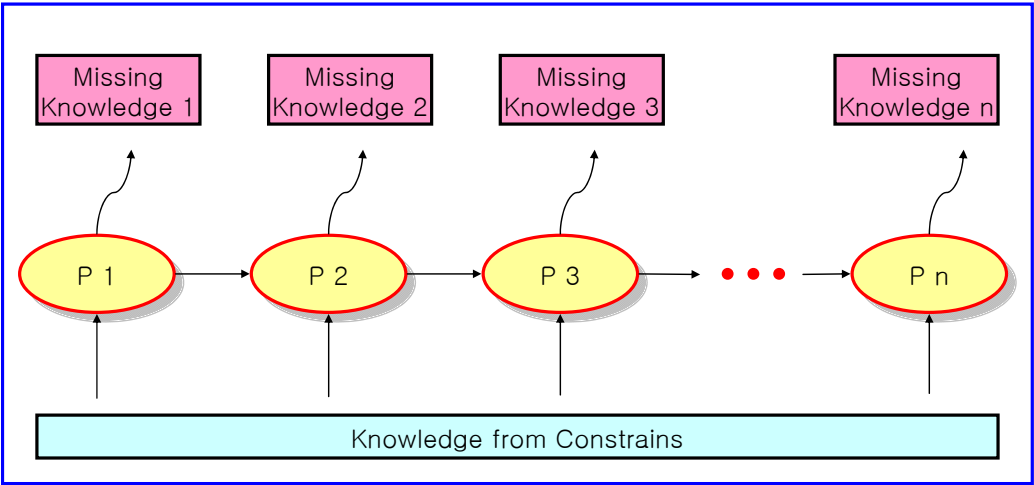


Figure 7-9 Missing knowledge in current product development knowledge framework

The inter-process knowledge framework can acquire and reuse the different domains knowledge, which has different constraints for each domain, using causal network structure update method during the product development processes.

Finally, we conduct inter-actor knowledge framework and inter-process knowledge framework, and product development knowledge for these frameworks for single product. The next step is applying this result to inter-product knowledge framework, which include heterogeneous products.

7.2.3 Inter-product Knowledge

Inter-product knowledge framework acquires and reuses different domains knowledge and different products knowledge using causal network and structure integration between different structures. The Figure 7-10 is showing inter-product knowledge framework, which integrates heterogeneous products' knowledge to general knowledge. However, this framework is not visible because heterogeneous products do not have the same structures, even not similar. If I integrate these heterogeneous products' knowledge, I will have huge general knowledge, which may not be represented by any network. Therefore, I propose unsupervised learning to categorize this heterogeneous products' knowledge to similar products' knowledge. First, classify this knowledge with similar products and then integrate this similar knowledge in categories (Figure 7-11).

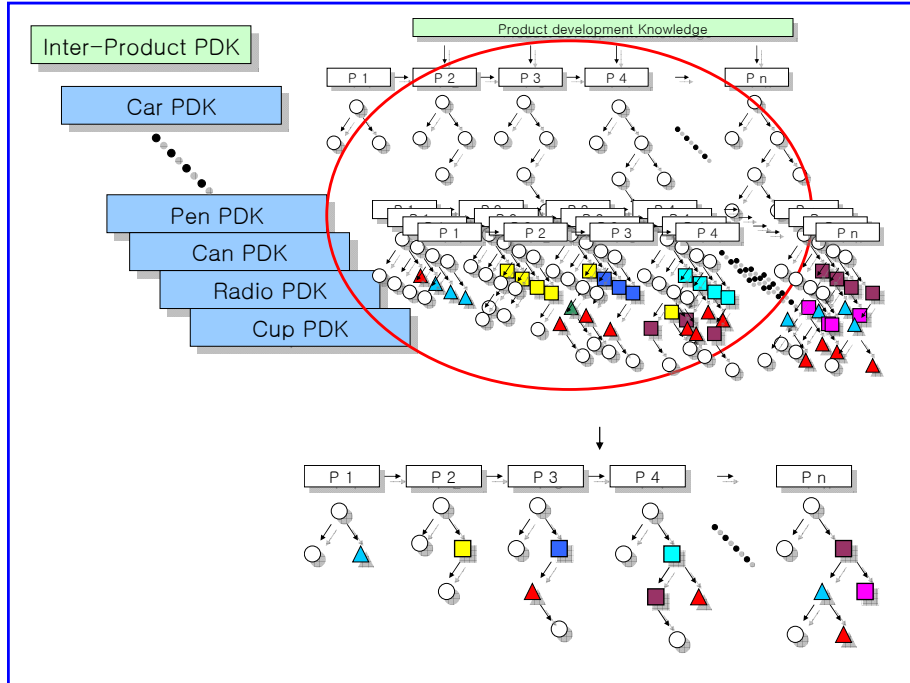


Figure 7-10 Heterogeneous product development knowledge framework

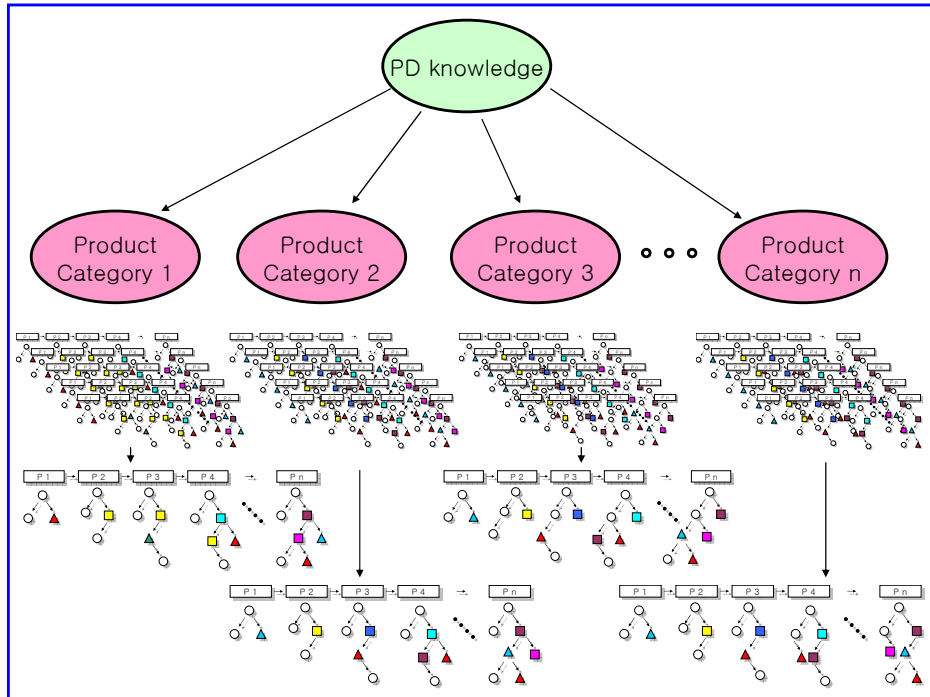


Figure 7-11 Inter-product knowledge framework with unsupervised learning

The inter-product knowledge framework can acquire and reuse different domains knowledge and different products knowledge using causal network and structure integration between different structures during the product development processes. Finally, we conduct inter-product knowledge framework using unsupervised learning for the categorization of the similar products and product development knowledge for these frameworks for heterogeneous products. The next chapter represents the causal network integration with inter-relational knowledge framework for the recursive product development knowledge in product development processes.

7.2.4 Integration of Causal Knowledge

Knowledge integration is an intelligent knowledge acquisition method from existing knowledge. Based on inter-relational knowledge framework, knowledge integration includes three different cases (Figure 7-12). Case 1 is from inter-actor knowledge framework and is only belief integration in the same structure. Case 2 is from inter-process knowledge framework and is added the knowledge structures for integration and is updated belief between the structures. Case 3 is from inter-process knowledge and inter-product knowledge framework and integrates the knowledge structures and is integrated the knowledge structures and belief. The combination of these three cases can cover all possible integration cases in product development knowledge.

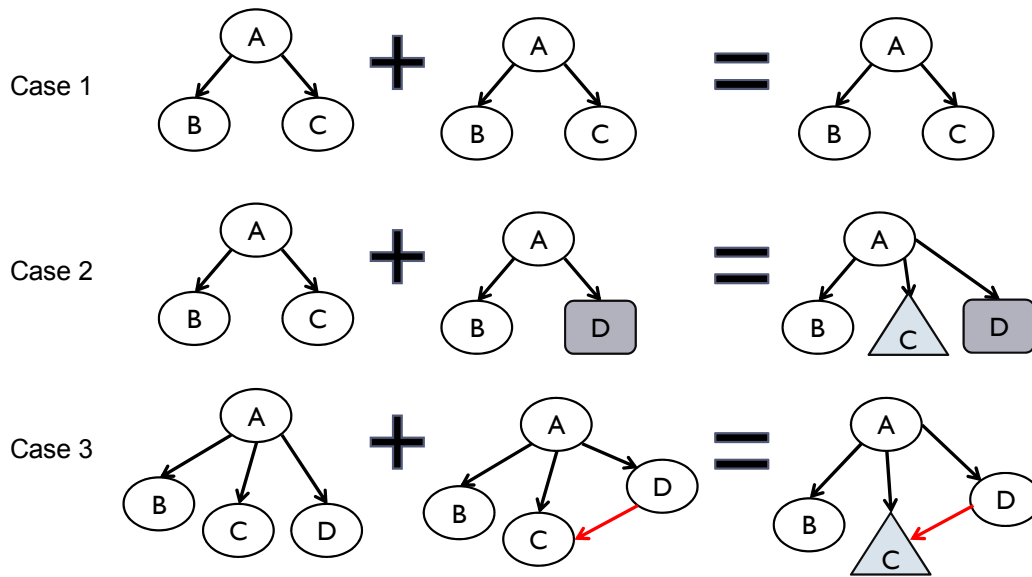


Figure 7-12 Knowledge integration cases

Based on knowledge integration cases, two main function is required, knowledge network identifier and integrator. The network identifier analyzes number of vertices, matching of vertices, structure of the knowledge network, and other considerable factors in knowledge. For the matching of vertices' name, ontological knowledge mapping, which will be addressed in section 7.2.4.1, can be used. After analyzing the knowledge, network identifier can select the combination of knowledge integration cases (Figure 7-12). Based on the selected combination cases for knowledge integration, knowledge network integrator generates a new knowledge using structure and probability integration.

7.2.4.1 Ontological Knowledge Mapping

To integrate heterogeneous design knowledge, which includes different knowledge name for the same knowledge, this research proposes the ontological causal network representation to match different knowledge name. The ontology is one of the ways to represent product development knowledge as mentioned in Chapter 3.5. Ontology is explicit formal specifications of the terms in the domain and relations among them [Gruber 1993]; a formal, explicit specification of a shared conceptualization. This research uses Bayesian belief network and Ontology to represent product design knowledge, which means the network is BBN and the nodes are defined by ontology (Figure 7-13).

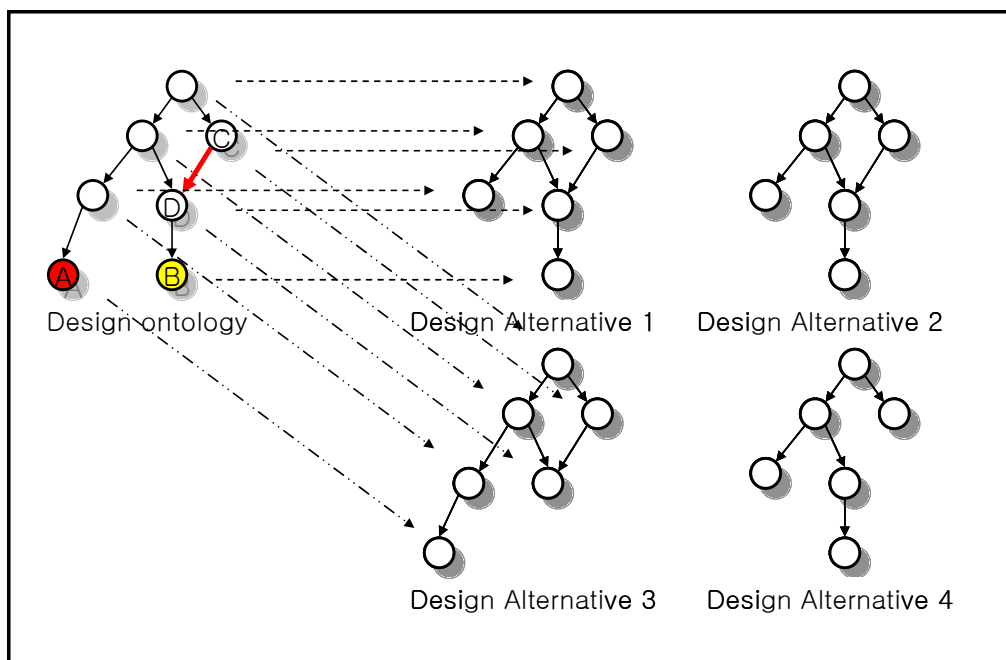


Figure 7-13 Ontological BBN design knowledge

In the Figure 7-13, the design alternatives are somewhat different, but the design ontology is the same for the specific product or part. Design alternative 1 inherits all of the nodes except node A. Design alternative 2 is the same with design alternative 1, but the probability of nodes are may vary. Design alternative 3 inherits all of the nodes except node B. Design alternative 4 is almost same with design alternative 1 and 2 except the arc from node C to node D. The ontological BBN is able to use for product development knowledge reasoning and mapping.

7.2.5 Utilization of Causal Knowledge Integration

Causal knowledge integration method is utilized for a new product design using existing one. For example, a cup with heating/cooling function, a pen with special logo on it, and a can with zip lock. These examples are modified existing design or are generated by knowledge integration. In this chapter, the utilization of causal knowledge integration is presented with a wheel design scenario. A designer want to design a new designed automotive wheel, which is modified from existing design, but current design knowledge is not enough to design the new wheel. Thus, an appropriate design knowledge should be generated from existing one. In other words, the design knowledge integration is required. The designer opens web-based causal product design knowledge management system in order to find existing wheel designs. In Figure 7-14 shows existing design in the system repository. Also, the knowledge evaluation results are provided for support designer's decision as shown in Figure7-14's table right side (DCR and DRC index).

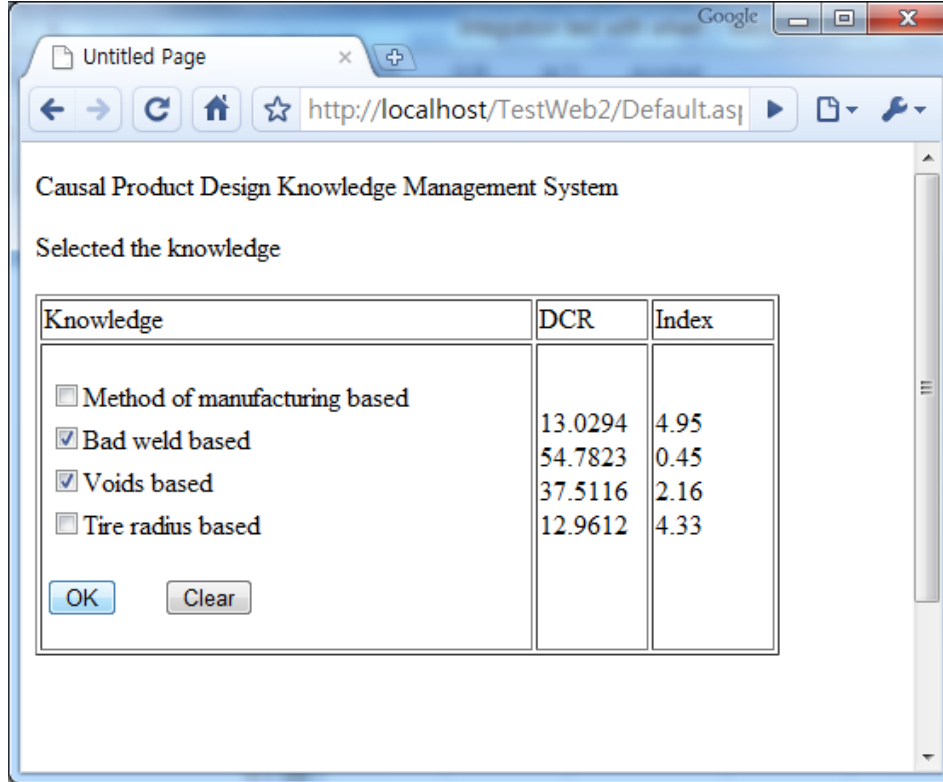


Figure 7-14 Snapshot of causal product design knowledge management system for knowledge integration

Among the existing designs, the designer selects two alternatives for the new wheel design, Bad weld based and voids based knowledge. The selected alternatives are integrated to generate a new design knowledge, which is named newNetwork-wheel. The integration result is presented in Figure 7-15. The result of integration provides the new design knowledge with DCR evaluation result, which includes DCR and DCR index (Figure 7-15). Finally, the design can select the new integrated knowledge, which can provide knowledge of two original ones for the new wheel design.

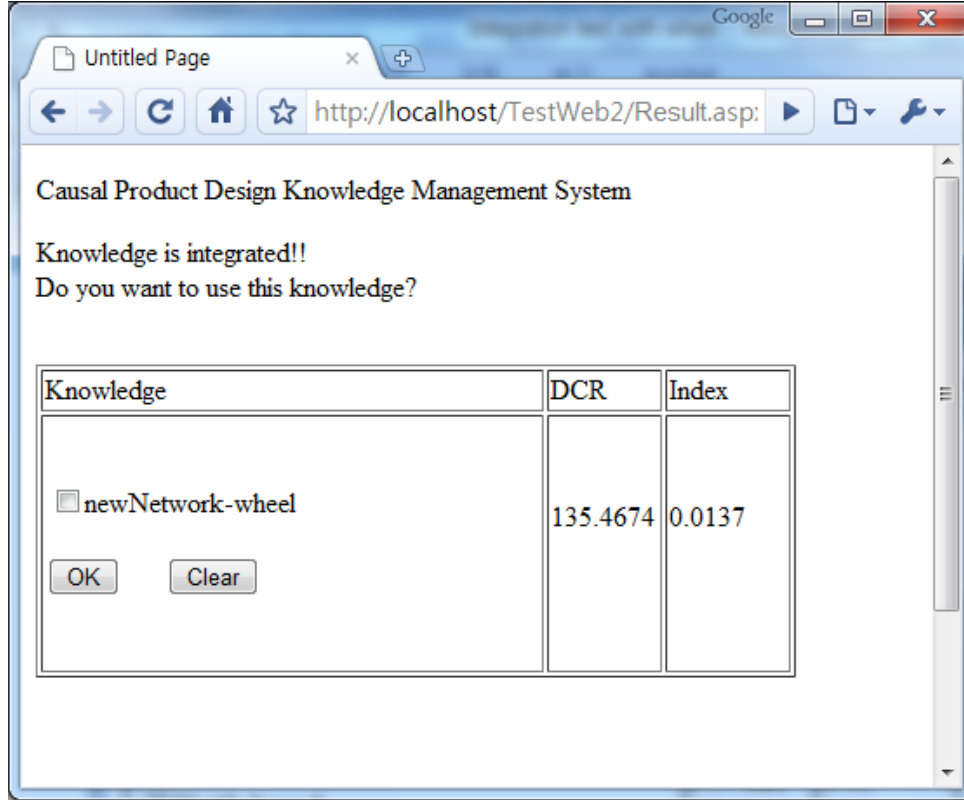
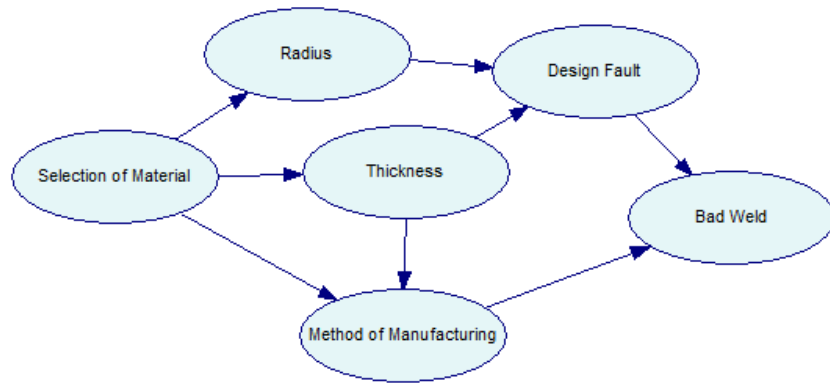
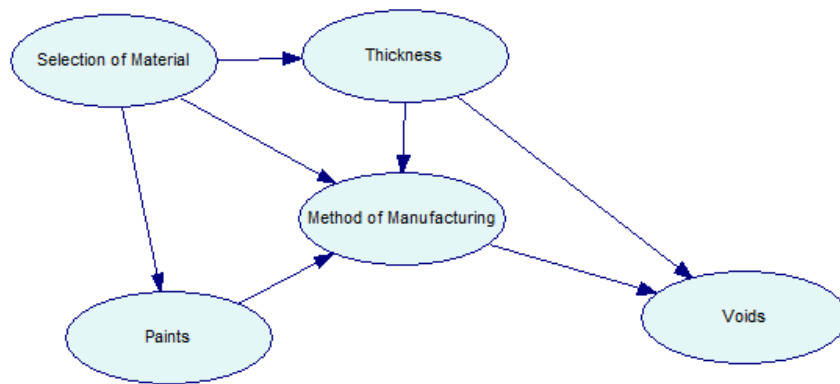


Figure 7-15 Result of knowledge integration with DCR evaluation

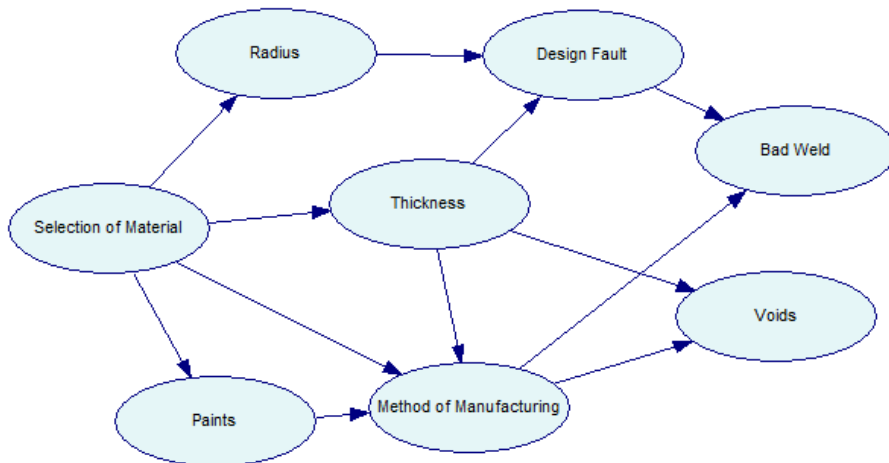
The original knowledge networks for Bad weld based and Voids based wheel knowledge are illustrated in Figure 7-16 (a) and (b). The analysis of these knowledge networks are as follows: for Bad weld based knowledge, DCR is 54.7822, causality is 23.3331, network connectivity is 2.3178, and DCR index is 0.45; for Voids based knowledge, DCR is 37.5116, causality is 16.255, network connectivity is 2.3076, and DCR index is 2.16.



(a) Knowledge network for Bad weld based



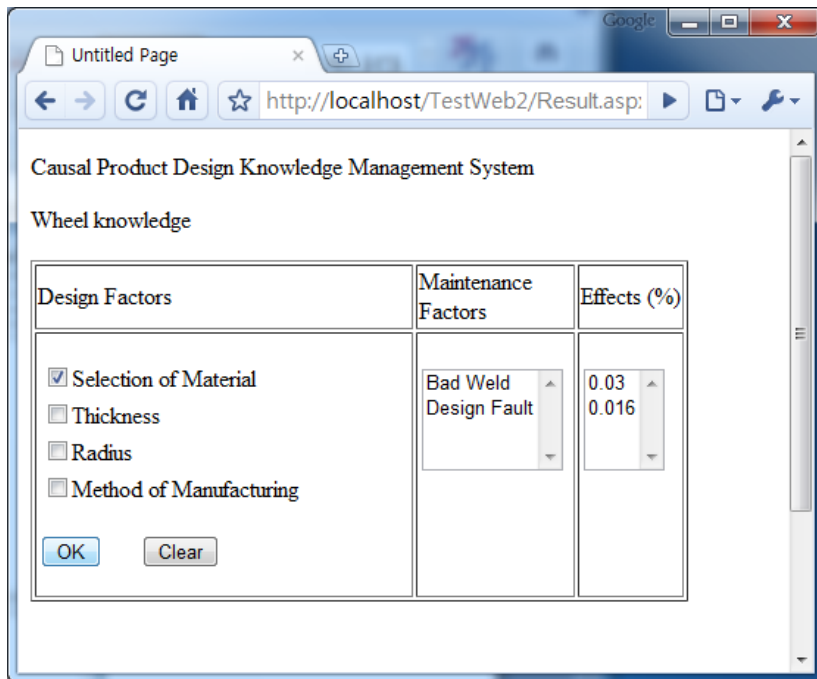
(b) Knowledge network for Voids based



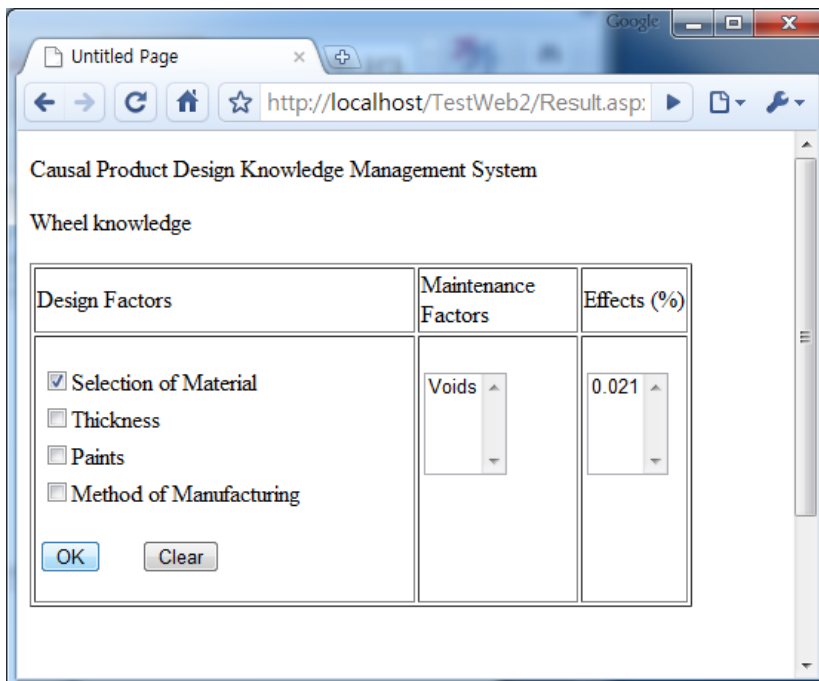
(c) Knowledge network for newNetwork-wheel

Figure 7-16 Knowledge network for integration with the same domain

DCR of bad weld based knowledge is higher than voids based knowledge, but the DCR index shows that voids based knowledge is higher than Bad weld based knowledge. It is explained that voids based knowledge has more degree of causal representation than the other one. Figure 7-16 (c) shows a new generated knowledge, newNetwork-wheel, which is analyzed that DCR is 135.4674, causality is 45.3893, network connectivity is 2.9845, and DCR index is 0.0137. This generated knowledge can represent two knowledge with updated their beliefs for wheel design. Based on the integrated knowledge, causal product design knowledge management system can provide the effects of the design modification as discussed in Chapter 6.4. Figure 7-17 presents the effects of design modification with two wheel knowledge, bad weld based and void based. if a designer modifies the *Selection of Material* among the design factors, the effects of maintenance factors are predicted. Two maintenance factors for bad weld based wheel knowledge are predicted, *Bad Weld* (0.03) and *Design Fault* (0.016). For the voids based wheel knowledge, *Voids* (0.021) is predicted. The numbers for maintenance factors represent the effects of the modification depended on each knowledge network structure and belief.



(a) Bad Weld based wheel knowledge



(b) Voids based wheel knowledge

Figure 7-17 Effects of design modification

Figure 7-18 illustrates the effects of design modification (*Selection of Material*) in the integrated wheel knowledge. The integrated wheel knowledge includes five design factors and three maintenance factors. Comparison with original knowledge and integrated knowledge indicates that 1) integrated knowledge provides more design factors to modify a design for a new product, 2) integrated knowledge predicts more maintenance factors to indicate the effects of the design modification, 3) integrated knowledge is enough to include the original knowledge in the representation of the effects of the modification. The integrated wheel knowledge provides more knowledge to a designer and the designer can make better decision for a new product design. This is one objective of using causal product design knowledge management system.

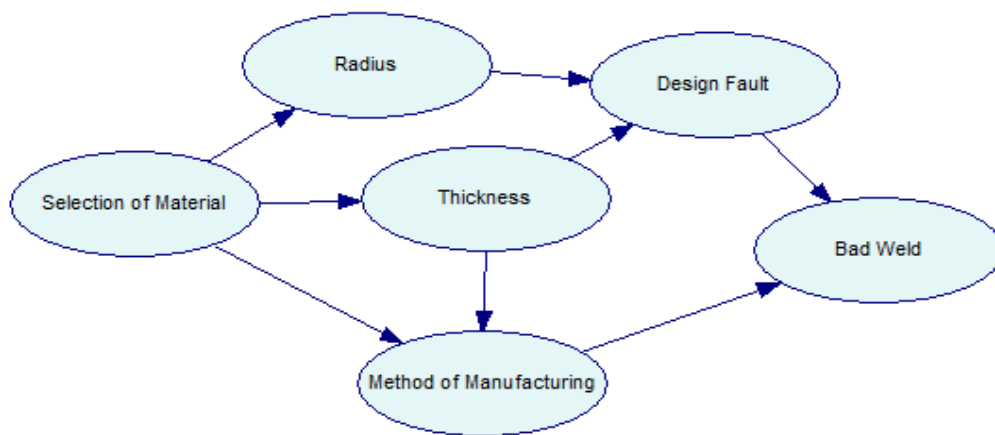
The screenshot shows a web browser window with the URL <http://localhost/TestWeb2/Result.asp>. The page title is "Causal Product Design Knowledge Management System". Under the heading "Wheel knowledge", there is a table with three columns: "Design Factors", "Maintenance Factors", and "Effects (%)".

Design Factors	Maintenance Factors	Effects (%)
<input type="checkbox"/> Radius <input type="checkbox"/> Thickness <input type="checkbox"/> Method of Manufacturing <input type="checkbox"/> Paints <input checked="" type="checkbox"/> Selection of Material	Design Fault Voids Bad Weld	0.03 0.021 0.014

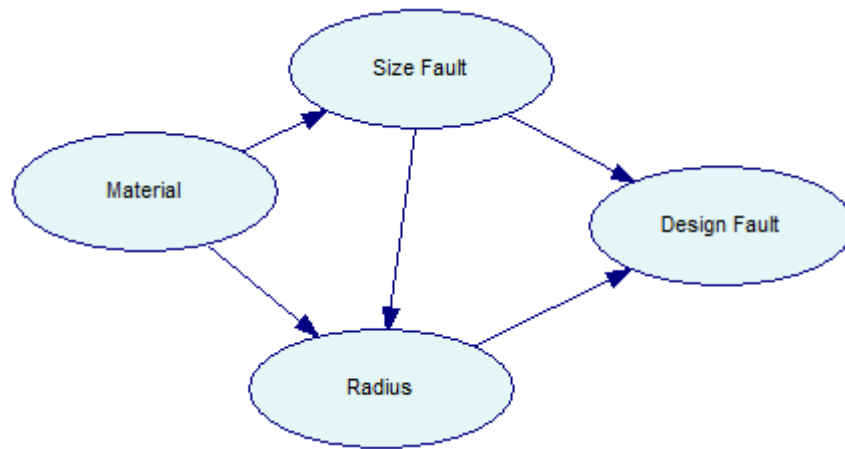
At the bottom of the table, there are two buttons: "OK" and "Clear".

Figure 7-18 Effects of design modification in integrated wheel knowledge

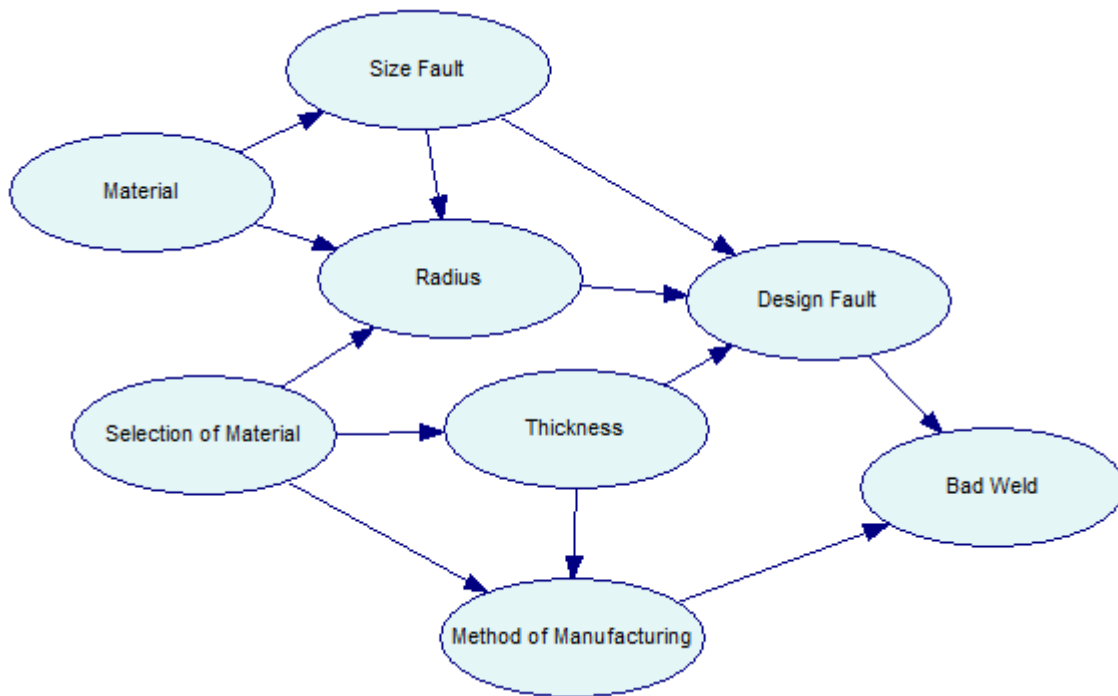
The utilization of knowledge integration in the same domain is presented and a utilization of knowledge integration in the different domain is following. Figure 7-19 (a) and (b) illustrate bad weld based wheel knowledge and tire knowledge, which are in different domains. Two of knowledge vertices for each knowledge are identical and others are totally different. The analysis of these knowledge networks are as follows: for bad weld based knowledge, DCR is 54.7822, causality is 23.3331, network connectivity is 2.3178, and DCR index is 0.45; for Tire knowledge, DCR is 15.4235, causality is 7.7007, network connectivity is 2, and DCR index is 5.8669. DCR of bad weld based knowledge is higher than tire knowledge, but the DCR index shows that tire knowledge is higher than bad weld based knowledge. It is explained that tire knowledge has more degree of causal representation than the other. The integrated wheel tire knowledge is presented in Figure 7-19 (c). It is integrated by knowledge network structure and beliefs. It shows a new generated knowledge, newNetowrk-wheel-tire, which is analyzed that DCR is 137.485, causality is 41.7467, network connectivity is 3.2933, and DCR index is 0.14.



(a) Bad weld based wheel knowledge



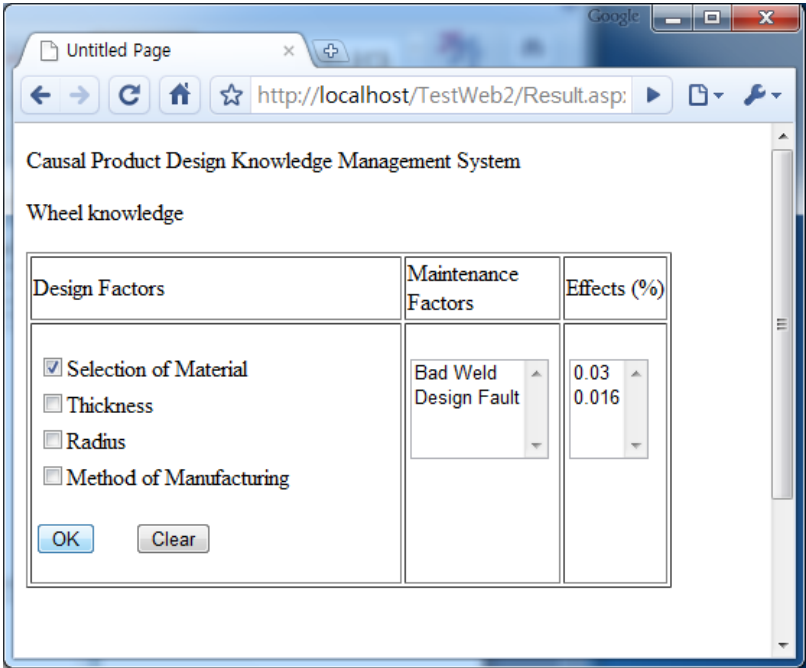
(b) Tire knowledge



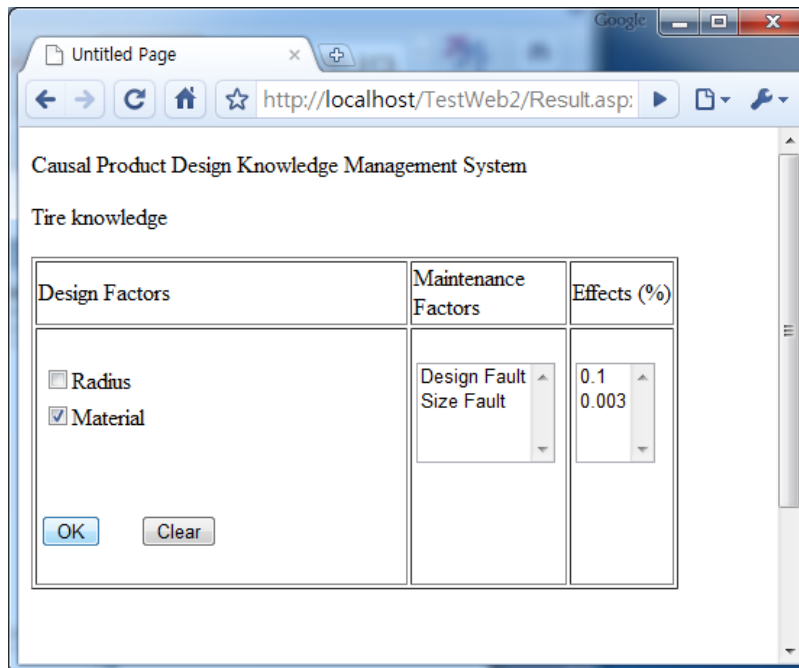
(c) Integrated wheel tire knowledge

Figure 7-19 Knowledge network for integration with different domains

Figure 7-20 presents the effects of design modification with two original knowledge with different domains, bad weld based wheel and tire knowledge. if a designer modifies the *Selection of Material* for bad weld based wheel knowledge and *Material* for tire knowledge among the design factors, the effects of maintenance factors are predicted. Two maintenance factors for bad weld based wheel knowledge are predicted, *Bad Weld* (0.03) and *Design Fault* (0.016). For the tire knowledge, *Design Fault* (0.1) and *Size Fault* (0.003) are predicted.



(a) Bad weld based wheel knowledge



(b) Tire knowledge

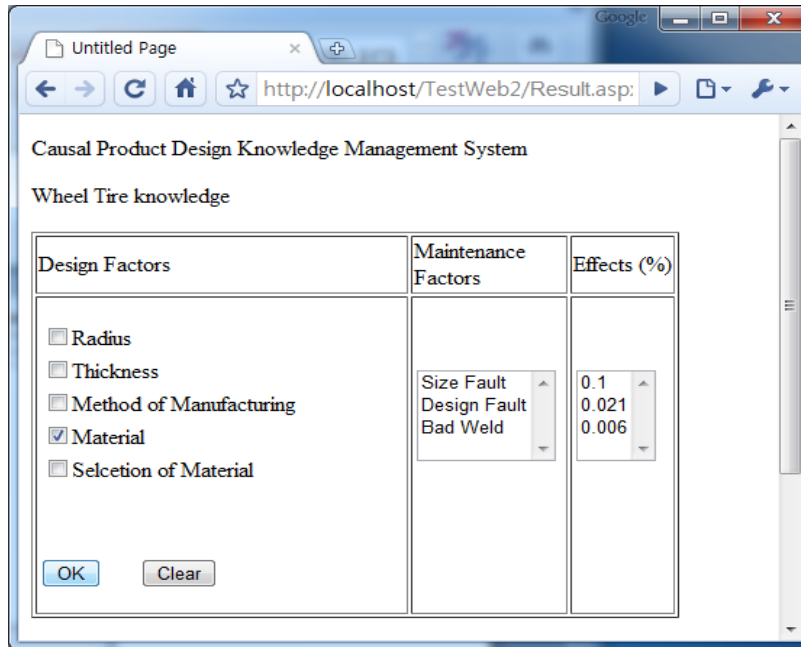
Figure 7-20 Effects of the design modification

Figure 7-21 illustrates the effects of design modification (*Selection of Material and Material*) in the integrated wheel knowledge. The integrated wheel knowledge includes five design factors and three maintenance factors. To compare between original knowledge and integrated knowledge, *Selection of Material* is selected for bad weld based wheel knowledge, *Material* is selected for tire knowledge, and *Selection of Material* or *Material* is selected for integrated knowledge. For the bad weld based wheel knowledge and integrated knowledge (Figure 7-20 (a) and 7-21 (a)), the effects of integrated knowledge provide more knowledge to make better decision; 1) *Size Fault* (0.348) has more effects than others (0.048, 0.017) in integrated knowledge, but *Bad Weld* (0.03) has more effects than *Design Fault*

(0.016) in bad weld based wheel knowledge, 2) even *Bad Weld* (0.03) has more effects than *Design Fault* (0.016) in bad weld based wheel knowledge, *Design Fault* (0.48) has more effects than *Bad Weld* (0.017) in integrated knowledge. For the tire knowledge and integrated knowledge (Figure 7-20 (b) and 7-21 (b)), the results of comparison is similar with the ones between bad weld based wheel knowledge and integrated knowledge.

Design Factors	Maintenance Factors	Effects (%)
<input type="checkbox"/> Radius <input type="checkbox"/> Thickness <input type="checkbox"/> Method of Manufacturing <input type="checkbox"/> Material <input checked="" type="checkbox"/> Selction of Material	Size Fault Design Fault Bad Weld	0.348 0.048 0.017

(a) For bad weld based wheel knowledge



(b) For tire knowledge

Figure 7-21 Effects of design modification in integrated wheel-tire knowledge

The integrated knowledge, which is within the same domain and in different domain, represents more knowledge than original knowledge. These results indicate that the integrated knowledge includes additional knowledge from tire knowledge and bad weld based wheel knowledge to provide better design decision for a new product design.

7.3 Conclusion

To use DCR evaluation method, one limitation of this method should be overcome. The limitation of DCR method is that it is strongly dependant with the number of vertices in causal knowledge network. This limitation restricts the comparison of multiple causal knowledge for selecting better design knowledge in product development. To overcome this limitation, new evaluation index, which is

called DCR index, is developed. Using DCR index, multiple causal knowledge with different number of nodes are compared. In this research, based on knowledge relationship, the new knowledge integration method is developed. The knowledge relationship classifies product development knowledge into three categories: inter-process, inter-actor, and inter-product knowledge in order to integrate heterogeneous existing product development knowledge. Based on these categories, the cases of the causal knowledge integration is developed. Finally, the innovative knowledge integration method is validated with wheel case.

CHAPTER 8

IMPLEMENTATION

8.1 Causal Design Knowledge Management System

In this research, web-based causal product design knowledge management system is developed with .net based developing environment (Figure 8-1). The detail developing environment is as follows: 1) the computer is PC based, 2) operating system is Microsoft Windows XP Professional SP2, 3) the database management system is MS SQL Express with ODBC for database connection, 4) basic programming language is C++, C#, and HTML/ASP for functions and web interface, 5) web server is Microsoft Internet Information Service, 6) and developing platform is Microsoft Visual Studio 2008. This developing environment is one of most popular configurations in web-based client and server system. The Visual Studio developing platform can easily connect C++ functions and C# web page development. In this platform, also, the Internet Information Service is ready to deploy the web service because all developing software is based on Microsoft applications.

Platform	Visual Studio
User Interface	HTML / ASP
Web Server	Internet Information Service
Language	C++
Meta-data	Ontology
DB connection	ODBC
DBMS	MS SQL
OS	Windows

Figure 8-1 .net based developing environment

Based on this developing environment, the web-based causal product design knowledge management system is implemented. The basic concept of this system is that a cad user wants to modify the existing design for a new one. The design support system can provide effects of the modification in real time as shown in Figure 8-2. The effects of the modification are based on causal knowledge inference using Bayesian belief network. Thus, the prediction of the modification includes the design and maintenance aspects of the product. This support system can be stand alone or API for CAD application.

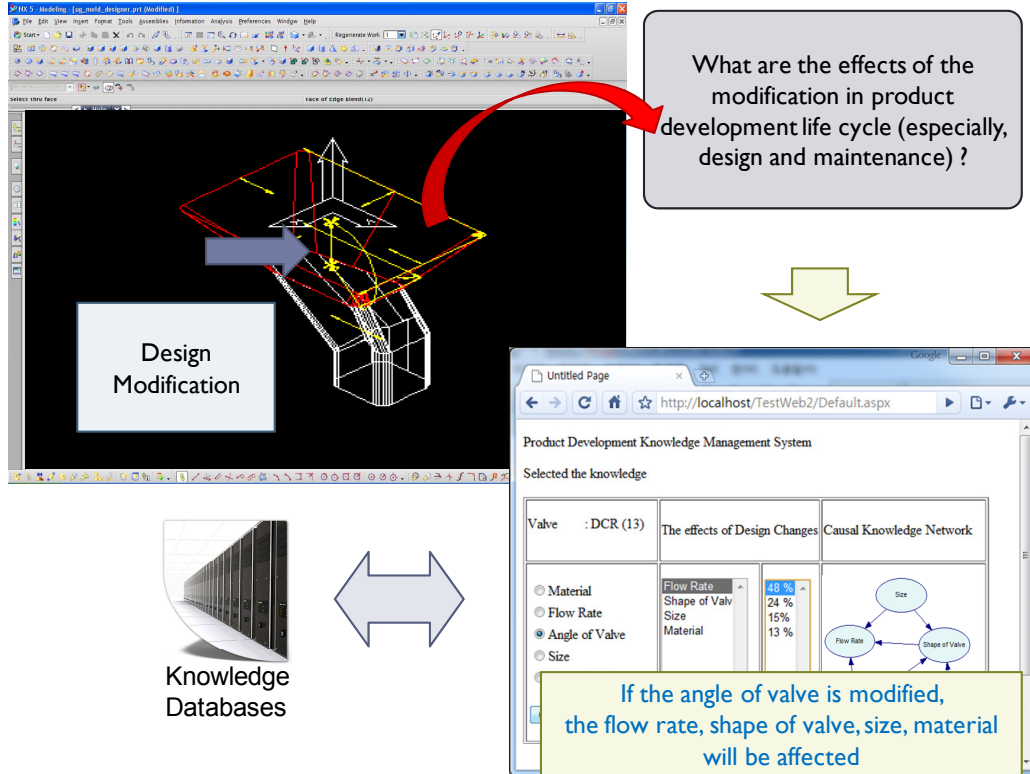


Figure 8-2 Basic concept: design support processes

In the Figure 8-3, the system architecture of the causal product design knowledge management is presented. This system is conducted with two main parts, design knowledge acquisition and reuse. For the design knowledge acquisition, the domain experts' knowledge is obtained through causal design management interface. The obtained knowledge is checked by causal design manage whether the knowledge is exist or not in the causal knowledge base. If there is the knowledge in the knowledge base, the new and existing knowledge are integrated by the knowledge integrator, the integrated knowledge is evaluated, and the knowledge and the evaluation results are stored to knowledge base. If there is not existing one in the knowledge base, the new knowledge is evaluated and is

stored with evaluation results. For the knowledge reuse, a designer requests a specific knowledge to causal design management interface, causal design manager searches for the requested knowledge in the knowledge base. If the requested knowledge is in the knowledge base, the knowledge and saved evaluation results are provided through report generator. If the requested knowledge is not in the knowledge base, causal design manager searches for similar alternatives and reports the alternatives to the designer with evaluation results. If the alternatives are needed to integrate, the alternatives are integrated by knowledge integrator and the generated knowledge is reported to designer with evaluation results.

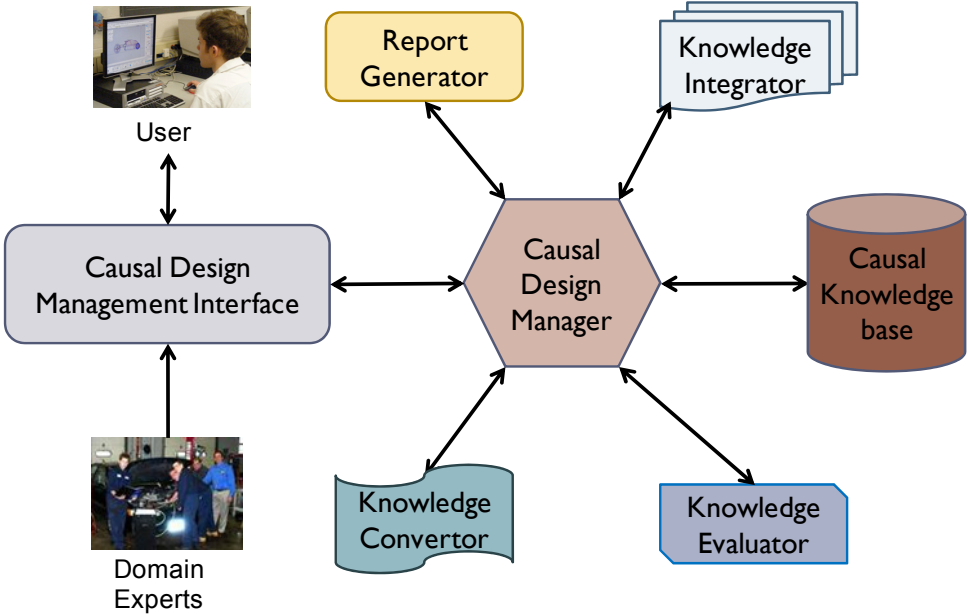
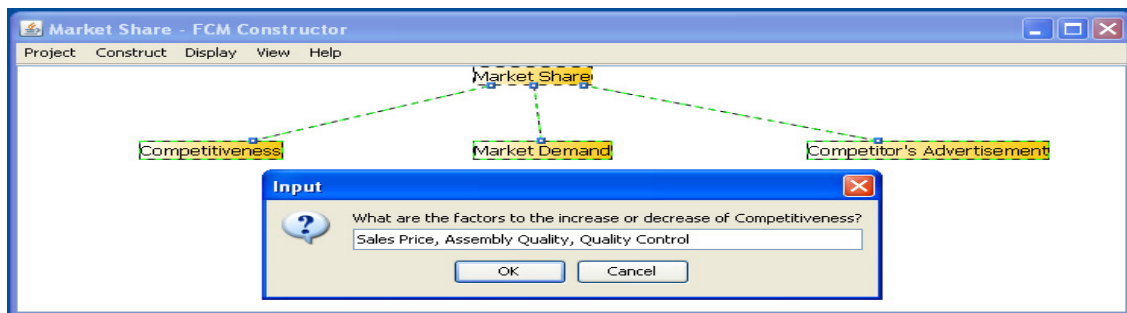


Figure 8-3 System architecture of the causal product design knowledge management system

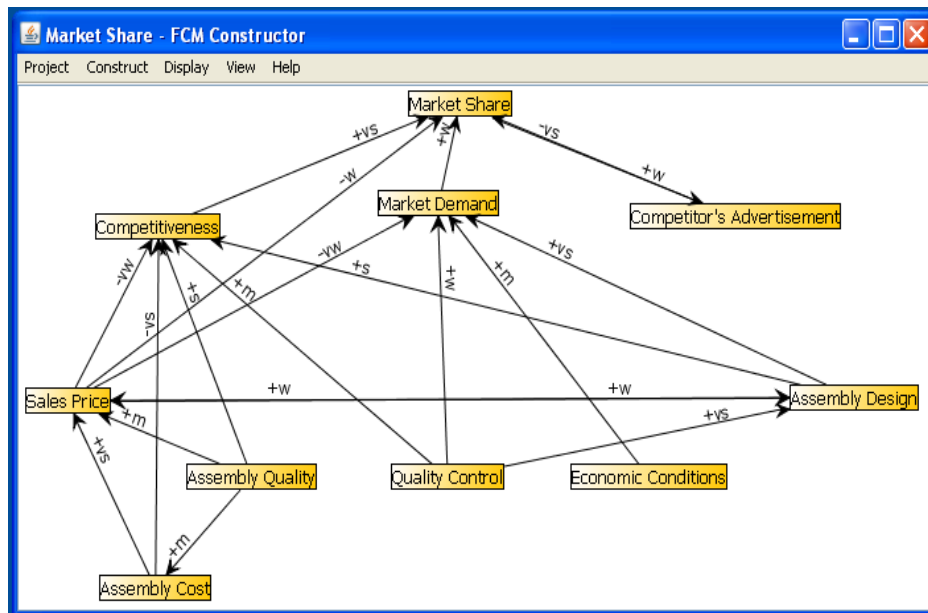
The systematic knowledge elicitation from domain experts, which is called FCM Constructor, is implemented. Figure 8-4 (a) shows elicitation of important variables in a specific knowledge to conduct knowledge network structure. After conducting knowledge structure, the relationship between variables, which can be positive or negative for the causal sign and very weak, weak, neutral, strong, very strong for casual strength, is obtained as shown in Figure 8-4 (b). When the causal relationship is completed, the fuzzy cognitive map is generated by FCM Constructor (Figure 8-4 (c)). Figure 8-4 (d) represents Bayesian belief network, which is converted from fuzzy cognitive map using FCM-BBN as mentioned in Chapter 4.



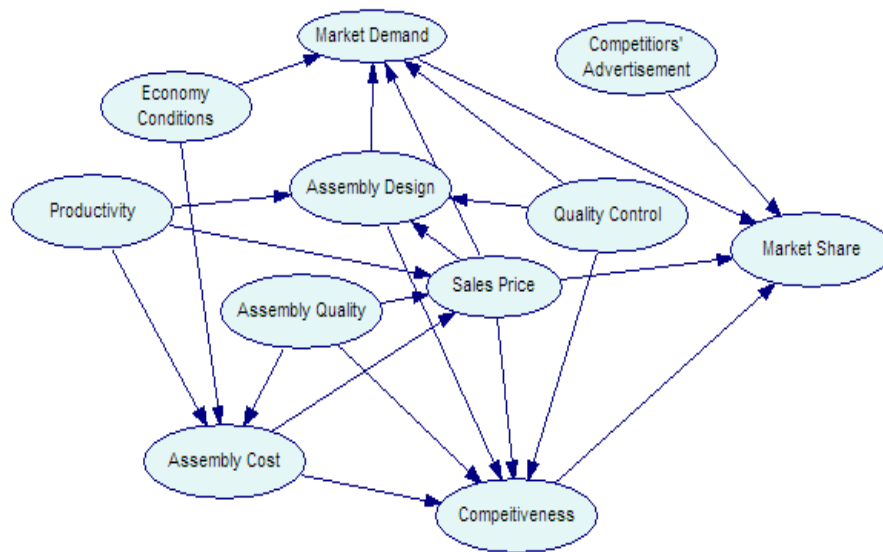
(a) Elicitation of important variables

Cause Variable	Effect Variable	Causal Sign	Causal Strength
Market Share	<input type="checkbox"/> Market Share	Positive	no causal effect
	<input type="checkbox"/> Competitiveness	Positive	no causal effect
	<input type="checkbox"/> Market Demand	Positive	no causal effect
	<input checked="" type="checkbox"/> Competitor's Advertisement	Positive	weak
	<input type="checkbox"/> Sales Price	Positive	no causal effect
	<input type="checkbox"/> Assembly Quality	Positive	no causal effect
	<input type="checkbox"/> Quality Control	Positive	no causal effect
	<input type="checkbox"/> Economic Conditions	Positive	no causal effect
	<input type="checkbox"/> Assembly Design	Positive	no causal effect
	<input type="checkbox"/> Assembly Cost	Positive	no causal effect

(b) Causal relationship between variables



(c) Systematical generated Fuzzy Cognitive Map

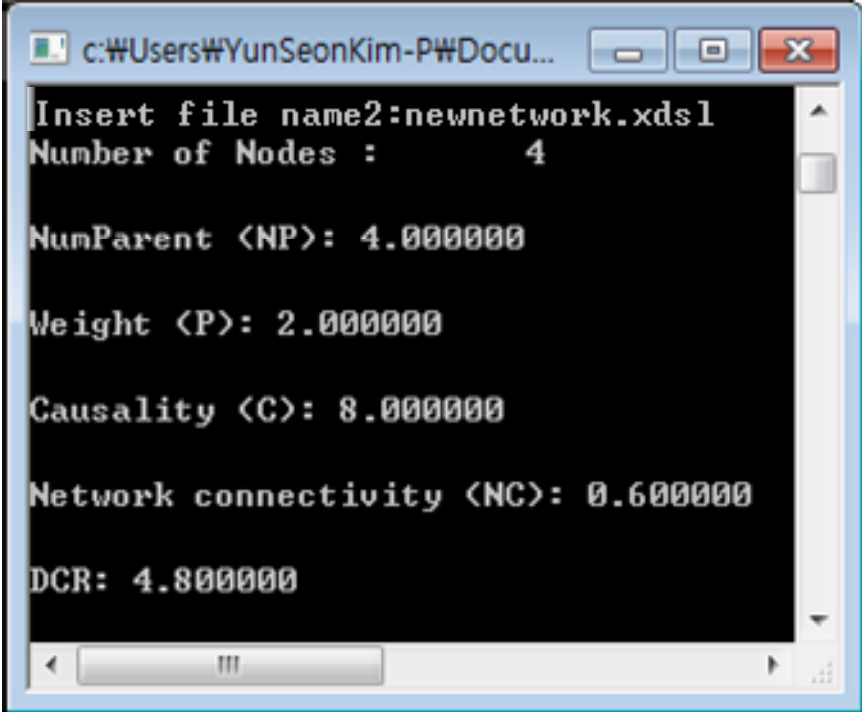


(d) Bayesian Belief Network from Fuzzy Cognitive Map

Figure 8-4 Systematic knowledge acquisition

The systematic generated causal knowledge is evaluated by DCR (Figure 8-5). This evaluation results include number of nodes, number of incoming arcs,

weight, causality, network connectivity, and DCR. The main function of DCR is implemented with C++ programming in Microsoft Visual Studio. The detail C++ codes is in the Appendix A. Also, the DCR index for multiple causal knowledge comparison is conducted as shown in Table 8-1. The DCR index includes minimum DCR (index is 0) and maximum DCR (index is 100) with different number of nodes (from 3 to 11). The detail evaluation results for index generation are attached in Appendix B.



```
c:\Users\YunSeonKim-P\Docu...  
Insert file name2:newnetwork.xds1  
Number of Nodes :      4  
  
NumParent <NP>: 4.000000  
  
Weight <P>: 2.000000  
  
Causality <C>: 8.000000  
  
Network connectivity <NC>: 0.600000  
  
DCR: 4.800000
```

Figure 8-5 Result of evaluation (DCR)

Table 8-1 Evaluation index (DCR index)

ID	# of Nodes	Connectivity	Probability	DCR	Index
1	3	Minimum	0.51	0.0036	0
	3	Maximum	0.99	34.5744	100
2	4	Minimum	0.51	0.0077	0
	4	Maximum	0.99	262.7654	100
3	5	Minimum	0.51	0.0154	0
	5	Maximum	0.99	1728.7200	100
4	6	Minimum	0.51	0.0282	0
	6	Maximum	0.99	12074.7334	100
5	7	Minimum	0.51	0.0473	0
	7	Maximum	0.99	98049.6898	100
6	8	Minimum	0.51	0.0738	0
	8	Maximum	0.99	981411.6411	100
7	9	Minimum	0.51	0.1088	0
	9	Maximum	0.99	10184059.6366	100
8	10	Minimum	0.51	0.1536	0
	10	Maximum	0.99	115732072.2365	100
9	11	Minimum	0.51	0.2093	0
	11	Maximum	0.99	1427091751.6989	100

The systematic generation of new causal design knowledge is implemented. Figure 8-6 shows causal knowledge integration with evaluation results for original knowledge. Also, the integration analysis results are provided at the bottom of Figure 8-6 (In this case, the knowledge networks' structures are identical). The integrated knowledge, which name is newNetwork.xdsl, is generated. The evaluation of the generated knowledge is analyzed as shown in Figure 8-8. The generated knowledge (4.8) has better DCR than two original knowledge (3.35, 2.69) in this integration case.

```
c:\Users\YunSeonKim-P\Documents...
Insert file name1:test1.xdsl
Insert file name2:test2.xdsl
Number of Nodes :      4

NumParent <NP>: 3.000000

Weight <P>: 1.861348

Causality <C>: 5.584044

Network connectivity <NC>: 0.600000

DCR: 3.350426
Number of Nodes :      4

NumParent <NP>: 3.000000

Weight <P>: 1.499800

Causality <C>: 4.499400

Network connectivity <NC>: 0.600000

DCR: 2.699640

First network -> number of nodes:4
Second network -> number of nodes:4
the number of matched node is 4
the networks' structure are identical
```

Figure 8-6 Causal knowledge integration example

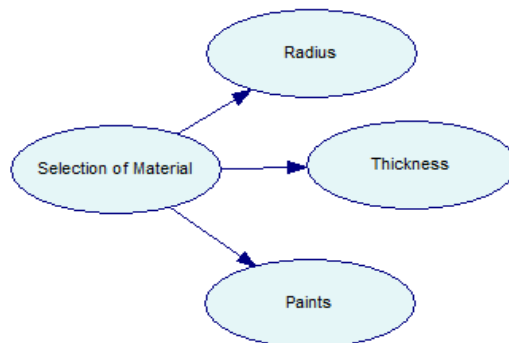
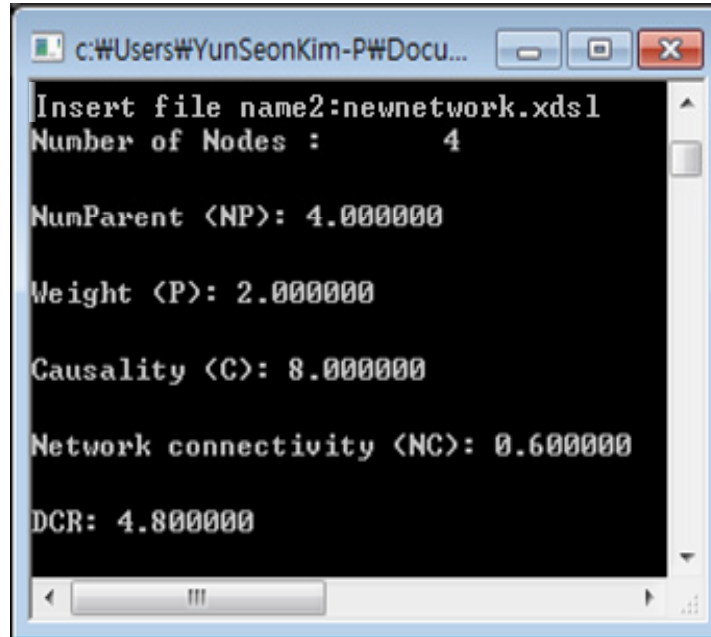


Figure 8-7 Integrated knowledge (newNetwork.xdsl)



```
c:\Users\YunSeonKim-PWDocu...
Insert file name2:newnetwork.xdsl
Number of Nodes : 4
NumParent <NP>: 4.000000
Weight <P>: 2.000000
Causality <C>: 8.000000
Network connectivity <NC>: 0.600000
DCR: 4.800000
```

Figure 8-8 integrated knowledge evaluation

The knowledge network interface engine is one of useful two outcomes from the implementation. This interface connects between knowledge inference engine and causal product design knowledge management system (Figure 8-9). The knowledge inference engine is a software, which is called GeNle and is developed by Decision Support Laboratory in University of Pittsburgh. This engine use .xdsl file format based on xml. Thus, the causal product design knowledge management system includes the interface engine for handling .xdsl file format. The GeNle's file is readable on the knowledge management system. Also, a generated file from the knowledge management system is readable on GeNle. The C++ programming codes of interface engine is attached in Appendix C.

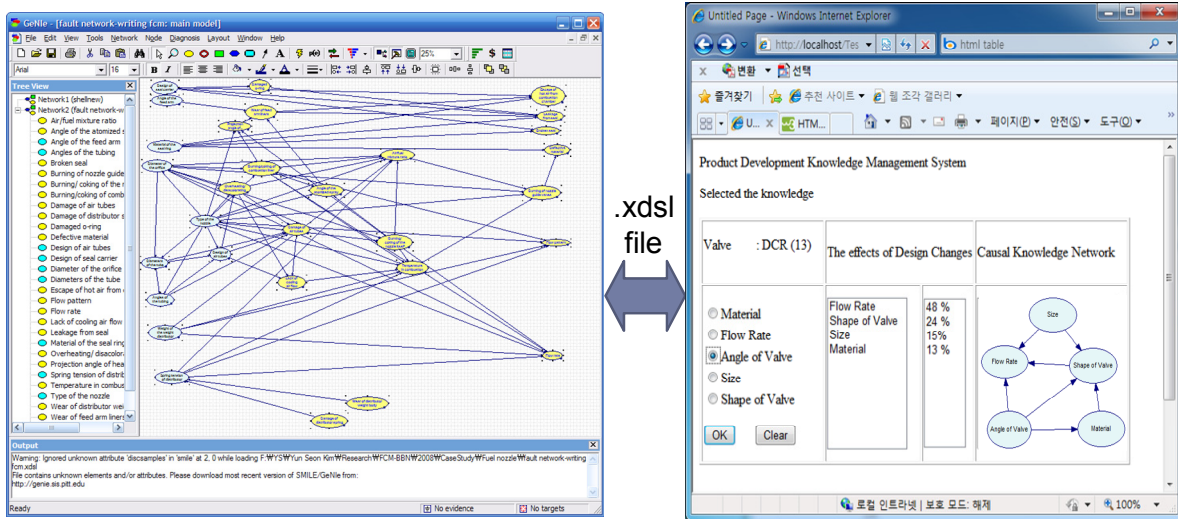


Figure 8-9 knowledge network interface engine between knowledge inference engine and causal product design knowledge management system

The causal design knowledge repository is the other of useful two outcomes. Using this management system, database is required. After using the system, the data are accumulated in the database and the accumulated repository includes product design factors, product design knowledge, and knowledge evaluation results. This causal design knowledge repository can be utilized for other knowledge systems.

Table 8-2 Causal design knowledge repository

Id	Kname	Kdomain	Kdate	DCR
1	Valve	automotive	03222010		16
2	Valve	Aero space	02132009		18
3	Wheel	Automotive	07252008		10
4	Fuel nozzle	Aero space	05262007		15
...					
n	Product A	Domain A	03262010		(20)

8.2 Case Study

This chapter presents and discusses three case studies—assembly design, wheel, and fuel nozzle—to evaluate knowledge representation, evaluation, and integration using the DCR-based method and causal knowledge integration method. All use a causal knowledge. The assembly design case represents knowledge, which is the relationship between assembly design and decision environment (Figure 8-10). The wheel and fuel nozzle cases illustrate the relationship of the design and maintenance aspects of each case (Figure 8-11 and 8-12). These three cases are conducted with three different causal knowledge: a Bayesian belief network (BBN), which is directly generated by the domain experts; a Bayesian belief network from fuzzy cognitive map (FCM-BBN), which is a knowledge network converted from FCM using the FCM-BBN method from our previous research as mentioned in Chapter 4; and a modified Bayesian belief network from fuzzy cognitive map (FCM-BBN-M), which is a FCM-BBN without direct edge if any indirect edge exists. Figure 8-11, 8-11, and 8-12 are one example of each case. The three causal knowledge models for each case are attached in Appendix D.

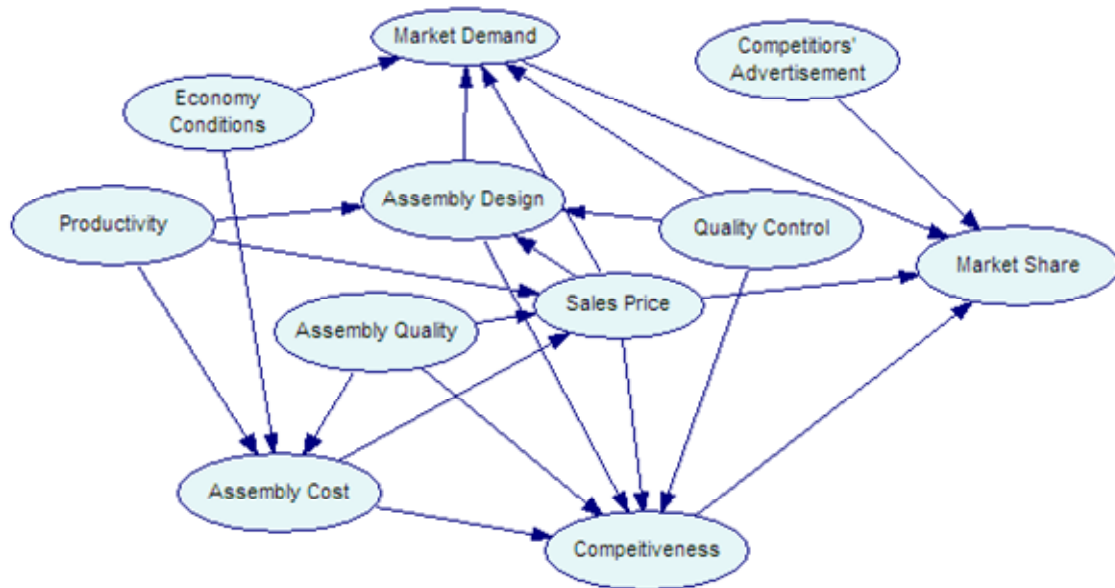


Figure 8-10 Example of assemble design

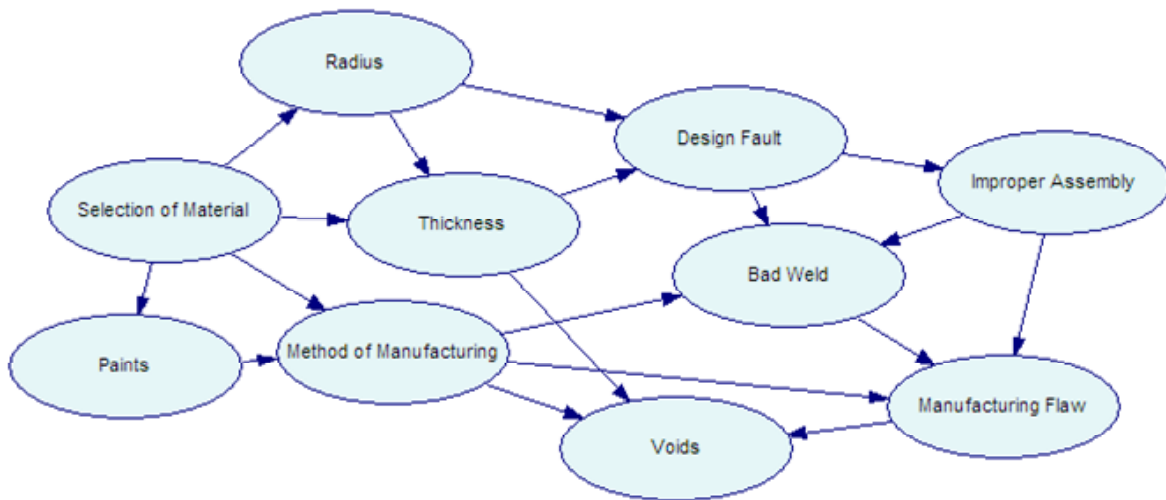


Figure 8-11 Example of wheel for automotive

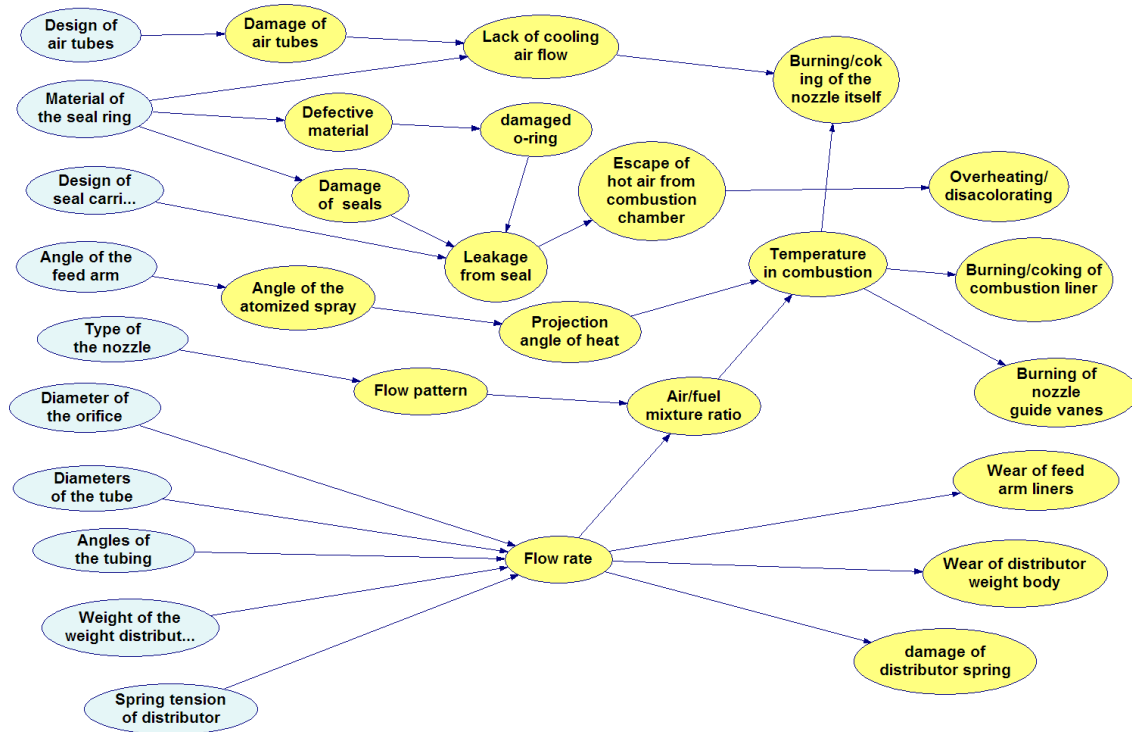


Figure 8-12 Example of fuel nozzle for aircraft engine

Table 8-3 presents the results of case study using DCR. For the assembly design case, BBN (109.672) has highest DCR; others are similar (83.414, 81.933). For the wheel case, FCM-BBN-M (58.841) has lowest DCR; others are similar (344.877, 360.621). For the fuel nozzle case, FCM-BBN (9702.75) highest DCR; FCM-BBN-M (2916.061) has better DCR than BBN (612.853). With each case, there are five sub criteria, number of vertices, number of incoming arcs, weight, causality, and network connectivity. For the assembly case, DCR for FCM-BBN (83.414) and FCM-BBN-M (81.933) are similar. However, if it is comparing the sub criteria, FCM-BBN (6.683) has more network connectivity than FCM-BBN-M (2.437). Oppositely, FCM-BBN (12.481) has less causality than FCM-BBN-M (33.614). This interpretation provides more useful information to analyze a DCR

evaluation result; even similar DCR knowledge can analyze with sub criteria for more details.

Table 8-3 Results of case study (DCR)

Case	Criterion	BBN	FCM-BBN	FCM-BBN-M
Assembly design	No. of Nodes	11	11	11
	No. of Parent	20	22	15
	Weight	0.76	0.567	2.241
	Causality	14.44	12.481	33.614
	Network Connectivity	7.595	6.683	2.437
	DCR	109.672	83.414	81.933
Wheel	No. of Nodes	10	10	10
	No. of Parent	16	18	11
	Weight	4.592	2.762	1.843
	Causality	73.479	49.716	20.28
	Network Connectivity	4.693	7.254	2.901
	DCR	344.877	360.621	58.841
Fuel nozzle	No. of Nodes	30	30	30
	No. of Parent	30	73	48
	Weight	6.545	10.646	10.121
	Causality	196.35	777.158	511.008
	Network Connectivity	3.121	12.485	5.706
	DCR	612.853	9702.75	2916.061

Based on DCR index (Table 8-1), currently, DCR index for fuel nozzle case is not covered because DCR index is conducted with number of vertices from three to eleven. Thus, assembly design and wheel cases are compared. For the assembly design case, DCR index are BBN (0.0009463), FCM-BBN (0.00007194), and FCM-BBN-M (0.00007066). For the wheel case, DCR index are BBN (0.00002415), FCM-BBN (0.00002525), and FCM-BBN-M (0.00000410). The assembly design knowledge is at least 2.7 times better than wheel knowledge with three knowledge models.

CHAPTER 9

CONCLUSTION

The US engineering industry base is facing a significant loss of knowledge due to large numbers of employees retiring in the next decade. Problems in various product developments including product design may arise when the expertise is no longer available or the knowledge is forgotten. Also, most of product design knowledge is not reusable, because product design knowledge in an organization remains un-codified. Generally, knowledge-based system can solve or infer these problems.

In this research, to solve knowledge retention and loss problems, a new web-based causal product design knowledge management system are developed. For this system, several new methodologies to capture, represent, store, and reuse experts' domain knowledge during the product development processes. To capture experts' domain knowledge, Chapter 4 addresses systematic knowledge acquisition and knowledge conversion from fuzzy cognitive maps to Bayesian belief network. For the knowledge representation formalism, Chapter 5 discusses the mathematical definitions of this research (procedural knowledge, causal knowledge, and knowledge transformation) and comparison and transformation between procedural knowledge and causal knowledge in order to use causal knowledge for knowledge representation formalism. After storing the causal knowledge, A system of causal design knowledge evaluation and support is presented and a new causal knowledge evaluation method is developed and

implemented in Chapter 6. This new causal knowledge evaluation method compares design knowledge using degree of causal representation (DCR). The results show that: 1) the more complex knowledge network model has higher DCR, 2) the knowledge network with higher weight has higher DCR, 3) the effect of weight increases with the more complex knowledge network. Chapter 7 discusses DCR index and knowledge integration. To use DCR evaluation method, one limitation of this method should be overcome. This limitation restricts the comparison of multiple causal knowledge for selecting better design knowledge in product development. To overcome this limitation, new evaluation index is developed. Using DCR index, multiple causal knowledge with different number of nodes are compared. Knowledge integration is an intelligent knowledge acquisition method from existing knowledge. Based on knowledge relationship, the new knowledge integration method is developed. The knowledge relationship classifies product development knowledge into three categories: inter-process, inter-actor, and inter-product knowledge in order to integrate heterogeneous existing product development knowledge. Based on this categories, the cases of the causal knowledge integration is developed. Chapter 8 implements web-based causal product design knowledge management system using Microsoft Visual Studio development environments.

With up to half a million engineers set to reach retirement age in the next decade, innovative and useful working environments, web-based collaborations, and underlying new technologies to support creative activities related to knowledge

retention and knowledge exchange are very important. Also, knowledge management is a very significant issue in product development. Knowledge management system is conducted with various methodologies and technologies, such as data mining, email, DBMS, and internet. Figure 9-1 illustrates a road-map of the knowledge management system requirements, which is adapted from the function's road-map of the knowledge management system addressed by Gardner Group.

This figure presents this research's contributions and is conducted with existing knowledge management system's requirement and outcomes of my research work. Vertical axis indicates the maturity of requirements and horizontal axis is separated by requirements in knowledge management system. The requirements include store&retrieve, send, structure&navigate, share, synthesize, and solve. Most of the dissertation's outcomes are located at low maturity, which means that this research is novel in these domains. To appropriately acquire domain experts' knowledge, FCM Constructor is developed and is a newer methodology in store & retrieve requirement. Causal knowledge representation formalism is a new area in the product development since there is no research to use causal knowledge representation. Also, knowledge transformation is a novel method in order to represent product design knowledge because most of product design knowledge is represented by procedural knowledge. Causal knowledge integration is an intelligent knowledge acquisition method from existing knowledge, which is not properly managed in current product development knowledge

framework. Causal reasoning is a proven technology for product development knowledge prediction and diagnosis in order to prevent potential failures in the product development lifecycle. DCR based causal knowledge evaluation method is a new evaluation method in causal knowledge. No research is conducted to evaluate causal knowledge. .xdsl file interface provides a communication environment between totally different applications. Web-based collaboration among stakeholder, who is currently working or retired, can provide true collaboration environment with discrete knowledge using real time based internet technology

This research extends design, technological and computational innovations in knowledge acquisition, knowledge representation, integration of knowledge, web-based knowledge management system to design problem solving processes. Results from this research are expected to advance our understanding of 1) capturing domain knowledge from experts, 2) systematic knowledge acquisition for current working engineering knowledge retention and for keeping retired professionals engaged in industry, 3) capturing and transforming existing procedural engineering knowledge to better knowledge representation formalism, 4) evaluating causal knowledge to make design decision, 5) comparing multiple design knowledge in heterogeneous product, 6) integrating existing design knowledge to generate refined knowledge, 7) and systematic knowledge management using information technologies and tools. Thus, this research leads to discovery and integration across these frontiers.

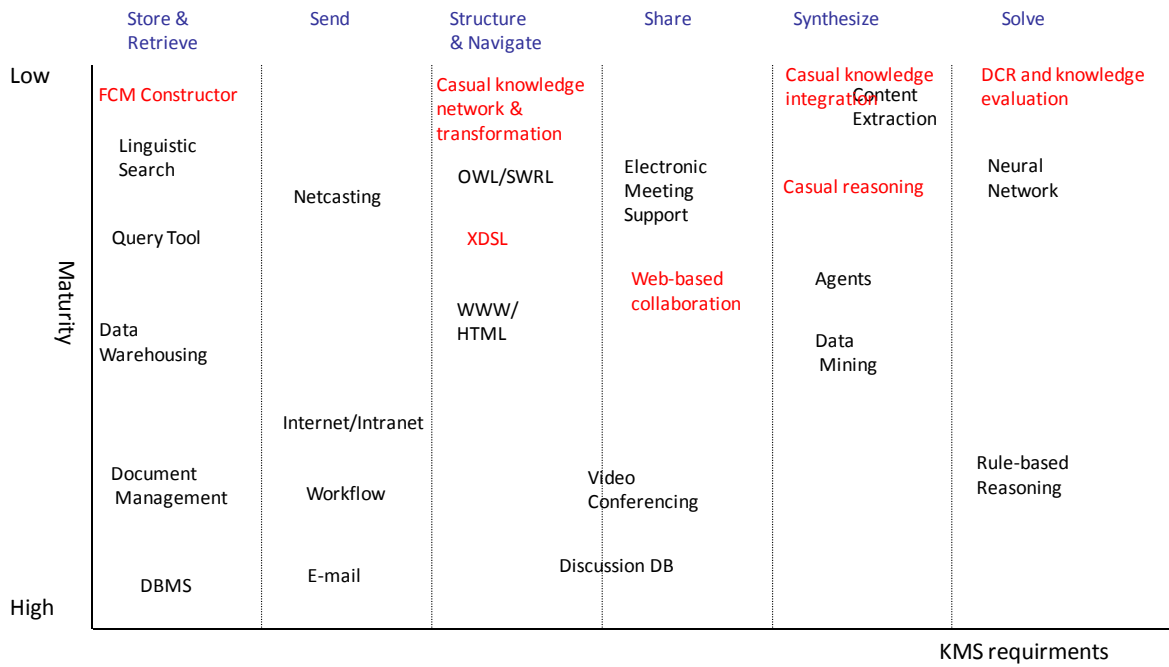


Figure 9-1 Contribution of the research in knowledge management system requirement

APPENDIX A

C++ programming code for DCR

```
#include "stdafx.h"
#include <stdio.h>
#include <windows.h>
#include <objbase.h>
#include <msxml2.h>
#include <string>
#include <vector>
#include "atlstr.h"
#include <iostream>
#include <math.h>

using namespace std;

// Macro that calls a COM method returning HRESULT value:
#define HRCALL(a, errmsg) \
do { \
    hr = (a); \
    if (FAILED(hr)) { \
        printf( "%s:%d HRCALL Failed: %s\n 0x%.8x = %s\n", \
            __FILE__, __LINE__, errmsg, hr, #a ); \
        goto clean; \
    } \
} while (0)

typedef std::vector<std::vector<std::vector<CString>>>> strVector3;
typedef std::vector<std::vector<CString>>> strVector2;
typedef std::vector<CString> strVector1;

typedef std::vector<std::vector<std::vector<double>>>> doubleVector3;
typedef std::vector<std::vector<double>>> doubleVector2;
typedef std::vector<double> doubleVector1;

typedef std::vector<std::vector<std::vector<int>>>> intVector3;
typedef std::vector<std::vector<int>>> intVector2;
typedef std::vector<int> intVector1;

// Helper function that put output in stdout and debug window
// in Visual Studio:
```

```

void dprintf( char * format, ...)
{
    static char buf[1024000000];
    va_list args;
    va_start( args, format );
    vsprintf_s( buf, format, args );
    va_end( args);
    OutputDebugStringA( buf);
    printf("%s", buf);
}

// Helper function to create a DOM instance:
IXMLDOMDocument * DomFromCOM()
{
    HRESULT hr;
    IXMLDOMDocument *pxmlDoc = NULL;

    HRESULT( CoCreateInstance(__uuidof(DOMDocument30),
        NULL,
        CLSCTX_INPROC_SERVER,
        __uuidof(IXMLDOMDocument),
        (void**)&pxmlDoc),
        "Create a new DOMDocument");

    HRESULT( pxmlDoc->put_async(VARIANT_FALSE),
        "should never fail");
    HRESULT( pxmlDoc->put_validateOnParse(VARIANT_FALSE),
        "should never fail");
    HRESULT( pxmlDoc->put_resolveExternals(VARIANT_FALSE),
        "should never fail");

    return pxmlDoc;
clean:
    if (pxmlDoc)
    {
        pxmlDoc->Release();
    }
    return NULL;
}

VARIANT VariantString(BSTR str)
{
    VARIANT var;

```

```

VariantInit(&var);
V_BSTR(&var) = SysAllocString(str);
V_VT(&var) = VT_BSTR;
return var;
}

void ReportParseError(IXMLDOMDocument *pDom, char *desc) {
    IXMLDOMParseError *pXMLErr=NULL;
    BSTR bstrReason = NULL;
    HRESULT hr;
    HRCALL(pDom->get_parseError(&pXMLErr),
           "dom->get_parseError: ");
    HRCALL(pXMLErr->get_reason(&bstrReason),
           "parseError->get_reason: ");

    printf("%s %S\n",desc, bstrReason);
clean:
    if (pXMLErr) pXMLErr->Release();
    if (bstrReason) SysFreeString(bstrReason);
}

CString stringfromBstr (BSTR bstr){

    TCHAR szFinal[255000];

    // direct conversion from BSTR to LPCTSTR only works in Unicode
    _stprintf(szFinal, _T("%s"), (LPCTSTR)bstr);

    // _bstr_t bstrIntermediate(bstr); // convert to _bstr_t

    CString strFinal;

    // you have to go through _bstr_t to have it work in ANSI and Unicode
    _stprintf(szFinal, _T("%s"), (LPCTSTR)bstr);

    // Or using MFC
    strFinal.Format(_T("%s"), (LPCTSTR)bstr);

    return strFinal;
}

```



```

int findIndirect(doubleVector2 adjcentRelationData, intVector1 childNameNum,int memory, int count, int self,
intVector1 preNode, int pre)
{
    intVector1 childNameNum1;

    for (int x=0; x<childNameNum.size();x++)
    {
        int prenodecount=0;
        count=0;
        for (int y=0;y<adjcentRelationData[childNameNum[x]].size();y++)
        {
            if(adjcentRelationData[childNameNum[x]][y])
            {
                for (int z=0;z<preNode.size();z++)
                {
                    if(preNode[z])
                    {
                        if(preNode[z]==childNameNum[x])
                        {
                            pre=1;
                            //printf("\tDD");
                        }
                    }
                }
            }
            if (childNameNum[x]==memory)
            {}
            else if (self==y | childNameNum[x]==y || pre)
            {
                pre=0;
            }
            else if (memory == y )
            {
                count++;
                preNode.push_back(childNameNum[x]);
                //printf("\t%d",count);printf("BB");
            }
            else{
                preNode.push_back(childNameNum[x]);
                childNameNum1.push_back(y);
                count+=findIndirect(adjcentRelationData, childNameNum1,memory, count, self, preNode, pre);
                childNameNum1.clear();
            }
        }
    }
}

```

```

    }

    }
    preNode.clear();
    }
return count;
}

```

```

doubleVector2 getindirectrelation(doubleVector2 relationDataTemp, strVector1 matchNode, intVector1
matchNodeNum, strVector3 nodeData,doubleVector2 adjacentRelationData)

```

```

{
doubleVector2 indirect=relationDataTemp;

strVector1 childName;
intVector1 childNameNum;
intVector1 preNode;
intVector1 childNameNum1;
CString temp123, temp1234;
double temp12345;

int memory=0;
int self=0;
int pre=0;

for (int i=0; i<indirect.size();i++)
{
self=i;
int count1=0;

for (int j=0;j<indirect[i].size();j++)
{

if (i==j)
{}
else
{
memory=j;
for (int x=0; x<adjacentRelationData.size();x++)
{
for (int y=0;y<adjacentRelationData[x].size();y++)
{
if (x == self && adjacentRelationData[x][y] )
{

```

```

        childNameNum.push_back(y);
        //printf("\t%S",adjcentRelationData[x][y]);
        //printf("\t%d",count);
    }
}
}
childName.clear();

int count=0;
for (int x=0; x<childNameNum.size();x++)
{
    int prenodecount=0;

    for (int y=0;y<adjcentRelationData[childNameNum[x]].size();y++)
    {
        if(adjcentRelationData[childNameNum[x]][y])
        {
            for (int z=0;z<preNode.size();z++)
            {
            }
            for (int z=0;z<preNode.size();z++)
            {
                if(preNode[z])
                {
                    if(preNode[z]==childNameNum[x])
                    {
                        pre=1;
                        //printf("\tDD");
                    }
                }
            }
            if (childNameNum[x]==memory)
            {}
            else if (self==y || childNameNum[x]==y || pre)
            {
                pre=0;
            }
            else if (memory == y )
            {
                count++;
                preNode.push_back(childNameNum[x]);
            }
            else{

```

```

        preNode.push_back(childNameNum[x]);
        childNameNum1.push_back(y);
        count+=findIndirect(adjcentRelationData, childNameNum1,memory, count, self, preNode, pre);
        childNameNum1.clear();
    }
}
preNode.clear();
}
count1=count;
}
}
childNameNum.clear();
indirect[i][j]=(double)count1;
}
}
return indirect;
}

```

//get the relation of nodes (direct + indirect)

```

doubleVector2 getRelation (strVector3 nodeData)
{
    doubleVector1 relationRowData;
    doubleVector2 relationDataTemp;
    doubleVector2 adjcentRelationData;
    doubleVector1 indirectrelationRowData;
    doubleVector2 indirectrelationDataTemp;
    doubleVector1 directrelationRowData;
    doubleVector2 directrelationDataTemp;
    doubleVector2 doubletemp8;

    double temp10;
    CString temp11,temp12;

    strVector1 matchNode;

    intVector1 matchNodeNum;

    // match matrix
    for (int x=0; x<nodeData.size();x++)
    {

```

```

matchNode.push_back(nodeData[x][0][0]);
matchNodeNum.push_back(x);

}

//adjcent matrix
for (int x=0;x<nodeData.size();x++)
{
    for(int y=0;y<nodeData.size();y++)
    {
        relationRowData.push_back(0);
    }
    relationDataTemp.push_back(relationRowData);
    relationRowData.clear();
}

//find direct relation
adjcentRelationData=relationDataTemp;
for (int x=0;x<nodeData.size();x++)
{
    for(int y=0;y<nodeData[x].size();y++)
    {
        for(int z=0;z<nodeData[x][y].size();z++)
        {
            if(y==2)
            {
                temp11=nodeData[x][y][z];
                for (int s=0;s<matchNode.size();s++)
                {
                    temp12=matchNode[s];
                    if (temp11==temp12)
                    {
                        temp10=matchNodeNum[s];
                        adjcentRelationData[temp10][x]=1;
                    }
                }
            }
        }
    }
}
directrelationDataTemp=adjcentRelationData;

```

```

//find indirect relation
indirectrelationDataTemp=getindirectrelation(relationDataTemp, matchNode, matchNodeNum, nodeData,
adjcentRelationData);

for (int x=0;x<indirectrelationDataTemp.size();x++)
{
for(int y=0;y<indirectrelationDataTemp[x].size();y++)
{
if (x==y)
{
indirectrelationDataTemp[x][y]=0;
}
}
}
for (int x=0;x<relationDataTemp.size();x++)
{
for(int y=0;y<relationDataTemp[x].size();y++)
{
relationDataTemp[x][y]=directrelationDataTemp[x][y]+indirectrelationDataTemp[x][y];
}
}
return relationDataTemp;
}

double getPower( double num, int p)
{
double c=num;
num=1;
for (int i=0;i<p;i++)
{
num*=c;
}
return num;
}

int _tmain(int argc, _TCHAR* argv[])
{
IXMLDOMDocument *pXMLDom=NULL;
IXMLDOMNodeList *pNodes=NULL;
IXMLDOMNodeList *pChildLists=NULL;
IXMLDOMNode *pNode=NULL;
IXMLDOMNamedNodeMap *pNodeMap=NULL;
IXMLDOMNamedNodeMap *pChildMap=NULL;

```

```

IXMLDOMNode *pNodeList=NULL;
IXMLDOMNode *pChildList=NULL;
IXMLDOMNode *pChildAttributeList=NULL;
    BSTR bstr = NULL;
BSTR bstr1 = NULL;
BSTR bstr3 = NULL;
BSTR temp = NULL;
    VARIANT_BOOL status;
    VARIANT var,var1;
    HRESULT hr;
    long length, length1, length2, length3;

FILE *ifs;
FILE *ofs;

strVector1 rowData;
strVector1 rowData1;
strVector1 rowData2;
strVector2 singleData;
strVector3 nodeData;
strVector3 nodeData1;
strVector3 nodeData2;

CString strTemp, strTemp1, strTemp2, strTemp3, strTemp4;

char address1[100];
cout << "Insert file name1:";
cin>>address1;
CString add11=(CString)address1;
BSTR add1 = add11.AllocSysString();

char address2[100];
cout << "Insert file name2:";
cin>>address2;
CString add22=(CString)address2;
BSTR add2 = add22.AllocSysString();

if (add1)
{
    Colnitialize(NULL);

    pXMLDom = DomFromCOM();
    if (!pXMLDom) goto clean;

```

```

VariantInit(&var);
var = VariantString(add1);
HRCALL(pXMLDom->load(var, &status), "dom->load(): ");

if (status!=VARIANT_TRUE) {
    ReportParseError(pXMLDom,
        "Failed to load DOM from stocks.xml");
    goto clean;
}

// Query a node-set.
ifs=fopen("nodeInofr.txt","w");

if (bstr) SysFreeString(bstr);
bstr = SysAllocString(L"//nodes[0]/*");
HRCALL(pXMLDom->selectNodes(bstr, &pNodes), "selectNodes ");
if (!pNodes) {
    ReportParseError(pXMLDom, "Error while calling selectNodes ");
}
else
{
    //printf("Results from selectNodes:\n");
    HRCALL(pNodes->get_length(&length), "get_length: ");

int countlength=(int)length;

    for (long i=0; i<length; i++) {
        if (pNode) pNode->Release();
        HRCALL(pNodes->get_item(i, &pNode), "get_item: ");
        if (bstr) SysFreeString(bstr);
        HRCALL(pNode->get_nodeName(&bstr), "get_nodeName: ");
        //printf("Node (%d), <%S>\n",i, bstr);
        if (bstr) SysFreeString(bstr);
        //HRCALL(pNode->get_xml(&bstr), "get_xml: ");

// For getting attributes in a node
HRCALL(pNode->get_attributes(&pNodeMap), "get_attributes: ");

//HRCALL(pNodeMap->get_length(&length1), "get_length: ");

```



```

for (long j=0;j<1;j++){

    HRCALL(pNodeMap->get_item(j,&pNodeList), "get_item: ");

    HRCALL(pNodeList->get_baseName(&bstr), "get_baseName: ");

    HRCALL(pNodeList->get_text(&bstr), "get_text: ");

    strTemp=stringfromBstr(bstr);
    rowData.push_back(strTemp);

    singleData.push_back(rowData);
    rowData.clear();

}

// For getting childnodes in a node
HRCALL(pNode->get_childNodes(&pChildLists), "get_childNodes: ");
HRCALL(pChildLists->get_length(&length2), "get_length: ");

for (long k=0;k<length2;k++){
    HRCALL(pChildLists->get_item(k,&pChildList), "get_item: ");

    HRCALL(pChildList->get_nodeName(&bstr1), "get_nodeName: ");

    if(k==0)
    {
        strTemp1=stringfromBstr(bstr1);
    }
    strTemp2=stringfromBstr(bstr1);
    HRCALL(pChildList->get_attributes(&pChildMap), "get_attributes: ");

    HRCALL(pChildMap->get_length(&length3), "get_length: ");

    if (length3 == 0){
    }
    else
    {length3 =1;}

    // For getting attributes in a childnode
    for (long l=0;l<length3;l++){

        HRCALL(pChildMap->get_item(l,&pChildAttributeList), "get_item: ");

```

```

HRCALL(pChildAttributeList->get_baseName(&bstr), "get_baseName: ");

HRCALL(pChildAttributeList->get_text(&bstr), "get_text: ");
strTemp=stringfromBstr(bstr);
  rowData.push_back(strTemp);
}

if ((strTemp1!=strTemp2)&&(length3!='0')){

HRCALL(pChildList->get_text(&bstr3), "get_text: ");
if (strTemp2=="probabilities"){
  strTemp2=stringfromBstr(bstr3);
  char *a=(char *)strTemp2.GetString();
  char aa[100000];

  for (int i=0;i<strTemp2.GetLength()*2+1;i++)
  {
    if(a[i] != NULL && a[i] != ' '){
      aa[i]=a[i];
      strTemp4+=aa[i];
    }
    else if (a[i] != NULL && a[i] == ' ' )
    {
      rowData2.push_back(strTemp4);
      strTemp4.Empty();
    }
    else if (i == strTemp2.GetLength()*2)
    {
      rowData2.push_back(strTemp4);
      strTemp4.Empty();
    }
  }
}
else
{
  strTemp=stringfromBstr(bstr3);

  char *a=(char *)strTemp.GetString();
  char aa[100000];
  for (int i=0;i<strTemp.GetLength()*2+1;i++)
  {
    if(a[i] != NULL && a[i] != ' '){

```

```

aa[i]=a[i];
strTemp3+=aa[i];
}
else if (a[i] != NULL && a[i] == ' ')
{
rowData1.push_back(strTemp3);
strTemp3.Empty();
}
else if (i == strTemp.GetLength()*2)
{
rowData1.push_back(strTemp3);
strTemp3.Empty();
}
}
}
}
}
}

```

```

singleData.push_back(rowData);
rowData.clear();
singleData.push_back(rowData1);
rowData1.clear();
singleData.push_back(rowData2);
rowData2.clear();

```

```

nodeData.push_back(singleData);
singleData.clear();
}

```

```

nodeData1=nodeData;
nodeData.clear();

```

```

// optimality analysis
intVector1 parentData;
doubleVector1 probData;
doubleVector1 causalData;
doubleVector1 nodeProbRowData;

```

```

doubleVector2 nodeProbData;
doubleVector2 relationData;
doubleVector2 ratioData;

```

```
CString doubleTemp, doubleTemp1, doubleTemp2, doubleTemp3, doubleTemp4;
```

```
int count100=0;
```

```
double total=0;
```

```
//Get the number of parents
```

```
for (int x=0;x<nodeData1.size();x++)  
{  
    for(int y=0;y<nodeData1[x].size();y++)  
    {  
        for(int z=0;z<nodeData1[x][y].size();z++)  
        {  
            if (y==2)  
            {  
                count100= count100+1;  
                //printf("\t%d",count100);  
            }  
        }  
        if(y==2)  
        {  
            parentData.push_back(count100);  
            count100=0;  
        }  
    }  
}
```

```
// Get the probabilities of each node
```

```
for (int x=0;x<nodeData1.size();x++)  
{  
    for(int y=0;y<nodeData1[x].size();y++)  
    {  
        for(int z=0;z<2;z++)  
        {  
            if (y==3)  
            {  
                doubleTemp=nodeData1[x][y][z];  
                nodeProbRowData.push_back(_wtof(doubleTemp));  
            }  
            if(y==3)  
            {  
                nodeProbData.push_back(nodeProbRowData);  
                nodeProbRowData.clear();  
            }  
        }  
    }  
}
```

```

    }
    }
    }
}
relationData=getRelation (nodeData1);

int maxNode= nodeData1.size();
printf("Number of Nodes : \t%d\n",maxNode);

for (int z=1;z<maxNode-1;z++)
{
    double num=1;
    double denum=1;

    for (int x=1;x<maxNode-1;x++)
    {
        num=num*x;
    }

    for (int y=1;y<maxNode-1-z;y++)
    {
        denum=denum*y;
    }
    total+=(num/denum);
}

double temp321=1;
for (int x=1;x<maxNode;x++)
{
    temp321*=x;
}

total=((maxNode-1)+total*(maxNode-1)/temp321);

doubleVector1 numParent;
doubleVector1 prob;
doubleVector2 ratioRelation=relationData;

for (int i=0;i<ratioRelation.size();i++)
{
    for (int j=0; j<ratioRelation[i].size();j++)
    {
        ratioRelation[i][j]=relationData[i][j]/total;
    }
}

```

```

    }
}

for (int i=0;i<nodeData1.size();i++)
{
    numParent.push_back(nodeData1[i][2].size());
}

for (int i=0;i<nodeData1.size();i++)
{
    double temp11=0;
    double temp12=0;
    double temp13=0;
    int numState=2;
    for (int j=0;j<numState;j++)
    {
        temp13=_wtof(nodeData1[i][3][j]);
        temp13-=0.5;
        temp12=getPower(temp13,2);
        temp11+=temp12;
    }
    temp11=temp11/(numState-1);
    prob.push_back(temp11);
}

double sumNumParent=0;
double sumProb=0;
double sumRatioRelation=0;

for (int i=0;i<nodeData1.size();i++)
{
    sumNumParent+=numParent[i];
    sumProb+=prob[i];

    for (int j=0; j<ratioRelation[i].size();j++)
    {
        sumRatioRelation+=ratioRelation[i][j];
    }
}

double DCR=0;
DCR=sumNumParent*sumProb*sumRatioRelation;
printf("\nNumParent (NP): %lf\n",sumNumParent);
printf("\nWeight (P): %lf\n",sumProb);

```

```

printf("\nCausality (C): %f\n",sumProb*sumNumParent);
printf("\nNetwork connectivity (NC): %f\n",sumRatioRelation);
printf("\nDCR: %f\n",DCR);
}
}

if (add2)
{
    ColInitialize(NULL);

    pXMLDom = DomFromCOM();
    if (!pXMLDom) goto clean;

    VariantInit(&var);
    var = VariantString(add2);
    HRESULT(pXMLDom->load(var, &status), "dom->load(): ");

    if (status!=VARIANT_TRUE) {
        ReportParseError(pXMLDom,
            "Failed to load DOM from stocks.xml");
        goto clean;
    }

    // Query a node-set.
    ifs=fopen("nodeInofr.txt","w");

    if (bstr) SysFreeString(bstr);
    bstr = SysAllocString(L"//nodes[0]/*");
    HRESULT(pXMLDom->selectNodes(bstr, &pNodes), "selectNodes ");
    if (!pNodes) {
        ReportParseError(pXMLDom, "Error while calling selectNodes ");
    }
    else
    {
        HRESULT(pNodes->get_length(&length), "get_length: ");
        int countlength=(int)length;

        for (long i=0; i<length; i++) {
            if (pNode) pNode->Release();
            HRESULT(pNodes->get_item(i, &pNode), "get_item: ");
            if (bstr) SysFreeString(bstr);
            HRESULT(pNode->get_nodeName(&bstr), "get_nodeName: ");

```

```

    if (bstr) SysFreeString(bstr);

// For getting attributes in a node
HRCALL(pNode->get_attributes(&pNodeMap), "get_attributes: ");
for (long j=0;j<1;j++){
    HRCALL(pNodeMap->get_item(j,&pNodeList), "get_item: ");
    HRCALL(pNodeList->get_baseName(&bstr), "get_baseName: ");
    HRCALL(pNodeList->get_text(&bstr), "get_text: ");

    strTemp=stringfromBstr(bstr);
    rowData.push_back(strTemp);

    singleData.push_back(rowData);
    rowData.clear();
}

// For getting childnodes in a node
HRCALL(pNode->get_childNodes(&pChildLists), "get_childNodes: ");
HRCALL(pChildLists->get_length(&length2), "get_length: ");

for (long k=0;k<length2;k++){
    HRCALL(pChildLists->get_item(k,&pChildList), "get_item: ");
    HRCALL(pChildList->get_nodeName(&bstr1), "get_nodeName: ");
    if(k==0)
    {
        strTemp1=stringfromBstr(bstr1);
    }
    strTemp2=stringfromBstr(bstr1);
    HRCALL(pChildList->get_attributes(&pChildMap), "get_attributes: ");

    HRCALL(pChildMap->get_length(&length3), "get_length: ");

    if (length3 == 0){
    }
    else
    {length3 =1;}

// For getting attributes in a childnode
for (long l=0;l<length3;l++){
    HRCALL(pChildMap->get_item(l,&pChildAttributeList), "get_item: ");
    HRCALL(pChildAttributeList->get_baseName(&bstr), "get_baseName: ");
    HRCALL(pChildAttributeList->get_text(&bstr), "get_text: ");
    strTemp=stringfromBstr(bstr);

```



```

    rowData.push_back(strTemp);
}

if ((strTemp1!=strTemp2)&&(length3!='0')){

HRCALL(pChildList->get_text(&bstr3), "get_text: ");
if (strTemp2=="probabilities"){
    strTemp2=stringfromBstr(bstr3);
    char *a=(char *)strTemp2.GetString();
    char aa[100000];

    for (int i=0;i<strTemp2.GetLength()*2+1;i++)
    {
        if(a[i] != NULL && a[i] != ' '){
            aa[i]=a[i];
            strTemp4+=aa[i];
        }
        else if (a[i] != NULL && a[i] == ' ' )
        {
            rowData2.push_back(strTemp4);
            strTemp4.Empty();
        }
        else if (i == strTemp2.GetLength()*2)
        {
            rowData2.push_back(strTemp4);
            strTemp4.Empty();
        }
    }
}
else
{
    strTemp=stringfromBstr(bstr3);
    char *a=(char *)strTemp.GetString();
    char aa[100000];

    for (int i=0;i<strTemp.GetLength()*2+1;i++)
    {
        if(a[i] != NULL && a[i] != ' '){
            aa[i]=a[i];

            strTemp3+=aa[i];
        }
        else if (a[i] != NULL && a[i] == ' ' )

```

```

    {
        rowData1.push_back(strTemp3);
        strTemp3.Empty();
    }
    else if (i == strTemp.GetLength()*2)
    {
        rowData1.push_back(strTemp3);
        strTemp3.Empty();
    }
}
}
}
}
singleData.push_back(rowData);
rowData.clear();
singleData.push_back(rowData1);
rowData1.clear();
singleData.push_back(rowData2);
rowData2.clear();
nodeData.push_back(singleData);
singleData.clear();
}

nodeData2=nodeData;
nodeData.clear();

// optimality analysis
intVector1 parentData;
doubleVector1 probData;
doubleVector1 causalData;
doubleVector1 nodeProbRowData;

doubleVector2 nodeProbData;
doubleVector2 relationData;
doubleVector2 ratioData;

CString doubleTemp, doubleTemp1, doubleTemp2, doubleTemp3, doubleTemp4;

int count100=0;
double total=0;

//Get the number of parents
for (int x=0;x<nodeData2.size();x++)

```

```

{
for(int y=0;y<nodeData2[x].size();y++)
{
for(int z=0;z<nodeData2[x][y].size();z++)
{
if (y==2)
{
count100= count100+1;
}
}
if(y==2)
{
parentData.push_back(count100);
count100=0;
}
}
}

// Get the probabilities of each node
for (int x=0;x<nodeData2.size();x++)
{
for(int y=0;y<nodeData2[x].size();y++)
{
for(int z=0;z<2;z++)
{
if (y==3)
{
doubleTemp=nodeData2[x][y][z];
nodeProbRowData.push_back(_wtof(doubleTemp));
}
if(y==3)
{
nodeProbData.push_back(nodeProbRowData);
nodeProbRowData.clear();
}
}
}
}

relationData=getRelation (nodeData2);
int maxNode= nodeData2.size();
printf("Number of Nodes : \t%d\n",maxNode);

```

```

for (int z=1;z<maxNode-1;z++)
{
    double num=1;
    double denum=1;

    for (int x=1;x<maxNode-1;x++)
    {
        num=num*x;
    }

    for (int y=1;y<maxNode-1-z;y++)
    {
        denum=denum*y;
    }
    total+=(num/denum);
}

double temp321=1;
for (int x=1;x<maxNode;x++)
{
    temp321*=x;
}
total=((maxNode-1)+total*(maxNode-1)/temp321);

doubleVector1 numParent;
doubleVector1 prob;
doubleVector2 ratioRelation=relationData;

for (int i=0;i<ratioRelation.size();i++)
{
    for (int j=0; j<ratioRelation[i].size();j++)
    {
        ratioRelation[i][j]=relationData[i][j]/total;
    }
}

for (int i=0;i<nodeData2.size();i++)
{
    numParent.push_back(nodeData2[i][2].size());
}

for (int i=0;i<nodeData2.size();i++)

```

```

{
double temp11=0;
double temp12=0;
double temp13=0;
int numState=2;
for (int j=0;j<numState;j++)
{
temp13=_wtof(nodeData2[i][3][j]);

temp13-=0.5;
//printf("\n%f",temp13);
temp12=getPower(temp13,2);
//printf("\n%f",temp12);
temp11+=temp12;

}
temp11=temp11/(numState-1);

prob.push_back(temp11);
}

double sumNumParent=0;
double sumProb=0;
double sumRatioRelation=0;

for (int i=0;i<nodeData2.size();i++)
{
sumNumParent+=numParent[i];
sumProb+=prob[i];

for (int j=0; j<ratioRelation[i].size();j++)
{
sumRatioRelation+=ratioRelation[i][j];
}
}
double DCR=0;
DCR=sumNumParent*sumProb*sumRatioRelation;
printf("\nNumParent (NP): %f\n",sumNumParent);
printf("\nWeight (P): %f\n",sumProb);
printf("\nCausality (C): %f\n",sumProb*sumNumParent);
printf("\nNetwork connectivity (NC): %f\n",sumRatioRelation);
printf("\nDCR: %f\n",DCR);

```

```
}  
}
```

```
int nodeData1cnt=1;
```

```
int nodeData2cnt=1;
```

```
for (int x=1;x<nodeData1.size();x++)
```

```
{
```

```
    nodeData1cnt++;
```

```
}
```

```
for (int x=1;x<nodeData2.size();x++)
```

```
{
```

```
    nodeData2cnt++;
```

```
}
```

```
printf("\n\nFirst network -> number of nodes:%d \n", nodeData1cnt);
```

```
printf("Second network -> number of nodes:%d \n", nodeData2cnt);
```

```
int matchNodecnt=0;
```

```
int matchParentcnt=0;
```

```
bool matchParent=1;
```

```
int match1 [100][2] ={0} ;
```

```
int match2 [100][2] ={0} ;
```

```
for (int x=0;x<nodeData1.size();x++)
```

```
{
```

```
    for (int y=0; y<nodeData2.size();y++)
```

```
    {
```

```
        if (nodeData1[x][0][0] == nodeData2[y][0][0])
```

```
        {
```

```
            ++matchNodecnt;
```

```
            match1[x][0]=1;
```

```
            match2[y][0]=1;
```

```
            if (nodeData1[x][2].size() == nodeData2[y][2].size())
```

```
            {
```

```
                for (int f=0; f<nodeData1[x][2].size();f++)
```

```
                {
```

```
                    for (int g=0; g<nodeData2[y][2].size(); g++)
```

```
                    {
```

```

        if (nodeData1[x][2][f] == nodeData2[y][2][g])
        {
            matchParentcnt++;
        }
        else
        {
            matchParent = 0;
        }
    }
}
if (nodeData1[x][2].size()==matchParentcnt)
{
    match1[x][1]=1;
    match2[y][1]=1;
}
}
}
else
{
    matchParent = 0;
}
}
}
if (matchParent)
{
    printf("the number of matched node is %d \n ", matchNodecnt);

    printf("the networks' structure are identical \n");

    strVector3 nodeData3 = nodeData1;
    for (int i=0; i<nodeData3.size();i++)
    {
        for (int j=0; j<nodeData3[i][3].size();j++)
        {
            double a =wcstod(nodeData1[i][3][j],NULL);
            double c =wcstod(nodeData2[i][3][j],NULL);
            double b = (a+c)/2;

            nodeData3[i][3][j].Format(_T("%lf"),b);
        }
    }
}

```



```

int ad=257;
for (int x=0;x<nodeData3.size();x++)
{
    fprintf(ofs, "\t\t\t<node id=\"%s\", (LPCTSTR)nodeData3[x][0][0]);
    fprintf(ofs, ">\n");
    fprintf(ofs, "\t\t\t<name>%s ", (LPCTSTR)nodeData3[x][0][0]);
    fprintf(ofs, "</name>\n");
    fprintf(ofs, "\t\t\t<interior color=\"e5f6f7\" />\n");
    fprintf(ofs, "\t\t\t<outline color=\"000080\" />\n");
    fprintf(ofs, "\t\t\t<font color=\"000000\" name=\"Arial\" size=\"8\" />\n");
    fprintf(ofs, "\t\t\t<position>%d %d %d %d</position>\n", aa, ab, ac, ad);
    fprintf(ofs, "\t\t\t</node>\n");
    aa=aa+25;
    ab=ab+25;
    ac=ac+25;
    ad=ad+25;

}
fprintf(ofs, "\t</genie>\n");
fprintf(ofs, "\t</extensions>\n");
fprintf(ofs, "</smile>\n");

}
else
{
    if (matchNodecnt == nodeData1.size() && matchNodecnt == nodeData2.size())
    {
        printf("the number of matched node is %d \n ", matchNodecnt);
        printf("the networks have the same number of matched node and differnet structures \n");

        strVector3 nodeData3 = nodeData1;
        int probcnt[100]={0};

        for (int x=0;x<nodeData3.size();x++)
        {
            for (int h=0; h<nodeData2.size();h++)
            {
                if (nodeData1[x][0][0]==nodeData2[h][0][0])
                {
                    if (match1[x][1]==1 && match2[h][1]==1)
                    {
                        {
                        }
                    }
                    else if (match1[x][1]==0 && match2[h][1]==0)

```

```

{
for (int y=0;y<nodeData3[x][2].size();y++)
{
probcnt[x]++;
}

for (int j=0;j<nodeData3[x][2].size();j++)
{
for (int k=0;k<nodeData2[h][2].size();k++)
{
if (nodeData3[x][2][j]==nodeData2[h][2][k])
{
}
else
{
probcnt[x]++;
}
}
}
}
}
}
}
}
double a = 0.0;
double c = 0.0;
double b = 0.0;

for (int i=0; i<nodeData3.size();i++)
{
for (int y=0; y<nodeData2.size(); y++)
{
if (nodeData1[i][0][0]==nodeData2[y][0][0])
{
if (match1[i][1]==1 && match2[y][1]==1)
{
for (int j=0; j<nodeData3[i][3].size();j++)
{
a =(double)wcstod(nodeData1[i][3][j],NULL);
c =(double)wcstod(nodeData2[y][3][j],NULL);
b = (double)(a+c)/2;

nodeData3[i][3][j].Format(_T("%lf"),b);
}
}
}
}
}
}
}

```



```

fprintf(ofs, "\\t\\t<genie version=\\\"1.0\\\" app=\\\"GeNle 2.0.3092.0\\\" name=\\\"newNetwork\\\"
faultnameformat=\\\"nodestate\\\">\\n");

```

```

int aa=48;
int ab=87;
int ac=147;
int ad=257;
for (int x=0;x<nodeData3.size();x++)
{
    fprintf(ofs, "\\t\\t\\t<node id=\\\"%s\\\", (LPCTSTR)nodeData3[x][0][0]);
    fprintf(ofs, "\\>\\n");
    fprintf(ofs, "\\t\\t\\t<name>%s ", (LPCTSTR)nodeData3[x][0][0]);
    fprintf(ofs, "</name>\\n");
    fprintf(ofs, "\\t\\t\\t<interior color=\\\"e5f6f7\\\" />\\n");
    fprintf(ofs, "\\t\\t\\t<outline color=\\\"000080\\\" />\\n");
    fprintf(ofs, "\\t\\t\\t<font color=\\\"000000\\\" name=\\\"Arial\\\" size=\\\"8\\\" />\\n");
    fprintf(ofs, "\\t\\t\\t<position>%d %d %d %d</position>\\n", aa, ab, ac, ad);
    fprintf(ofs, "\\t\\t\\t</node>\\n");
    aa=aa+25;
    ab=ab+25;
    ac=ac+25;
    ad=ad+25;
}
fprintf(ofs, "\\t\\t</genie>\\n");
fprintf(ofs, "\\t</extensions>\\n");
fprintf(ofs, "</smile>\\n");
}
else
{
    printf("the number of matched node is %d \\n ", matchNodecnt);
    printf("the networks have differnet number of matched node and differnet structures \\n");
    strVector3 nodeData3 = nodeData1;

    double a = 0.0;
    double c = 0.0;
    double b = 0.0;

    for (int i=0; i<nodeData3.size();i++)
    {

        for (int y=0; y<nodeData2.size(); y++)
        {

```



```

for (int y=0;y<nodeData3[x][2].size();y++)
{
    fprintf(ofs, "%s", (LPCTSTR)nodeData3[x][2][y]);
}
fprintf(ofs, " </parents>\n");
fprintf(ofs, "\t\t\t<probabilities>"/*, (LPCTSTR)nodeData3[x][3][0]*/);
for (int z=0;z<nodeData3[x][3].size();z++)
{
    fprintf(ofs, "%s", (LPCTSTR)nodeData3[x][3][z]);
}
fprintf(ofs, " </probabilities>\n");
fprintf(ofs, "\t\t</cpt>\n");
}
for (int x=0; x<nodeData3.size(); x++)
{
    for (int y=0; y<nodeData2.size(); y++)
    {
        if (match1[x][0]==0 && match2[y][0]==0)
        {
            if (match1[x][1]==0 && match2[y][1]==0)
            {
                if (nodeData1[x][0][0]==nodeData2[y][0][0])
                {}
                else
                {
                    fprintf(ofs, "\t\t<cpt id=\"%s", (LPCTSTR)nodeData2[y][0][0]);
                    fprintf(ofs, "\" diagtype=\"target\" ranked=\"true\">\n");
                    fprintf(ofs, "\t\t\t<state id=\"%s", (LPCTSTR)nodeData2[y][1][0]);
                    fprintf(ofs, "\" fault=\"true\" />\n");
                    for (int k=1;k<nodeData2[y][1].size();k++)
                    {
                        fprintf(ofs, "\t\t\t<state id=\"%s", (LPCTSTR)nodeData2[y][1][k]);
                        fprintf(ofs, "\" />\n");
                    }
                    //if (nodeData3[x][2][0])
                    fprintf(ofs, "\t\t\t<parents>"/*, (LPCTSTR)nodeData3[x][2][0]*/);

                    for (int k=0;k<nodeData2[y][2].size();k++)
                    {
                        fprintf(ofs, "%s", (LPCTSTR)nodeData2[y][2][k]);
                    }
                    fprintf(ofs, " </parents>\n");
                    fprintf(ofs, "\t\t\t<probabilities>"/*, (LPCTSTR)nodeData3[x][3][0]*/);
                }
            }
        }
    }
}

```

```

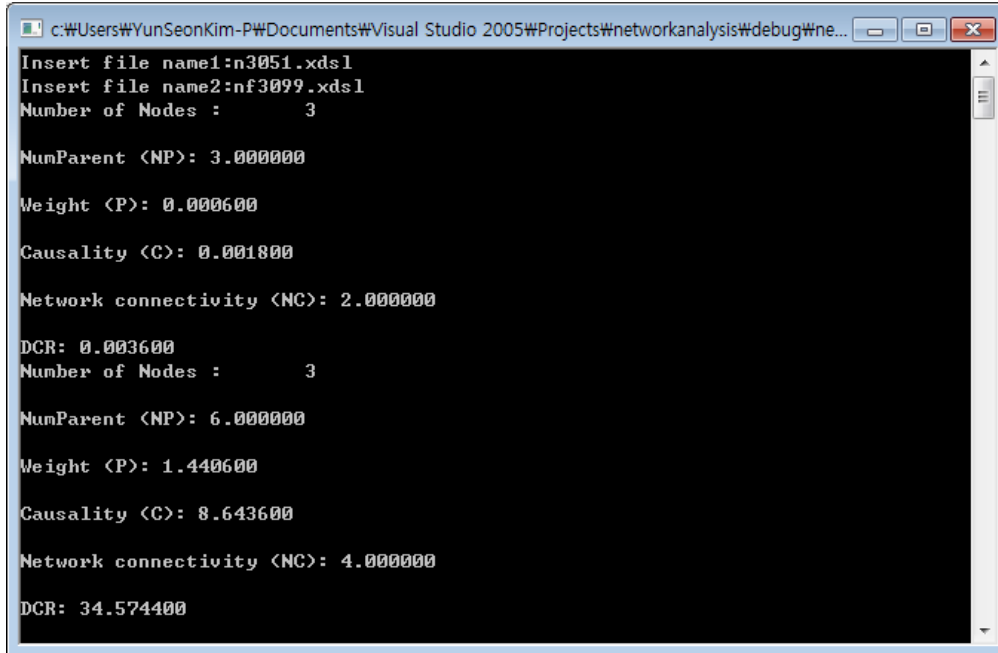
    for (int z=0;z<nodeData2[y][3].size();z++)
    {
        double a = (double)wcstod(nodeData2[y][3][z],NULL);
        a=a/2;
        fprintf(ofs, " %f",a);
    }
    fprintf(ofs, " </probabilities>\n");
    fprintf(ofs, "\t\t</cpt>\n");
}
}
}
}

fprintf(ofs, "\t</nodes>\n");
fprintf(ofs, "\t<extensions>\n");
fprintf(ofs, "\t\t<genie version=\"1.0\" app=\"GeNIe 2.0.3092.0\" name=\"newNetwork\"
faultnameformat=\"nodestate\">\n");
int aa=48;
int ab=87;
int ac=147;
int ad=257;
for (int x=0;x<nodeData3.size();x++)
{
    fprintf(ofs, "\t\t\t<node id=\"%s\", (LPCTSTR)nodeData3[x][0][0]);
    fprintf(ofs, ">\n");
    fprintf(ofs, "\t\t\t\t<name>%s ", (LPCTSTR)nodeData3[x][0][0]);
    fprintf(ofs, "</name>\n");
    fprintf(ofs, "\t\t\t\t<interior color=\"e5f6f7\" />\n");
    fprintf(ofs, "\t\t\t\t<outline color=\"000080\" />\n");
    fprintf(ofs, "\t\t\t\t<font color=\"000000\" name=\"Arial\" size=\"8\" />\n");
    fprintf(ofs, "\t\t\t\t<position>%d %d %d %d</position>\n", aa, ab, ac, ad);
    fprintf(ofs, "\t\t\t</node>\n");
    aa=aa+25;
    ab=ab+25;
    ac=ac+25;
    ad=ad+25;
}
for (int x=0; x<nodeData3.size(); x++)
{
    for (int y=0; y<nodeData2.size(); y++)
    {
        if (match1[x][0]==0 && match2[y][0]==0)

```


APPENDIX B

Evaluation results for DCR index generation



```
c:\Users#YunSeonKim-P#Documents#Visual Studio 2005#Projects#networkanalysis#debug#ne...
Insert file name1:n3051.xds1
Insert file name2:nf3099.xds1
Number of Nodes :      3

NumParent <NP>: 3.000000

Weight <P>: 0.000600

Causality <C>: 0.001800

Network connectivity <NC>: 2.000000

DCR: 0.003600
Number of Nodes :      3

NumParent <NP>: 6.000000

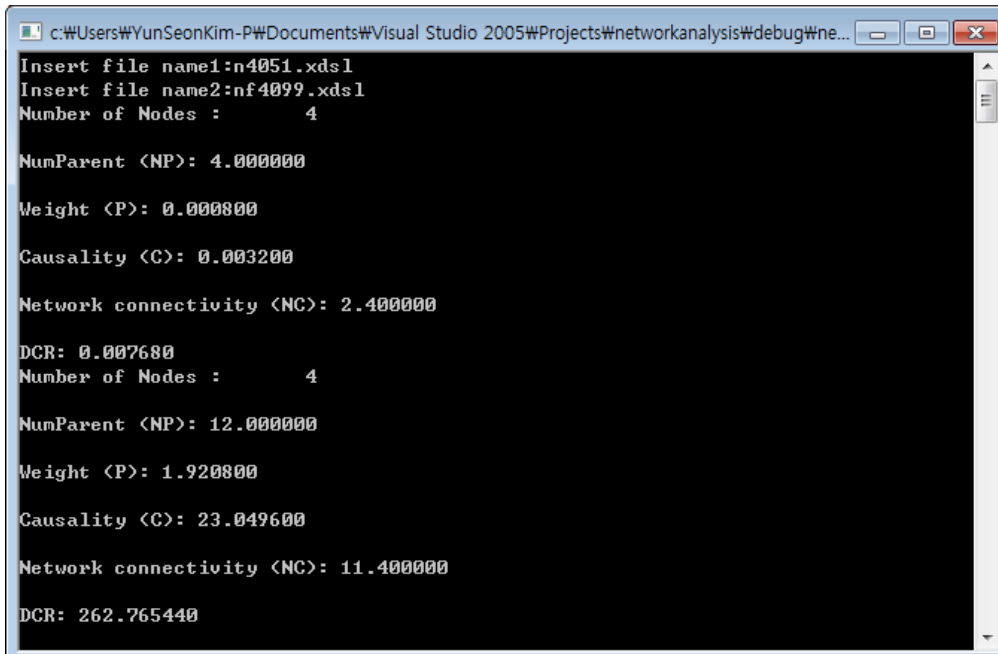
Weight <P>: 1.440600

Causality <C>: 8.643600

Network connectivity <NC>: 4.000000

DCR: 34.574400
```

Figure B-1 Evaluation results with three vertices



```
c:\Users#YunSeonKim-P#Documents#Visual Studio 2005#Projects#networkanalysis#debug#ne...
Insert file name1:n4051.xds1
Insert file name2:nf4099.xds1
Number of Nodes :      4

NumParent <NP>: 4.000000

Weight <P>: 0.000800

Causality <C>: 0.003200

Network connectivity <NC>: 2.400000

DCR: 0.007600
Number of Nodes :      4

NumParent <NP>: 12.000000

Weight <P>: 1.920800

Causality <C>: 23.049600

Network connectivity <NC>: 11.400000

DCR: 262.765440
```

Figure B-2 Evaluation results with four vertices

```
c:\Users\YunSeonKim-P\Documents\Visual Studio 2005\Projects\networkanalysis\debug#ne...
Insert file name1:n5051.xdsl
Insert file name2:nf5099.xdsl
Number of Nodes :      5

NumParent <NP>: 5.000000

Weight <P>: 0.001000

Causality <C>: 0.005000

Network connectivity <NC>: 3.076923

DCR: 0.015385
Number of Nodes :      5

NumParent <NP>: 20.000000

Weight <P>: 2.401000

Causality <C>: 48.020000

Network connectivity <NC>: 36.000000

DCR: 1728.720000
```

Figure B-3 Evaluation results with five vertices

```
c:\Users\YunSeonKim-P\Documents\Visual Studio 2005\Projects\networkanalysis\debug#ne...
Insert file name1:n6051.xdsl
Insert file name2:nf6099.xdsl
Number of Nodes :      6

NumParent <NP>: 6.000000

Weight <P>: 0.001200

Causality <C>: 0.007200

Network connectivity <NC>: 3.913043

DCR: 0.028174
Number of Nodes :      6

NumParent <NP>: 30.000000

Weight <P>: 2.881200

Causality <C>: 86.436000

Network connectivity <NC>: 139.695652

DCR: 12074.733391
```

Figure B-4 Evaluation results with six vertices

```
c:\Users\YunSeonKim-P\Documents\Visual Studio 2005\Projects\networkanalysis\debug#ne...
Insert file name1:n7051.xdsl
Insert file name2:nf7099.xdsl
Number of Nodes :      7

NumParent <NP>: 7.000000

Weight <P>: 0.001400

Causality <C>: 0.009800

Network connectivity <NC>: 4.822967

DCR: 0.047265
Number of Nodes :      7

NumParent <NP>: 42.000000

Weight <P>: 3.361400

Causality <C>: 141.178800

Network connectivity <NC>: 694.507177

DCR: 98049.689845
```

Figure B-5 Evaluation results with seven vertices

```
c:\Users\YunSeonKim-P\Documents\Visual Studio 2005\Projects\networkanalysis\debug#ne...
Insert file name1:n8051.xdsl
Insert file name2:nf8099.xdsl
Number of Nodes :      8

NumParent <NP>: 8.000000

Weight <P>: 0.001600

Causality <C>: 0.012800

Network connectivity <NC>: 5.763293

DCR: 0.073770
Number of Nodes :      8

NumParent <NP>: 56.000000

Weight <P>: 3.841600

Causality <C>: 215.129600

Network connectivity <NC>: 4561.955403

DCR: 981411.641084
```

Figure B-6 Evaluation results with eight vertices

```
c:\Users\YunSeonKim-P\Documents\Visual Studio 2005\Projects\networkanalysis\debug#ne...
Insert file name1:n9051.xds1
Insert file name2:nf9099.xds1
Number of Nodes :      9

NumParent <NP>: 9.000000

Weight <P>: 0.001800

Causality <C>: 0.016200

Network connectivity <NC>: 6.717636

DCR: 0.108826
Number of Nodes :      9

NumParent <NP>: 72.000000

Weight <P>: 4.321800

Causality <C>: 311.169600

Network connectivity <NC>: 32728.324478

DCR: 10184059.636621
```

Figure B-7 Evaluation results with nine vertices

```
c:\Users\YunSeonKim-P\Documents\Visual Studio 2005\Projects\networkanalysis\debug#ne...
Insert file name1:n10051.xds1
Insert file name2:nf10099.xds1
Number of Nodes :     10

NumParent <NP>: 10.000000

Weight <P>: 0.002000

Causality <C>: 0.020000

Network connectivity <NC>: 7.680325

DCR: 0.153607
Number of Nodes :     10

NumParent <NP>: 90.000000

Weight <P>: 4.802000

Causality <C>: 432.180000

Network connectivity <NC>: 267786.737555

DCR: 115732072.236532
```

Figure B-8 Evaluation results with ten vertices

APPENDIX C

C++ programming code for knowledge interface engine

```
// Helper function to create a DOM instance:
IXMLDOMDocument * DomFromCOM()
{
    HRESULT hr;
    IXMLDOMDocument *pxmlDoc = NULL;

    HRESULT( CoCreateInstance(__uuidof(DOMDocument30),
        NULL,
        CLSCTX_INPROC_SERVER,
        __uuidof(IXMLDOMDocument),
        (void*)&pxmlDoc,
        "Create a new DOMDocument"));

    HRESULT( pxmDoc->put_async(VARIANT_FALSE),
        "should never fail");
    HRESULT( pxmDoc->put_validateOnParse(VARIANT_FALSE),
        "should never fail");
    HRESULT( pxmDoc->put_resolveExternals(VARIANT_FALSE),
        "should never fail");

    return pxmDoc;
clean:
    if (pxmDoc)
    {
        pxmDoc->Release();
    }
    return NULL;
}

VARIANT VariantString(BSTR str)
{
    VARIANT var;
    VariantInit(&var);
    V_BSTR(&var) = SysAllocString(str);
    V_VT(&var) = VT_BSTR;
    return var;
}
```

```

void ReportParseError(IXMLDOMDocument *pDom, char *desc) {
    IXMLDOMParseError *pXMLErr=NULL;
    BSTR bstrReason = NULL;
    HRESULT hr;
    HRESULT hrCall1 = HRCALL(pDom->get_parseError(&pXMLErr),
        "dom->get_parseError: ");
    HRESULT hrCall2 = HRCALL(pXMLErr->get_reason(&bstrReason),
        "parseError->get_reason: ");

    printf("%s %S\n",desc, bstrReason);
clean:
    if (pXMLErr) pXMLErr->Release();
    if (bstrReason) SysFreeString(bstrReason);
}

CString stringfromBstr (BSTR bstr){

    TCHAR szFinal[255000];

    // direct conversion from BSTR to LPCTSTR only works in Unicode
    _stprintf(szFinal, _T("%s"), (LPCTSTR)bstr);

    // _bstr_t bstrIntermediate(bstr); // convert to _bstr_t

    CString strFinal;

    // you have to go through _bstr_t to have it work in ANSI and Unicode
    _stprintf(szFinal, _T("%s"), (LPCTSTR)bstr);

    // Or using MFC

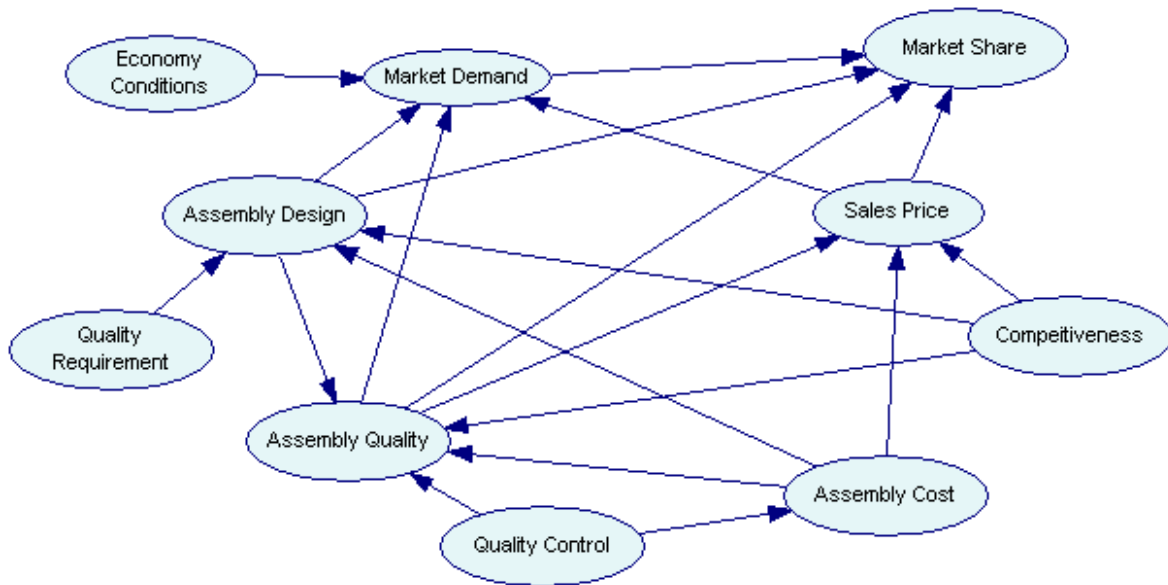
    strFinal.Format(_T("%s"), (LPCTSTR)bstr);

    return strFinal;
}

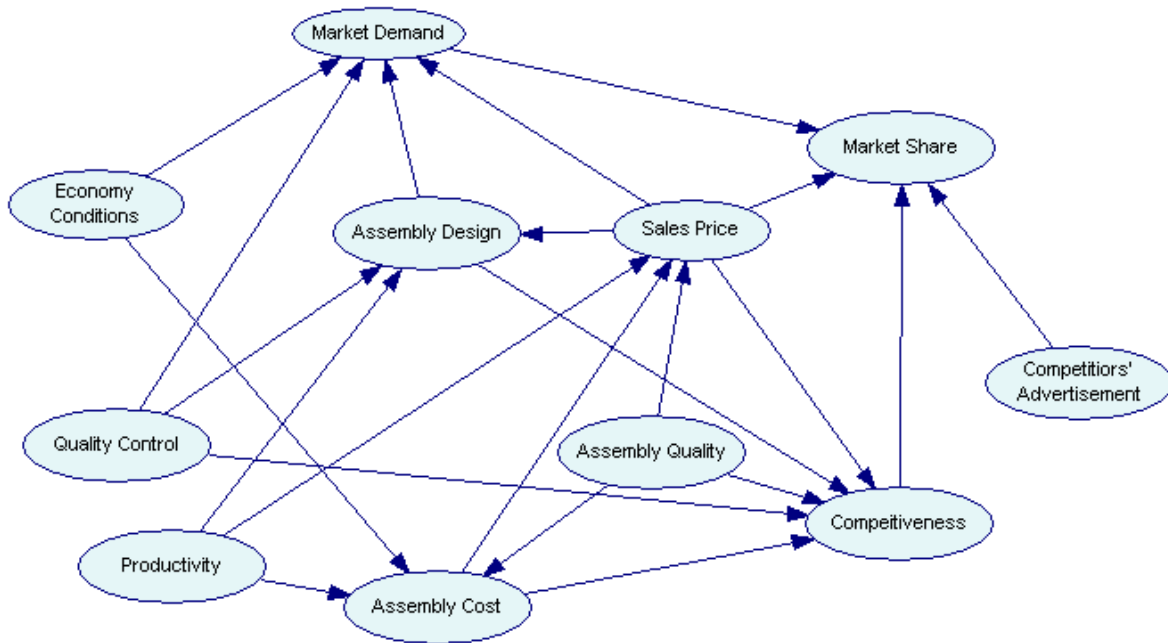
```

APPENDIX D

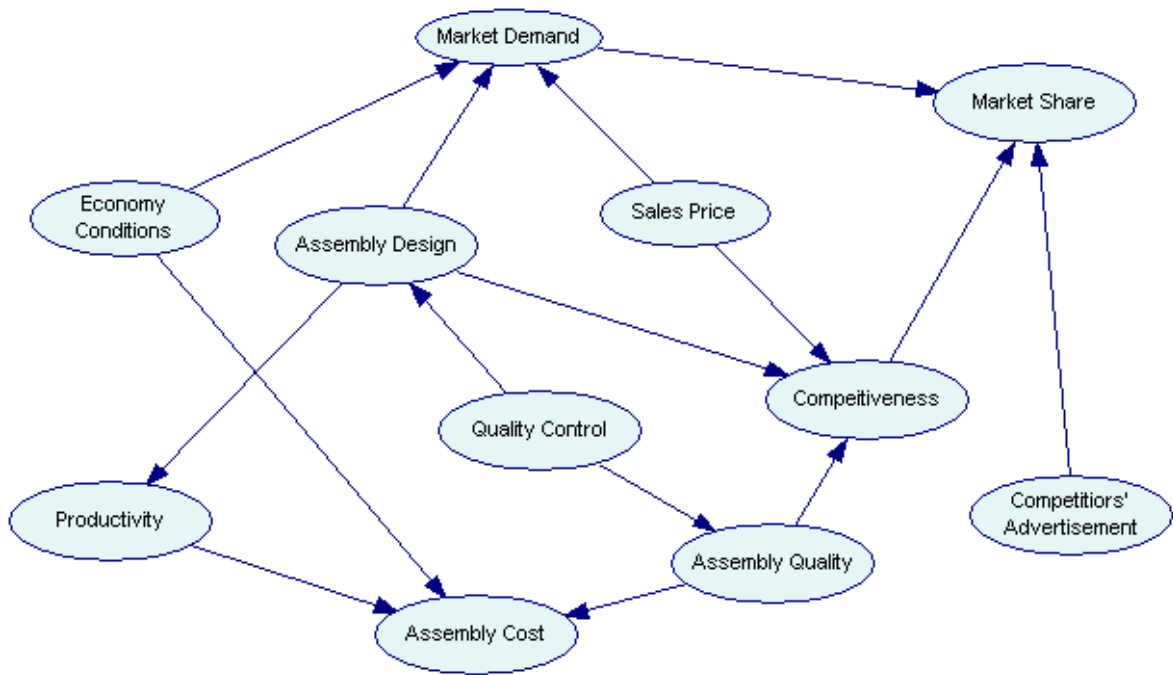
Three knowledge models for Assembly design, Wheel, and Fuel nozzle cases



(a) BBN

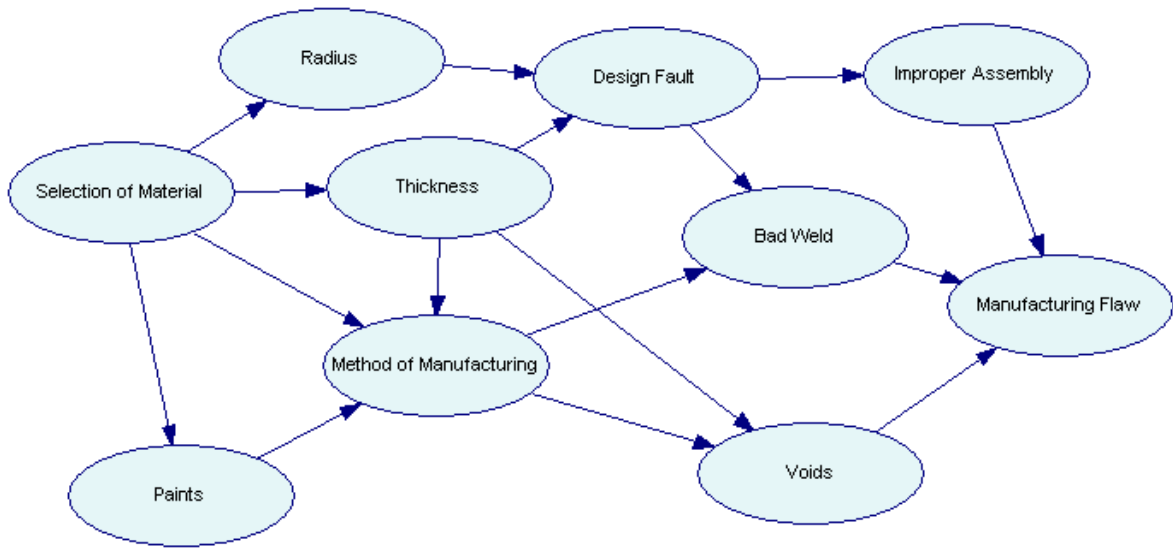


(b) FCM-BBN

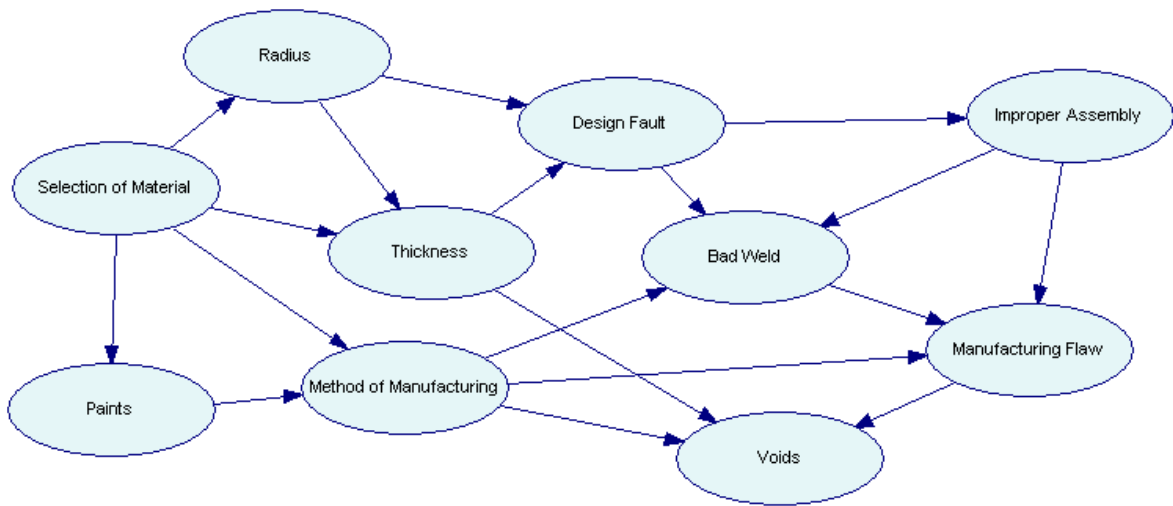


(c) FCM-BBN-M

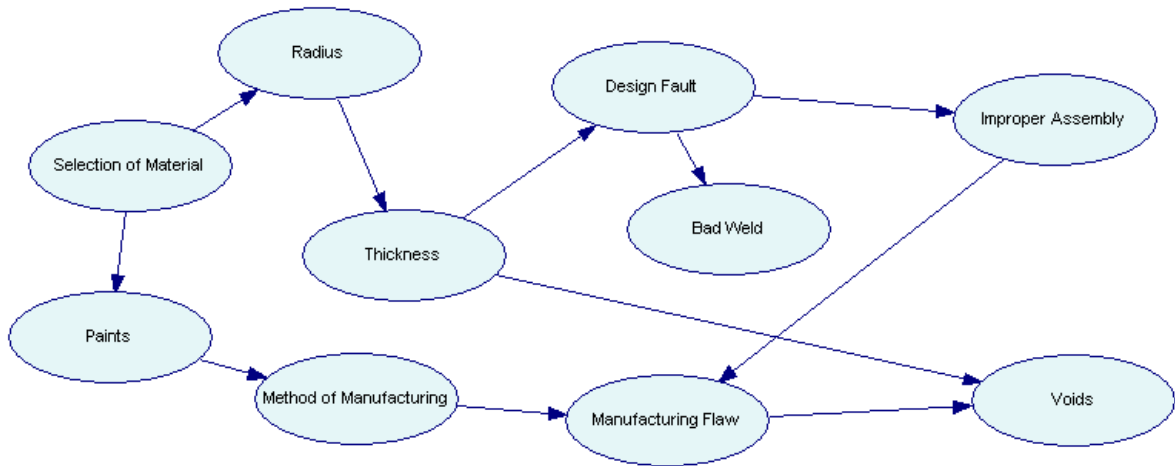
Figure D-1 Assembly design knowledge networks



(a) BBN

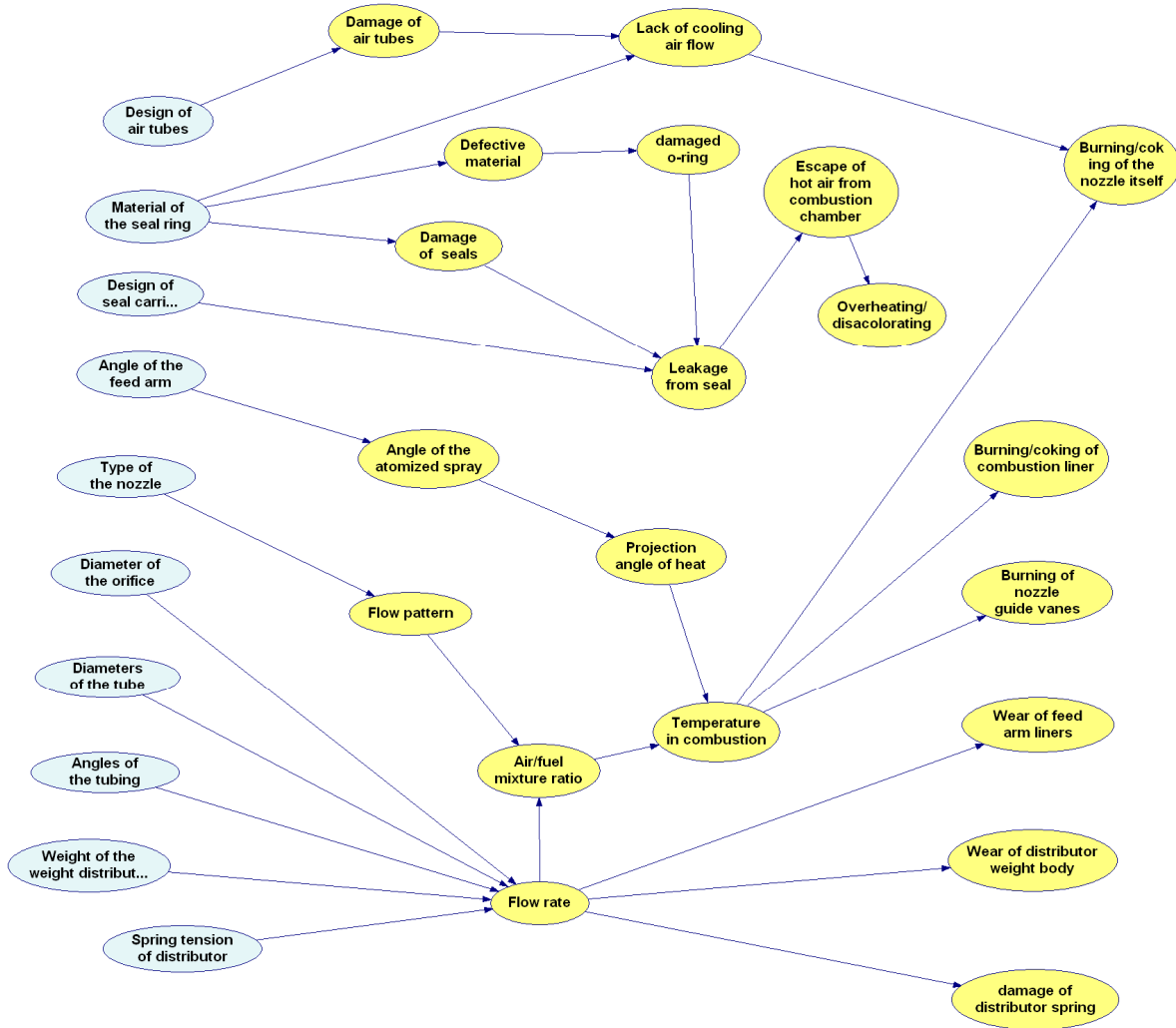


(b) FCM-BBN

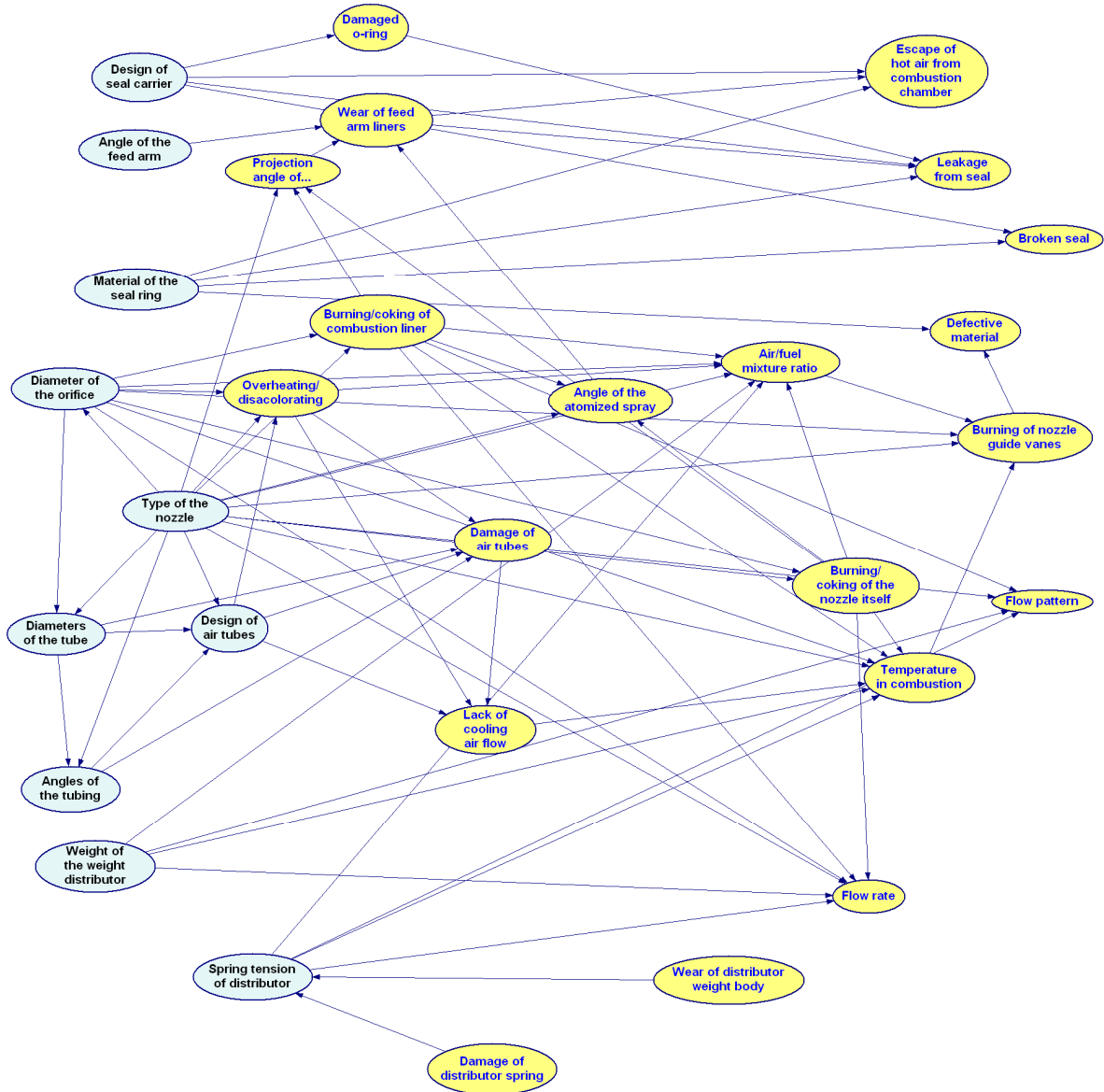


(c) FCM-BBN-M

Figure D-2 Wheel knowledge networks



(a) BBN



(b) FCM-BBN

REFERENCES

- [Alderson 1997] Alderson, J., Clapham, C., and Steel, D., (1997), "Metalinguistic knowledge, language aptitude, and language proficiency," *Language Teaching Research*, Vol. 1, pp 93–121.
- [Arkell 2007] Arkell, D., (2007), "Get our heads into it," *Boeing Frontiers*.
- [Baker 2006] Baker, S.,(2006), "Open-source Moves into Prosthetics," www.businessweek.com/the_thread/blogspotting/archives/2006/02/open-source_mov.html.
- [Barnard 2003] Barnard, Y. and Rothe, A., (2003), "Knowledge management in engineering: supporting analysis and design processes in innovative industries," *Building the knowledge Economy, Issues, Application, Case studies*, IOS Press, pp. 931-938.
- [Barr 1983] Barr, A., and Feigenbaum, E., (1983), "Handbook of artificial intelligence," Los Altos: Kaufmann.
- [Baxter 2007] Baxter, D., Gao J., Case, K., Harding, J., Young, B., Cochrane, S., and Dani, S., (2007), "An Engineering Design Knowledge Reuse Methodology Using Process Modelling," *Research in Engineering Design*, Vol. 18, No. 1, pp 37-48.
- [Berger 1996] Berger, P. L., and Luckman, T., (1996), "The Social Construction of Reality," New York: Anchor Books.
- [Blakler 1995] Blackler, F., (1995), "knowledge, knowledge work and organization: an overview and interpretation," *organization*

studies, 16, pp. 1021-1046.

- [Blanton 2007] Blanton, G., and Burke, F., (2007), "Carrier Team One Knowledge Management: The Challenges," APQC's 12th Annual Knowledge Management Conference and Training, Houston, TX, May7-11.
- [Boot 2007] Boot, L. and Murphy, G.L., (2007), "subtyping as a knowledge preservation strategy in category learning," *Memory & cognition*, Vol. 35, No. 3, pp. 432-443.
- [Briand 1993] Briand, L., Basili, V., and Hetmanski, C., (1993), "Developing interpretable models with optimized set reduction for identifying high-risk software components," *IEEE Transactions on Software Engineering*, 19 (11), pp. 1028 - 1044.
- [Busby 1999] Busby, J.S., (1999), "The Problem with Design Reuse: An Investigation into Outcomes and Antecedents," *Journal of Engineering Design*, Vol. 10, No. 3, pp 277-297.
- [Cheah 2007] Cheah, W.P., Kim, K.Y., Yang, H.J., Choi, S.Y., and Lee, H.J., (2007), "A Manufacturing-Environmental Model using Bayesian Belief Networks for Assembly Design Decision Support," *Lecture Notes in Artificial Intelligence*, Accepted, The 20th Int. Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE 2007), Kyoto.
- [Chen 1993] Chen, J., (1993), "Predicting system based on combining an

- adaptive predictor and a knowledge base as applied to a blast furnace," *Journal of Forecasting*, 12 (2), pp. 93 – 102.
- [Chen 2001] Chen, J., (2001), "A predictive system for blast furnaces by integrating a neural network with qualitative analysis," *Engineering Applications of Artificial Intelligence*, 14, pp. 77 – 85.
- [Cloonan 1993] Cloonan, M.V., (1993), "The preservation of knowledge," *Library Trends*, Spring.
- [Cycorp 2004] Cycorp, Inc. <<http://www.cyc.com/cyc>> May 20, 2004.
- [Daconta 2003] Daconta, M.C., Obrst, L.J., and Smith, K.T., (2003), "The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management," Wiley Publishing, Inc., Indianapolis.
- [Das 2004] Das, B., (2004), "Generating Conditional Probabilities for Bayesian Networks: Easing the Knowledge Acquisition Problem," *Journal CoRR*, cs.AI/0411034, 2004.
- [Davenport 1998] Davenport, T. H., and Prusak, L. (1998), "Working Knowledge," Harvard Business School Press, Boston, Malhotra, Y. *Beyond Hi-Tech Hidebound*.
- [Dean 1989] Dean, T., and Kanazawa, K., (1989) "A model for reasoning about persistence and causation," *Computational Intelligence*, 5(3), pp.142-150.
- [DeLong 2003] DeLong, D. and Mann, T., (2003), "Stemming the brain drain,"

Outlook 2003, No. 1.

- [DeLong 2004] DeLong, d. W., (2004), "Lost knowledge : confronting the Threat of an Aging Workforce", Oxford University Press.
- [Dieter 2001] Dieter, G.E., (2001), "Engineering Design: A materials and processing approach", McGraw-Hill Higher Education, Third edition, USA, ISBN: 0-07-366136-8.
- [Dong 1998] Dong, A and Agogino, A.M., (1998) "Managing design information in enterprise-wide CAD using 'smart drawings'," Computer-Aided Design, Vol. 30 (6), 425-435.
- [Engineous 2005] 2005 Engineous International Symposium & Workshop, Novi, MI. USA, October 10-12, 2005
- [Feigenbaum 1983] Feigenbaum, E.A., and McCorduck, P., (1983), "The Fifth Generation," Addison-Wesley, Reading, MA.
- [Fensel 2001] Fensel, D., (2001), "Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce," Springer-Verlag Berlin Heidelberg.
- [Fikes 1971] Fikes, R.E. and Nilsson, N.J., (1971), "Strips: a new approach to the application of theorem proving to problem Solving," Artificial Intelligence, Vol. 2, pp189–208.
- [Finger 1998] Finger, S., (1998), "Design reuse and design research - keynote paper," In: Engineering Design Conference '98, Brunel University, UK, Professional Engineering Publishing Ltd.

- [FIPER 2001] FIPER, (2001) National Institute of Standard Technolgy Annual Review on FIPER, General Electric Aircraft Engines, Springdale, OH, December 12-13.
- [Fox 1992] Fox, M.S., (1992) "The TOVE Project: Towards A Commonsense Model of the Enterprise," Enterprise Integration Laboratory Technical Report.
- [Fox 1998] Fox, M.S., and Gruninger, M., (1998), "Enterprise Modelling," AI Magazine, AAAI Press. pp. 109-121.
- [Friedman 1998] Friedman, N., (1998), "The bayesian structural em algorithm," In G.F. Cooper and S. Moral, editors, Uncertainty in Artificial Intelligence, proceedings of the Fourteenth Conference, Madison, Wisconsin, Morgan Kaufmann, pp. 129-138.
- [Gable 2005] Cable, G.G., (2005), "The enterprise system lifecycle: through a knowledge management lens," Strategic Change, Vol. 14, Issue 5, pp. 255-263.
- [Gable 2005] Gable G.G., (2005), "The enterprise system lifecycle: through a knowledge management lens," Strategic change, Vol. 14 (5), 255-263.
- [Gazeau 1998] Gazeau, M., (1998), "The management of the Knowledge," Etasts de Veille, pp 1-8.
- [Gilchrist 1993] Gilchrist, W., (1993), "Modeling Failure Modes and Effects Analysis," International Journal of Quality & Reliability

Management, 10(5).

- [Giles 2005] Giles, J., (2005), "Internet encyclopedias go head to head," Nature, Vol. 438, pp. 900-901.
- [Gopnik 2002] Gopnik, A., Glymour, C., (2002), "Causal maps and Bayes nets: a cognitive and computational account of theory-formation", in: P. Carruthers, S. Stich, M. Siegal (Eds.) The Cognitive Basis of Science, Cambridge University Press, pp. 117-132.
- [Gopnik 2004] Gopnik, A., Glymour, C., Sobel, D.M., Schulz, L.E., Kushnir, T., Danks, D., (2004), "A theory of causal learning in children: causal maps and Bayes nets", Psychological Review, Vol. 111 (1), pp. 3-32.
- [Gruber 1993] Gruber, T.R., (1993), "A Translation Approach to Portable Ontology Specification." Knowledge Acquisition. 5(2), pp. 199-220.
- [Gruninger 2003] Gruninger, M., Sriram, R.D., Cheng, J., Law, K., (2003), "Process Specification Language for Project Information Exchange," International Journal of IT in Architecture, Engineering & Construction.
- [Halmos 1960] Halmos, P.R., (1960), "Naive Set Theory," D. Van Nostrand Company, Princeton, NJ.
- [Hansen 2006] Hansen, V.L., (2006), "Functional Analysis," World Scientific, Mathematics.

- [Harrison 1988] Harrison, W., (1988), "Using software metrics to allocate testing resources," *Journal of Management Information Systems*, 4 (4), pp. 93-105.
- [Horváth 1998] Horváth, I., Pulles, J.P.W., Bremer, A.P., and Vergeest, J.S.M., (1998), "Towards an Ontology-based Definition of Design Features." *SIAM Workshop on Mathematical Foundations for Features in Computer Aided Design, Engineering, and Manufacturing*.
- [Horvitz 1998] Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K., (1998), "The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users," *Proceedings of the Conference of the Uncertainty in Artificial Intelligence*, pp. 256-265.
- [Howard 1998] Howard, R., (1998), "Uncertainty about probability: A decision-analysis perspective," *Risk Analysis*, 8, pp. 91-98.
- [Huang 1999] Huang, G.Q. and Mak, K.L., (1999) "Design for manufacturing and assembly on the Internet," *Computers in Industry*, Vol. 38 (1), 17-30.
- [Huang 2000] Huang, G.Q. and Mak, K.L., (2000), "WeBid: A Web-based Framework to Support Early Supplier Involvement in New Product Development," *Robotics and Computer Integrated Manufacturing*, Vol. 16 (2-3), 169-179.

- [ISIGHT 2002] ISIGHT, (2002), International iSIGHT Users' Conference and FIPER Workshop, Washington, D.C, July 15 -18.
- [Iyer 2005] Iyer, N., Jayanti, S., Lou, K., Kalyanaraman, Y., and Ramani, K., (2005), "Shape-based searching for product lifecycle applications," *Computer-Aided Design* 37, pp. 1435-1446.
- [Jordan 1998] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., Saul, L.K., (1998), "An introduction to variational methods for graphical models," In M.I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, the Netherlands.
- [Kan 2001] Kan, H.Y., Duffy, V.G. and Su, C.J., (2001), "An Internet Virtual Reality Collaborative Environment for Effective Product Design," *Computers In Industry*, Vol. 45 (2), 197-213.
- [Kim 1983] Kim, J.H., Pearl, J., (1983), "A computational model for combined causal and diagnostic reasoning in inference systems," In proceedings IJCAI-83, 190-193, Karlsruhe, Germany.
- [Kim 2008] Kim, Y.S., Kim, K.Y., Cheah, W.P., Yang, H.J., (2008), " CAUSAL DESIGN KNOWLEDGE ACQUISITION BY CONSTRUCTING BBN THROUGH FCM," *Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, August 3-6, Brooklyn, New York, USA
- [Kitamura 2002] Kitamura, Y., Sano, T., Namba, K., and Mizoguchi R., (2002) "A

Functional Concept Ontology and Its to Automatic Identification of Functional Structures.” *Advanced Engineering Informatics*, 16(2), pp. 145-63.

[Kitamura 2003] Kitamura, Y., and Mizoguchi R., (2003) “Ontology-based Description of Functional Design Knowledge and Its Use in a Functional Way Server,” *Expert Systems with Application*, 24(2), pp. 153-66.

[Kitamura 2004a] Kitamura, Y., and Mizoguchi R., (2004), “Ontology-based Systematization of Functional Knowledge,” Unpublished manuscript for *Journal of Engineering Design*.

[Kitamura 2004b] Kitamura, Y., Kashiwase, M., Masayoshi, F., and Mizoguchi, R., (2004), “Deployment of an Ontological Framework of Function Design Knowledge,” Unpublished manuscript for *Advanced Engineering Informatics*.

[Knowledge 1999] *Knowledge Management: Strategic Information Systems for the New World of Business*, Working Paper, BRINT Research Institute, 1999.

[Kosko 1988] Kosko, B., (1988), “Hidden patterns in combined and adaptive knowledge networks”, *Proceedings of the IEEE International Conference on Neural Networks*, pp. 377-393.

[Kosko 1995] Kosko, B., (1995), “Combining fuzzy systems”, *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 1855-

1863.

- [Kosko 1997] Kosko, B., (1997), "Fuzzy Engineering", Prentice-Hall, Englewood Cliffs, NJ.
- [KPMG 1998] KPMG Management Consulting. Case Study: Building a Platform for Corporate Knowledge, 1998.
- [Kumar 1994] Kumar, V., Glicksman, J. and Kramer, G.A., (1994), "A SHARED Web To Support Design Teams," in IEEE Proceedings of the Third workshop on Enabling Technologies: Infrastructure for collaborative Enterprises, April 17-19, Morgantown, West Virginia, pp.178-182.
- [Lanubile 1997] Lanubile, F., and Visaggio, G., (1997), "Evaluating predictive quality models derived from software measures: lessons learned," Journal of Systems and Software, 38, pp. 225 – 234.
- [Lauritzen 1988] Lauritzen, S.L., Spiegelhalter, D.J., (1988), "Local computations with probabilities on graphical structures and their application to expert systems (with discuss)," Journal of the Royal Statistical Society, Series B, 50(2), pp. 157-224.
- [Lauritzen 1995] Lauritzen, S.L., (1995), "The EM algorithm for graphical association models with missing data," Computational Statistics and Data Analysis, 19, pp. 191-201.
- [Ledbetter 2007] Ledbetter, D., and Dixon, N., (2007), "Core Practices to Elicit and Convert Tacit Knowledge from Subject Matter Experts to

Explicit Knowledge," APQC's 12th Annual Knowledge Management Conference and Training, Houston, TX, May7-11.

[Liebowitz 2001] Liebowitz, J., (2001), "Knowledge Management: Learning from Knowledge Engineering," CRC press.

[Lin 1996] Lin, J., Fox, M. S., and Bilgic, T., (1996), "A Requirement Ontology for Engineering Design," Concurrent Engineering: Research and Applications. Sage Publications, Inc. 4(3), pp. 279-91.

[Liu 2001] Liu, Z.Q., (2001), "Causation, Bayesian networks and cognitive maps", ACTA Automatica Sinica, Vol. 27 (4), pp. 552-566.

[Long 1989] Long, W., (1989), "Medical Diagnosis Using A Probabilistic Causal Network," Applied Artificial Intelligence, 3(2-3), pp. 367-383.

[Lutters 1997] Lutters, D., Streppel, A.H., Kals, H.J.J., (1997), "The role of information structures in design and engineering processes," 3rd Workshop on Product Structuring.

[Malhotra 1999] Malhotra, Y., (1999), "Beyond Hi-Tech Hidebound, Knowledge Management: Strategic Information Systems for the New World of Business," Working Paper, BRINT Research Institute.

[Mandler 2004] Mandler, J. M., (2004), "The Foundations of Mind: Origins of Conceptual Thought," New York, NY: Oxford University Press.

[Markus 2001] Markus, M.L., (2001), "Toward a theory of knowledge reuse:

types of knowledge reuse situations and factors in reuse success,” *Journal of Management Information Systems*, Vol. 18, No. 1, pp 57-93.

[Maryam 2001] Maryam, A., and Dorothy, E.L., (2001), “Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues,” *MIS Quarterly*, 25(1), pp. 107-136.

[Matsumoto 2005] Matsumoto, I.T., Stapleton, J., Class, J., and Thorpe, T., (2005), “A knowledge-capture report for multidisciplinary design environments,” *Library Hi Tech News incorporating On-line and CD notes*, Vol. 9, No. 3, pp. 83-92.

[McCarty 1996] McCarty, E.D., (1996), “Knowledge as Culture – The new sociology of knowledge,” London and New York: Routledge.

[McQueen 1998] McQueen, R., (1998), “Four Views of Knowledge and Knowledge Management,” in *Proceedings of the Fourth Americas Conference on Information Systems*, E. Hoadley and I. Benbasat (eds.), pp. 609-611.

[Mizoguchi 2003] Mizoguchi, R., (2003), “Tutorial on Ontological Engineering Part 1: Introduction to Ontological Engineering,” *New Generation Computing*. Ohm-Sha & Springer, 21(4), pp. 365-84.

[Nadkarni 2004] Nadkarni, S., and Shenoy, P., (2004), “A Causal Mapping Approach to Constructing Bayesian Networks”, *Decision*

Support Systems, Vol. 2, pp.259-281.

- [Nahapiet 1998] Nahapiet, J. and Ghoshal, S., (1998), "Social Capital, Intellectual Capital and the Organizational Advantage," in Lesser, Eric L.(ed.) 2000. Knowledge and social Capital: Foundations and Applications. Oxford: Butterworth-Heinemann.
- [Neville 1986] Neville, M., (1986), "Monitoring behavior and supervisory control," Handbook of perception and human performance, New York, Wiley-Interscience, pp.40-51.
- [Niebel 1994] Niebel, B.W., (1994), "Engineering Maintenance Management," Marcel Dekker.
- [O.Dell 1998] O.Dell, C., and Grayson, C. J., (1998), "If Only We Knew What We Know: Identification and Transfer of Internal Best Practices," California Management Review 40(3), pp. 154-174.
- [O'Hara 2002] O'Hara, K. and Shadbolt, N., (2002), "Managing Knowledge Capture Economic Technological and Methodological Considerations," Technological and Methodological Considerations. Technical Report, Dept of Electronics & Computer Science, University of Southampton.
- [Open 2008] Open Prosthetics, (2008), "The Open Prosthetics Project: An Initiative of the Shared Design Alliance," www.sharddesign.org.
- [OpenMoko 2008] OpenMoko, (2008), OpenMoko, www.openmoko.com.
- [Patton 1989] Patton, R.J., Frank, P.M., and Clark, R.N., (1989), "Fault

diagnosis in dynamic systems, theory and application," Control engineering series, London, Prentice Hall.

- [PDSEC 2007] 2007 PDSEC Workshop on Systems Engineering, PLM and its impacts, May 27,2007, Detroit, MI, USA.
- [Pearl 2000] Pearl, J., (2000), "Causality: Models, Reasoning, and Inference," Cambridge University Press.
- [Pearl 1986] Pearl, J., (1996), "Fusion, Propagation, and Structuring in Belief Networks," Artificial Intelligence, 29(3), pp. 241-288.
- [Pearl 1998] Pearl, J. (1998), "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," Morgan Kauffmann, San Mateo, CA.
- [Polanyi 1958] Polanyi, M., (1958), "Personal knowledge," Chicago: University of Chicago Press.
- [Polanyi 1967] Polanyi, M., (1967), "The tacit dimension," London: Routledge & Kegan Paul Ltd.
- [Pomerrol 2001] Pomerrol, J.C., and Brezillon, P., (2001), "about some relationships between knowledge and context," lecture note in computer science, 2116, pp. 461-464.
- [Rangan 2005] Rangan, R.M., Rohde, S.M., Peak R., Chadha, B., and Bliznakov, P., (2005), "Streamlining Product Lifecycle Processes: A Survey of Product Lifecycle Management Implementations, Directions, and Challenges," Journal of Computing and

- Information Science in Engineering, Vol. 5, Issue 3, pp. 227-237.
- [Rebane 1987] Rebane, G., and Pearl, J., (1987), "The Recovery of Causal Poly-trees from Statistical Data," Proceedings, 3rd Workshop on Uncertainty in AI, (Seattle, WA) pp. 222-228.
- [Roberts 2003] Roberts, M.L., and Ashton, R.H., (2003), "Using declarative knowledge to improve information search performance," Journal of the American Taxation Association, 25(1), pp. 21-38.
- [Romhardt 1997] Romhardt, K., (1997), "Processes of Knowledge Preservation: Away from a Technology Dominated Approach," 21st Annual German Conference on AI'97, Freiburg, Germany, September.
- [Ruggles 1998] Ruggles, R., (1998), "The State of the Notion: Knowledge Management in Practice," California Management Review 40(3), pp. 80-89.
- [Ryle 1949] Ryle, G., (1949), "The concept of mind," London: Hutchinson & Company.
- [Schacter 1996] Schacter, D. L., (1996), "Searching for memory: The brain, the mind, and the past," New York: Basic Books.
- [Schlenoff 1999] Schlenoff, C., Ivester, R., Libes, D., Denno, P., and Szykman, S., (1999), "An Analysis of Existing Ontological Systems for Applications in Manufacturing and Healthcare," NISTIR 6301, National Institute of Standards and Technology, Gaithersburgh, MD.

- [Shahin 1999] Shahin, T.M.M., Andrews, P.T.J., and Sivaloganathan, S., "A design reuse system," Proc. the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture. Vol. 213, No. 6, pp 621-627, 1999.
- [Shneiderman 2007] Shneiderman, B., (2007), "Creativity Support Tools: Accelerating Discovery and Innovation," Communications of the ACM, Vol. 50, No. 12, pp. 20-32.
- [Simani 2006] Simani, S., and Fantuzzi, C., (2006), "Dynamic system identification and model-based fault diagnosis of an industrial gas turbine prototype," Mechatronics, 16, pp. 341–363.
- [Skaanning 2000] Skaanning, C., Jensen, F., and Kjaerulff, U., (2000), "Printer Troubleshooting Using Bayesian Networks," In R. Loganantharaj, G. Palm, Eds., Intelligent Problem Solving, Methodologies and Approaches, LNCS 1821, pp. 367-379.
- [Spirtes 1991] Spirtes, P., and Glymour, C., (1991), "An algorithm for fast recovery of sparse causal graphs," Social Science Computer Review, 9, pp. 62-72.
- [Spirtes 1993] Spirtes, P., Glymour, C., and Scheines, R., (1993), "Causation, Prediction, and Search," New York: Springer-Verlag.
- [Sriram 1993] Sriram, D. and Logcher, R., (1993), "The MIT Dice Project," IEEE Computer, Vol. 26 (1), PP. 64-65
- [Suh 2000] Suh, E., Youn, S., and Yoo, K., (2000), "Development of a

Methodology for Building a Knowledge Map,” proceedings of IFORMS/KORMS 2000.

- [Szykman 2001] Szykman, S., Sriram, R. D., and Regli, W. C., (2001), “The role of knowledge in next-generation product development systems,” *Journal of Computing and Information Science in Engineering*, 1, pp. 3-11.
- [Taber 1987] Taber, W.R., Siegel, M., (1987), “Estimation of expert weights with fuzzy cognitive maps”, *Proceedings of the IEEE International Conference on Neural Networks*, pp. 319-325.
- [Taber 1991] Taber, W.R., (1991), “Knowledge processing with fuzzy cognitive maps”, *Expert Systems with Applications*, Vol. 2 (1), pp. 82-87.
- [Taber 2007] Taber, R., Yager, R.R., Helgason, C.M., (2007), “Quantization effects on the equilibrium behavior of combined fuzzy cognitive maps”, *International Journal of Intelligent Systems*, Vol. 22 (2), pp. 181-202.
- [Tang 2007] Tang, A., Nicholson, A., Jin, Y., and Han, J., (2007) “Using Bayesian Belief Networks for Change Impact Analysis in Architecture Design,” *Journal of Systems and Software*, 80(1), pp. 127-148.
- [Tapscott 2006] Tapscott, D. and Williams, A.D., (2006), "Wikimomics: How Mass collaboration Changes Everything," Penguin Group (USA).
- [Thierry 1993] Thierry, M.C., Salomon, M., Van Nunen, J., and Van

- Wassenhove, L., (1993), "N. Strategic production and operations management issues in product recovery management," Rapport, Note(s) 19, pp. 1989-1993.
- [Ullman 1997] Ullman, D.G., (1997), "The mechanical design process," 2nd ed. New York, McGraw-Hill.
- [Uschold 1998] Uschold, M., King, M., Moralee, S., and Zorgios Y., (1998), "The Enterprise Ontology" The Knowledge Engineering Review. 13, Special Issue on Putting Ontologies to Use.
- [Verma 1990] Verma, T., and Pearl, J., (1990), "Equivalence and Synthesis of Causal Models," Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, Cambridge, MA, pp. 220-227.
- [Vesely 1981] Vesely, W.E., Goldberg, F.F., Roberts, N.H., and Haasl, D.F., (1981), "Fault Tree Handbook," NUCLEAR REGULATORY COMMISSION WASHINGTON DC.
- [W3C-WWW 1992] World Wide Web Consortium, (1992), "World Wide Web-Summary."
- [Wales 2005] Wales, J., (2005), "Internet encyclopaedias go head to head," Special Report December 2005, Nature, 438, pp. 900-901.
- [Wan 1999] Wan, Y.H., Marzuki, K., and Syed, A.F.S.Z., (1999), "Transformer fault diagnosis using fuzzy logic interpretations," Instrument Asia Technical Symposium'99, Singapore, p. 10.
- [Yoo 2006] Yoo, K., (2006), "Knowledge-based Knowledge Management

System deploying Ubiquitous computing Technologies: Toward the Intelligent and Autonomous Knowledge Acquisition,” Unpublished doctoral dissertation, Pohang University of Science and Technology, Pohang, South Korea.

[Zack 1998] Zack, M., (1998), “An Architecture for Managing Explicated Knowledge,” Sloan Management Review.

ABSTRACT

CAUSAL PRODUCT KNOWLEDGE MANAGEMENT

by

YUN SEON KIM

December 2010

Advisor: Dr. Kyoung-Yun Kim

Major: Industrial Engineering

Degree: Doctor of Philosophy

The US engineering industry base is facing a significant loss of knowledge due to large numbers of employees retiring in the next decade. Problems in various product developments including product design may arise when the expertise is no longer available or the knowledge is forgotten. Also, most of product design knowledge is not reusable, because product design knowledge in an organization remains un-codified. Generally, knowledge-based system can solve or infer these problems. However, knowledge-based systems have been developed solely through the use of rule-based approach, which allows for easy modeling of expert reasoning, but such an approach is not general and for a specific use; thus, existing experience and analyses show that this approach has serious limitations on associations between observable findings and diagnostic hypotheses. Furthermore, the product development knowledge cannot be appropriately acquired, represented, and reused by these techniques. To address these

challenges, this research develops new methodologies and tools to capture, represent, store, and reuse domain knowledge from experts and implement a novel web-based causal product design knowledge management system to systematically utilize the knowledge from experts, who are currently working or retired. The particular emphasis is on these research areas: 1) design knowledge acquisition, 2) causal knowledge representation, 3) causal knowledge evaluation and index, 4) causal knowledge integration, 5) and causal design knowledge management system.

AUTOBIOGRAPHICAL STATEMENT

Yun Seon Kim is a Research Associate of the Department of Industrial and Manufacturing Engineering, Wayne State University. He has been involved in product development knowledge management research for the past four years. His areas of research interest are in product development decision making, telerehabilitation, artificial intelligence, semantic web, and knowledge management. He has a Ph.D. in Industrial Engineering from the Department of Industrial and Manufacturing Engineering, Wayne State University, an MSc. in Information Sciences from the School of Information Sciences at the University of Pittsburgh, and B.As. in Computer Sciences and Management from Handong Global University in Pohang, South Korea.