PREDICTING AND MANAGING SYSTEM INTERACTIONS AT EARLY PHASE OF THE PRODUCT DEVELOPMENT PROCESS

by

QI DONG

B.S., Mechanical Engineering
University of Kentucky, 1997

S.M., Mechanical Engineering
Massachusetts Institute of Technology, 1999

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUNE 2002

Signature of Author…….
Department of Mechanical Engineering
February 22, 2002

Certified by…….
Dr. Daniel E. Whitney, Senior Research Scientist
Center for Technology, Policy, and Industrial Development
Thesis Supervisor

Certified by……..
David Wallace, Associate Professor of Mechanical Engineering
Chairman, Doctoral Committee

Certified by……..
Warren P. Seering, Professor of Mechanical Engineering
Committee Member

Certified by……..
Steven D. Eppinger
Professor of Management Science and Engineering Systems
Committee Member

Accepted by…….
Ain A. Sonin
Department of Committee on Graduate Students

Predicting and Managing System Interactions at Early Phase of the Product Development Process

by

QI DONG

Submitted to the Department of Mechanical Engineering
on February 22, 2002 in Partial Fulfillment of the
requirements for the Degree of Doctor of Philosophy in
Mechanical Engineering

ABSTRACT

The activity of designing and developing large, complex, discrete, physical, and engineered products faces the challenges in the physical product system, the organization of people, and the larger systems in which the product resides—the natural and societal systems.

This thesis defines system interactions as the interactions amongst design variables within the physical product. Knowing system interactions early in the product development process is critical for project management, design concept selection, and system architecture decisions. However, existing methods that address the system interactions issues, such as the Design Structure Matrix (DSM), are good analysis tools, but cannot be used during conceptual synthesis when the most important decisions about the system designs are made.

System level knowledge is defined as the knowledge concerning system interactions. System level knowledge is organizational knowledge that resides in the collective minds of members in the organization. System level knowledge is critical to the success of the design of large systems, yet is often missing due to its empirical nature. A knowledge management framework was proposed in this thesis and tested in industry cases from Ford and CVC.

This thesis developed a method to predict and analyze system interactions at early phase of the design process. The method transforms an Axiomatic Design's Design Matrix (DM) into a DSM based on solving systems of linear equations using substitution. Since a DM is more easily constructed during early design phases, we can use this method to obtain a DSM during concept design. Consequently, the advantages of the DSM system analysis tools and methods can be applied to make better decisions on system design, system architecture, and project management.

The method was tested using two industry cases at CVC and Johnson and Johnson Ortho-clinical Diagnostics. Both case studies showed that the method was effective in real engineering projects. Further observations in the case studies also revealed that a DSM could also be easily converted back into a DM. The interchangeability between DSM and DM

allows engineering organizations to predict system interactions early on in a project, while capturing and managing system level knowledge throughout the product lifecycle.

Thesis Supervisor: Dr. Daniel E. Whitney
Title: Senior Research Scientist, Center for Technology, Policy, and Industrial Development

Acknowledgement

As I am finishing up the writing of this thesis, I come to realize that the completion of this piece of work is more than my one person's effort. Without the support and help from many people and organization, I would not have arrived at this point.

First, I would like to thank NSF Engineering Research Center program for providing the necessary funding for this research (Cooperative Agreement No: EEC-9529140). I would also like to thank the MIT Center for Innovation in Product Development (CIPD) for providing me the opportunities to collaborate with industries. Product development research is not the kind of research one can do by sitting in a room and thinking up some ideas. Product development researchers must get out to the real engineering world to identify where the problems are, and why they exist. Product development research results also must be able to withstand the test of case studies and eventually the implementation in real engineering projects. CIPD has been a great support in providing the right environment for product development researches, and facilitating the network of a group of passionate researchers and industry practitioners. I cannot say enough thanks for how much I have benefited in this environment.

Next, I would like to thank my thesis advisor Dr. Daniel E. Whitney. I could not have asked for better guidance, help, and support than what Dr. Whitney has given to me starting at the formation of the ideas of this thesis research until all the way through the writing of this thesis. Yet, looking beyond the completion of this piece of research, the more important things that I am taking away with me by working with Dr. Whitney in the past five year are how to think as a researcher. I learned from him how to believe yet doubt everything that is said, and how to trust oneself yet be critical to oneself. I learned from him that life is a continuous learning process, and everything can be relevant to everything else. I learned from him to be confident in myself yet be modest. All of these lessons, I believe, will take me far in life beyond what a PhD diploma can do. For this, I owe great debts to Dr. Whitney.

I would also like to thank the rest of my thesis committee members—Professor Warren Seering, Professor Steve Eppinger, and Professor Dave Wallace for their help and guidance

throughout the thesis research. I benefited greatly for having diverse opinions and comments in each of the committee meeting, which helps me to see the complete picture. I have also many thanks to them for reading and commenting my on my thesis drafts.

The completion of this thesis is impossible without the help from many of the engineers and manager in the case study companies. Although some of the names I will mention below do not work at the same company any more, their support for this thesis cannot be forgotten. At CVC, I want to thank Christine Whitman and Bill Starks for accepting my preliminary research ideas and encouraging me to conduct the first case study in the company. Matthew Coon, Xiangqun (Shawn) Chen, Bill Messner, and Dino Mastrosimone helped me to gather data and were very supportive and encouraging to my research. At Johnson and Johnson Ortho Clinical Diagnostics, I want to first thank Brian Blaser for recognizing the value of my research work and introducing me to the company executives. Laura LaBauve, the OASIS program manager, Jeff Helfer and Ted Farrel, the OASIS systems engineering managers provided me all the necessary help and support for my case study. I also want to thank Mark Raymond and the OASIS system engineering team for their help.

I want to dedicate this thesis to my late grandfather Zhenshun Dong (董振舜). Grandpa had always been a role model for me in my life. He was a medical doctor, a well-respected senior chemical engineer, and a loving grandpa. I still remember the nice time and good conversations we shared since I was a little kid. When I just started learning reading, I used to spend all my days indulging in children's storybooks. In the evening, when grandpa came home from work and we sat together under the starry night sky, I would make up stories based on what I read in the day. He was always the patient audience for all my childish elaborations and reflections on things and people. He left us a year too early to see me finishing this thesis. But I know somewhere up there, he is probably smiling at this piece of not-so-childish-any-more work. Or maybe he is smiling because when I told him those stories I made up, I did not think they were childish at all.

The completion of this thesis would not have been possible without the rest of my family. My 85-years old grandmother Shuhe Hou (侯淑和) handled grandpa's death with a very

strong heart. She has become an example for me in the past year to stay strong and finish my research work. I am so glad she will be present at my hooding ceremony. My mother Dalin Ma (馬大林), father Shaojing Dong (董紹靜), and brother Huayi (驊怡) have always provided me support and encouragement. My fiancé Onno endured years of separation and traveling with still so much love to support me so that I could focus on my thesis. Finally, this thesis work is coming to an end, and we will start a new chapter in our lives.

Of course, many of the thanks shall also go to my numerous friends and colleagues who helped me and encouraged me in the past five years. My accomplishment here is a collective effort, and I am in debt with everyone who had a part of this experience.

Qi Dong (董騏)

February 20, 2002

Table of Contents

List of Figures

List of Tables

List of Acronyms

DM: Design Matrix (from Axiomatic Design)

DP: Design Parameter (from Axiomatic Design)

DSM: Design Structure Matrix

ESC: Electro-static Chuck

FR: Functional Requirements (from Axiomatic Design)

HAMG: Hazard Analysis and Mitigation Document (from Johnson and Johnson case study).

JNJ: Johnson and Johnson

MOCVD: Metal Organic Chemical Vapor Deposition (from Guru Prasanna's CVC project)

OCD: Ortho-clinical Diagnostics

PRD: Product-level Requirements Document. May also be used to refer to the actual product requirements (from Johnson and Johnson case study).

SSRD: Subsystem-level Requirements Document. May also be used to refer to the actual subsystem requirements (from Johnson and Johnson case study).

1 Introduction

1.1 Research Motivation

> *Changes in the World's political, economic and technological realms in the past century have placed great stresses on approaches to management and system design. The changes that have had the greatest impact are the increase in size and complexity of the human organizations and technical systems needed in the world today, and the rate of change in the external environment with which these organizations and systems must cope.*
>
> ***Joel Moses, Provost, MIT*** *[Hughes 1998 p.3]*

1.1.1 Living with Systems

We live in a universe of systems. The systems around us include three categories: The natural system, the technological system, and the societal system. Human beings can determine technological system's and societal system's structures. Therefore, these two systems can be viewed as man-made systems in contrast to natural system whose existence and behavior are not dictated by human wishes [Simon (1981)].

The natural system is the most basic system we are in. Human bodies are biological and chemical systems, which must be viewed as a whole consisting of interacting parts [Zimmer (1999), Weng, et al. (1999), Koch, et al. (1999), Systems Biology (2001)]. Human, other living beings, and the environment form the ecological system [Service (1999), Parrish, et al.g (1999)]. The ecological system on the earth is greatly influenced by the solar system we are in [Werner (1999)]. The sun affects the health of the lives on earth. The moon changes the tide in the water bodies and maybe even the mental state of human beings. The solar system is only a small part of the Milky Way Galaxy. It rotates round the center of the Milky Way together with other stars. The relative motion of the Milky Way changes with respect to

other galaxies, indicating interactions between our galaxy and the rest of the universe [Kaufman and Freedman (2000)].

Living within the natural system, humans created man-made systems. To maintain the stability of their lives, humans created the societal system such as the political and economical systems. To make their lives more convenient, humans developed technological systems, such as the airplanes, the automobiles, the telephone network, etc. Historian of Technology Thomas P. Hughes (1998 p.3) observes that by the twentieth century, "Americans had transformed a natural world into a human-built one characterized by technological systems and unmatched complexity." This trend is continuing in the 21st Century.

In the process of developing large complex systems, human beings have learned through mistakes about the interactions between the societal system and the technological system, and the interaction between the man-made systems and the natural system. The environmental regulations are the responses from the societal system in order to control the effect of the technological system on the natural system. Marketing is the work to bridge the societal system and the technological system. In short, the man-made systems and the natural system are highly interactive. Anything we humans do must be put into the context of the systems.

1.1.2 Challenges in the Design and Development of Large Complex Systems

The interest of this thesis is on man-made systems. The natural system exists regardless of human wishes. It is the duty of scientists to observe and discover the laws governing the behaviors of the natural systems. As an engineering thesis, this research concerns how humans can best design and develop man-made systems, especially large complex ones.

Large complex man-made systems are interesting because of two reasons. First, in the past century, many large complex man-made systems have fundamentally changed the way we live. For instance, the automobiles and the airplanes have improved people's mobility. The

telephone system has changed people's ability of communication, and signal is forever separated from its carrier [Mindell (2000)]. The satellite system has enhanced our understanding to the atmosphere and changed way we view the world. People desire more, better, and bigger such systems in the future. The second reason for which large complex man-made systems are worth studying is the challenges we face in developing such systems. One good example can be found in Walton's *Car* (1997) that depicts the trouble and chaos Ford Motor Company went through in developing the Taurus. Yet, despite the challenges, people still want these products, and want them even bigger and better. Companies want to design and develop these systems better, faster, and cheaper. Therefore, it is worthwhile to understand the challenges in the design and development of large complex systems, so that we can discover ways to overcome them.

This section summarizes some of the challenges found in the literatures on developing large complex systems. The list below is not intended to be exhaustive, but rather to bring the readers' awareness of the issues, and to direct the readers to see the issues in the light of systems.

1.1.2.1 Understanding the Challenges from Systems' Perspective

Hughes and Hughes (2000) observe, "After World War II, a systems approach to solving complex problems and managing systems came into vogue among engineers, scientists, and managers." A systems perspective is critical because large complex systems are not only systems themselves, but also interact with other major systems (Figure 1-1). The successful design and development of large complex systems require a mix of knowledge from natural science, social science, and engineering technology [Rechtin (1991) p. xiii-xiv]. No one person in any of these fields can do the job alone.

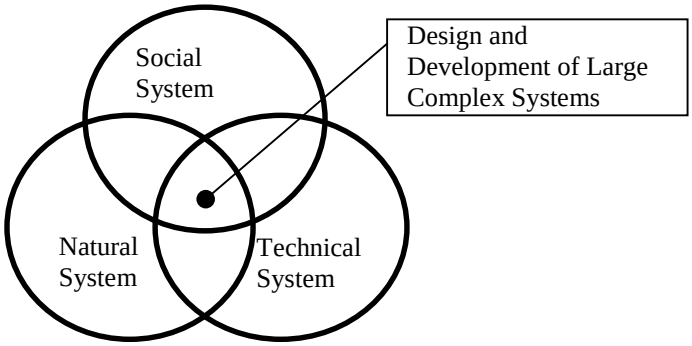


Figure 1-1: System View of Product Development

Yet, each of us was trained to know only some aspects of one or more circles in Figure 1-1. The challenge of taking a system perspective is to stay unbiased by our own background, and to stay aware of all aspects of the problem. We must consider the impact of all three systems in Figure 1-1 on the object we are designing and developing. As Rechtin (1991 p.3) puts it, "complex product design has many constraints imposed on them. Hence a technical optimum may not be politically or economically optimum." We must be aware of the constraints imposed on a system by the "larger world" the current system is in.

1.1.2.2 The Interface Dilemma

One of the most significant characteristics of any system is the interconnection among a set of elements [Rechtin (1991), Alexander (1968), Simon (1981)]. Grady (2000) points out "it is through these interfaces that a system attains its superiority over an unorganized collection of things (p. 272)." The richer the complexity, the more synergy we get from the system. Yet, dealing with large amount of system interfaces is challenging for human beings. The difficulties come from both individual human beings' cognitive capability and how people work together in organizations.

Individual human beings are incapable of dealing with large complex system interfaces without proper outside aids. Miller's research (1956) indicates seven plus minus two is the

limit for average people to deal with items and their relationships along a single dimension or system attribute without any outside aids. Complex systems usually consist of more than seven interfaces and more than one system attribute, and hence are difficult for people who try to understand and manage them. As evidence, Sterman (2000 p.669) observed in many decision making process, the complexity of the feedbacks among the variables makes it impossible to think of the optimal strategy. Usually, people resort to heuristics and rules of thumb. Furthermore, as Simon (1957 p. 198) points out, "the capacity of the human mind for formulating and solving complex problem is very small compared with the size of the problem whose solution is required for objectively rational behavior in the real world or even for a reasonable approximation to such objective rationality." The recognition of Bounded Rationality won Simon the Nobel Prize in economics in 1979. Hence, in order to work on large complex systems, we must understand human limitations and develop tools to help ourselves overcome the limitations.

Due to the limitation of individual human beings, complex systems are usually worked on by organizations of people. Complex systems are challenging for human organizations. To divide the work among the people in an organization, complex systems are typically decomposed into many pieces called subsystems or components [Alexander (1968), Simon (1981), Rechtin (1991)]. One or more people from various disciplines and backgrounds work on various pieces of the system. To manage the people in the organization, each organization typically has a hierarchical reporting structure. The structure of the organization usually does not match the architecture of the physical system being designed. The reason is that an organization can work on more than one system. Organizing the people differently for each product is impractical. Therefore, each group in the organization must collaborate to correctly address the system interface and system integration issues.

The first challenge regarding organization communication is what information is exchanged among groups of people. Each person may form a different picture about the system based on his/her view. Sterman (2000) calls these "pictures" the mental model (p.694). The mental models will determine how information is used and how decisions are made. Different mental models among the people who work on the same system will result in the so-called

"Spread-thinking", where people try to discuss an issue using different definitions [Warfield (1995)]. Information may get lost or misunderstood. The result is frustration among the people and sub-optimal or even incorrect decisions by the groups.

The second challenge regarding organization communication is how effectively the communication can take place. Allen (1977) pioneered a stream of research to investigate how effective communications stimulate the performance of development organizations, and hence the resulting performance of the systems they are working on. Allen (1977 and 1997) suggests the distance between the people affects how much they would communicate with each other. Ulrich and Eppinger (2000) emphasize the need to facilitate the exchange of essential information in order to speed up the development process. Griffin and Hauser (1992) show that using Quality Function Deployment (QFD) practices enhances technical communication within the boundaries of the development teams, but reduces the communication levels across team's boundaries. Sosa (2000) further reveals the following observations regarding the relationship between the product architecture and the communication between development teams:

1. *The distributed nature of the integrative systems forces design teams to overcome organizational barriers in order to handle design interfaces with all the systems. That is, effects of organizational barriers are more severe among teams that design modular systems.*

2. *Design interfaces across modular systems are more difficult for design experts to recognize than interfaces with integrative systems.*

3. *Design teams handle some design interfaces according to their type. We found that spatial-type design interfaces are largely addressed in the design of modular systems while transfer-type design interfaces are more likely to be handled in the design of integrative systems.*

Therefore, human beings have limitations in dealing with complex system interfaces when working individually or as groups. We must acknowledge the limitation and find ways to improve the situation.

In summary, the interface dilemma shows the incompatibility between the natural system (humans) and the man-made system. On one hand, complex system interaction is what makes systems more desirable than individual parts. On the other hand, due to the inherent human limitation, complex system interfaces are difficult for us to deal with. Finding the right balance between the system interface complexity and the human capability is the dilemma.

1.1.2.3 Emergent Property

Emergent property is the same as saying the whole is more than the sum of parts [Bertalanffy (1968) p.55]. In other words, emergent property of the system rises from the interactions of its lower level entities, none of which shows it. For instance, the properties of gases are not the properties of the individual gas molecules. The function of the brain is reflected in the structure makeup, rather than individual neural cells [Koch et al. (1999) p. 97].

Emergent property of the system has two sides. On the positive side, this characteristic makes systems more desirable than the collection of its components. When designing man-made systems, we can design the system components to interact in such a way that the system as a whole delivers desirable functions that its components cannot.

On the negative side, this property brings the possibility that some of the system behaviors may not be what was initially intended. A more severe implication is that the designers of the system may not be able to predict all of the system behaviors. The investigation of Apollo 13's problem revealed exactly this point [Compton (1989), NASA Apollo 13 (2001)]. An Oxygen tank from previous mission was used on Apollo 13 whose heating switch was operating under 28 volts instead of the new standard 65 volts. The operation voltage difference should not have mattered if the Oxygen tank were operating normally. However,

a loose fitting tube caused the Oxygen tank not to empty properly during testing prior to flight. The tank was heated up in order to empty the Oxygen. When the tank was heated up to 80 degree F, the heater switches in the tank opened. The current generated by 65 volts rather than 28 volts caused arcing between the two ends of the heater and permanently welded them together. So the heater in the Oxygen tank B was constantly working when they should not be. The overheated components melted the Teflon insulation, and eventually caused explosion when exposed to the pure Oxygen. From Apollo 13's lesson, we learned that system behaviors are not always predictable. The emergent property of systems can come from long chain of causes. Some of the causes may occur only accidentally, and therefore making the end result surprising and dreadful.

1.1.2.4 Dynamic Behavior

Senge (1994 p.71) and Sterman (2000 p. 21) point out there are two types of complexity. One is the detailed complexity, and the other is the dynamic complexity. Detailed complexity is about system variables and the possible combinations among them. An example of detailed complexity is airline scheduling. Dynamic complexity is about the outcomes of system interactions over time. A system with only a few elements (low in detailed complexity) can give high dynamic complexity. A good example is the Beer Game [Sterman (1989)]. Suh (1999) also suggested the concept of time dependent complexity. He says the future could be unpredictable and introduce a lot of uncertainties into the system, which are not initially considered by the designers. Yet the outcome of the system will still be dependent on past decisions made.

Thus the design and development of large complex systems must take into consideration the dynamics of the natural and societal systems. The customer requirements and market needs may change. The competitors may introduce new products. The technology may become obsolete. We must recognize the dynamic environment we are in and try to discover and use tools to help us better achieve our goals.

1.1.2.5 Both Science and Art

The design and development of large complex systems are both science and art. Although scientists have learned a lot about this world, we really don't have a complete understanding about how everything works in this universe. The capability of science is limited. As Rechtin puts it, "Single-value optimization and multi-value trade-offs are not sufficient for design situations…in dealing with complex system, analytical knowledge is not enough. Experience and judgment are necessary (2000, p.XV, p.6)." Therefore, the field of the design and development of large complex systems is challenging and interesting. We must continuously learn from our experience and develop new methods and tools. We must revise or abandon obsolete methods and tools when necessary. We must stay open-minded and acknowledge the need to learn from other unfamiliar fields.

1.2 Research Goals

1.2.1 Research Scope

This research focuses on the design and development of large complex products, specifically discrete engineered physical products, such as automobiles, airplanes, copy machines, etc. Therefore, the object of study here is the technical system (Figure 1-1), not the natural system or the societal system. In addition, products such as books and clothing are not the focus of this study because they are not large complex systems. Software and service are also not the emphasis here because they are not discrete physical products.

Nonetheless, narrowing down the research scope is only to focus the study. This research takes a general systems approach. The results of this research can be the foundation for improving the design and development of other systems and products.

1.2.2 Challenges in Product Development

Product development is about orchestrating two large complex systems—the large complex product and the organization that develops the large complex product. The product development process must take into consideration the coordination between the above two

systems. Prior discussions on the challenges in developing large complex systems are applicable to product development. Furthermore, these challenges can be detailed for product development situations in the following list [Ulrich and Eppinger (2000) p.6]:

1. Trade-offs: recognize the product itself is a system. One attribute may be optimized at the expense of another attribute.
2. Dynamics: recognize the uncertainties the societal, natural, and other technical systems may introduce over time.
3. Details: recognize the complexity of decision-making in product development. One must take a system perspective and consider all three systems in Figure 1-1. One must consider the dynamics. This challenge is caused by human limitation in managing detailed complexity as mentioned in the "*The Interface Dilemma"* section.
4. Time Pressure: This is a pressure coming from the societal system where we want to deliver products to the customers before other competitors. This pressure also comes from the dynamic environment the product development is in. Technology and the market change fast. If we do not deliver the product quickly, the technology may become obsolete and the market may not be there any more. Time pressure may force decisions to be made without complete information. Therefore, the design and development of large complex products are both science and art.
5. Economics: This challenge reveals the fundamental relationship between the societal and the technical system (Figure 1-1).
6. Creativity and Team Management: This is also the challenge caused by the interactions between the technical system and the societal system. Different from the economical challenges, this challenge comes from the recognition that humans are the ones that develop the technical systems. The products can only be good if the people are doing the right thing. Better team management and team diversity can address "The Interface Dilemma". Allowing creativity is acknowledging the need for both art and science in product development.

This list may not be exhaustive. Yet it is enough for us to conclude that the design and development of large complex products face the same challenges as dealing with system

complexity. Therefore, this thesis takes a systems perspective to address the issues in product development. Not only existing product development methods, but also system engineering methods are examined in the Literature Review section of this thesis.

1.2.3 Research Questions

Issues relevant to the design and development of large complex products are numerous. The main research questions in this thesis are:

1. How to predict system interactions in the product at early stage of the design process?
2. How to manage the system level knowledge in the organization?

A series of sub-questions are listed below for each of the main research questions.

1.2.3.1 Obtaining System Interactions at Early Stage of the Design Process

1.2.3.1.1 What are System Interactions

System interactions here are defined as the interactions among the key design variables in the system of interest. In this study, the systems are large complex products. The design variables can be either physical parts or the features and design parameters of the product. For instance, let's take an automobile as a system. The door and the body frame have interaction between them. They must be designed to fit snugly. Therefore, the physical parts in the system--the door and the body frame--are the design variables. The system interaction between these design variables is a packaging relationship. Take another example; the engine must be designed so that it has enough power to pull the body weight. Therefore, the body frame cannot be too heavy for the engine chosen, or the engine cannot be too small for the body size. In this case, the design variables are the engine power and the body weight, which are design parameters rather than physical parts. The system interaction between these two design variables is an energy relationship.

The tasks in product development are not the design variables in this thesis. Examples of tasks are "create CAD models", "order material from vendor", "build prototype", etc. Hence, the system interactions studied here are the interactions among design variables in the product, not the tasks involved in creating the product. To understand how to manage the tasks in a product development project, the readers may want to consult literatures such as Eppinger's DSM research [Eppinger et al. (1994)], Gantt charts, Pert charts, Critical Path, and other project management techniques [Ulrich and Eppinger (2000) p. 321].

In addition, the system interactions here are not the direct interaction between people involved in the design and development of the product, such as the interactions studied by Allen (1977 and 1997). However, this is not to say that human interactions have nothing to do with the system interactions here. From the discussion in the "The Interface Dilemma" section, the system interface in the physical product will affect how the people who are designing the product should interact with one another. Furthermore, although this thesis focuses on the design and development of physical products, the same technique can be used to design organization structures. When the system is the human organization and the design variables are the people in the organization, then the interactions among the people become the system interactions.

In short, this thesis takes a product-view of the product development process. The author believes that the interactions in the product itself should determine how the tasks in the project need be completed and how people need to interact during product design and development. This product-view is a hypothesis. One of the future tasks generated from this thesis can be to test this hypothesis. Nonetheless, this hypothesis does not affect the research in this thesis. This thesis is based on the fact that large complex products are systems. Taking a systems view on the product helps to design and develop the product better and faster.

1.2.3.1.2 Why It is Important to Obtain System Interactions Early

> *The biggest mistakes in any large system design are usually made on the first day.*
>
> ***Dr. Robert Spinard***
> *Vice Presendent of Xerox Corporation*
> *[Hooks and Farry (2000) p. 3]*

Imagine you are going to an unfamiliar town. What is the best way to get to your destination? It is probably using the help of a map of that town. A map can help you to avoid unnecessary detours and reach the destination quickly.

Knowing system interactions about a product is like having the map of a town. Large complex products are complex systems. It is a common knowledge in product development that as time elapses in a project, the flexibility for design change decreases and the cost of design change increases (Figure 1-2). If we do not put in effort to understand system interactions early, we may discover that we designed the product without knowing some of the important system interfaces. Late design changes can have ripple effects throughout the entire system and affect every engineering department. Such changes take a lot of time and money to implement. Besides project cost and schedule overrun, if not all relevant departments in the organization are informed about the design change, product quality problems may arise, and cause the company to lose revenue. Therefore, the earlier we know about system interactions in the product, the better decision we can make about the design. Consequently, late design changes because of the discovery of new system interfaces may be reduced. Yet, we must realize that due to the emergent properties of systems, it is impossible to predict all of the system interfaces upfront. The next section will carry out further discussion regarding this point.

Knowing system interactions early is important for project management. The design and development of large complex products usually takes the work of more than a dozen to

hundreds people over the time span of several months to several years. The Interface Dilemma tells us that human beings need outside aids to deal with the complexity in the products. When system interactions are known, there exist methods to predict project length, budget estimation, and how to facilitate communications between people and different groups (see sections about DSM and other project management tools in the Literature Review chapter). Therefore, the earlier we know about system interactions, the better management decisions we can make for the product development process.

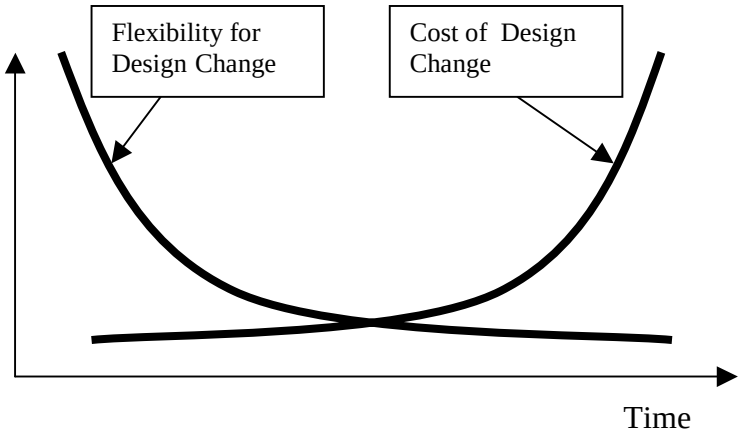


Figure 1-2: Design Changes in Product Development Process

Knowing system interactions can help concept selection and system architecture decisions. When comparing different concepts for a large complex product, the design that makes the system interactions simpler is the more desirable design. Being able to tell what system interactions are in the system will help us to choose the right concept to start. Of course, sometimes due to technology risk, market trend, etc., we may end up choosing a concept that has more complicated system interactions. Then knowledge of system interactions becomes even more important to help us navigate through the project more easily.

1.2.3.1.3 The Possibility of Obtaining All System Interactions from Early On

Based on the previous discussion on system emergent property, it is impossible to obtain all system interactions during early phase of the design process. We will always learn about more new system interactions as the project moves on. So is it still necessary to even try to predict system interactions?

The answer is yes. Managers and engineers are making decisions without complete information at early phase of the design process. The Interface Dilemma says any tool and method that can help the decision making when dealing with large complex system is needed. With all the benefits of knowing system interactions early, it is still worth the work to find ways to predict system interactions from early on. How helpful this prediction can be will be one of the questions this thesis research will look at.

1.2.3.1.4 Research Questions Under This Topic

Before inventing anything new, the first step should be to take a look at what existing methods are there and whether they serve the purpose of predicting system interactions. This will include an analysis of the strengths and weaknesses of the existing methods. The Design Structure Matrix (DSM) method is of particular interest here because it has shown tremendous capability in dealing with system interactions. Therefore, its strengths, limitations, and the ability to predict system interactions are reviewed. If there are no existing methods that can predict system interactions, then what method can be developed? How well does the method work? How complete is the prediction?

The research questions related to predicting system interactions at early phase of the design can be summarized into the following list.

Q1-a. What methods have been used in the past to capture system level interactions? What are the strengths and weaknesses of existing methods? Is DSM a good way to predict system level interactions? What are its strengths and limitations?

Q1-b. How to predict system interactions early? How to predict system interactions for new technology?

Q1-c. If we can predict system interactions, how complete is the prediction?

1.2.3.2 Managing System Level Knowledge

1.2.3.2.1 What is System Level Knowledge

System level knowledge is a concept only applicable to large complex systems. The design and development of large complex systems rely on the work of more than a dozen to hundreds of people. These people are from different disciplines and they work on different parts of the system. For instance, in the automotive industry, industrial designers work on the styling of the car. Mechanical engineers work on the body frame, engine, etc. Electrical engineers work on the electrical system, motors, etc. Material scientists work on the material selection for safety. Such division of work is based on disciplines. Another way of specialization is based on product attributes. There are engineers who specifically deal with fuel economy, Noise-Vibration-Harshness, crash testing, etc. The cause of such division of labor is the human's cognitive limitation in dealing with large complex systems. Each person must focus on portions of the systems instead of the whole system in order to deliver useful work. The result of such labor division in the design and development of large complex systems is that no one single person knows everything about the system. The knowledge about the system resides in many people's head.

We can define the knowledge each person has about his/her piece of the system as "Component Level Knowledge." Component level knowledge is something one individual can easily get his/her arms around without relying on other people. The "System Level Knowledge" is the knowledge about the entire system, which must be learned through collaboration with other people. System Level Knowledge consists of the following four parts:

1. What the system components are;
2. How system components interface with each other to achieve the desired functions and the undesired system behaviors;

3. Who has the knowledge about each system component;
4. Where to find the documented knowledge about each system component.

1.2.3.2.2 Why Pay Attention to Managing System Level Knowledge

The management of system level knowledge involves capturing, storing, and providing easy retrieval of system level knowledge. From the discussion about what system level knowledge is, we can conclude that system level knowledge is organization knowledge, not individual's knowledge. Because of the human cognitive limitation, no one person can remember everything about a large complex system. The company must initiate the effort to document system level knowledge. Yet, companies currently are doing poorly on documenting system level knowledge. A study at Ford Motor Company [Dong (1999)] revealed system level knowledge is not well documented (see Figure 1-3). Ford Motor Company relies heavily on experts to address system level issues. When Figure 1-3 was shown to representatives from other non-automotive industries, the author was told time after time that it was true for their companies too.

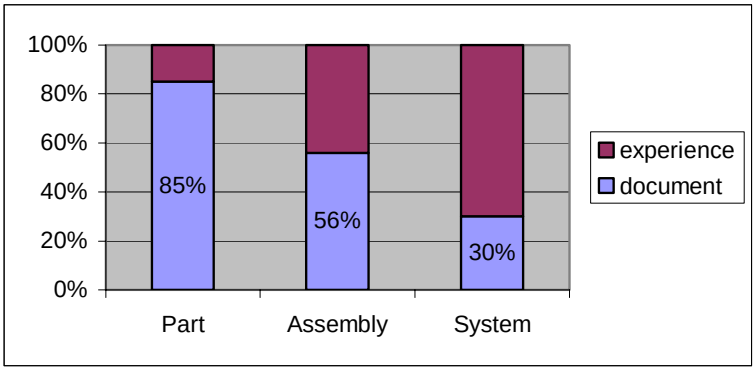


Figure 1-3: Where was Ford Throttle Body Design Knowledge [Dong (1999)]

Poor documentation of system level knowledge naturally turns the control of the system to the subjective empirical knowledge of experts. Relying on people's experience is important, but also could cause many problems in a company that designs large complex systems. The discussion earlier on the Interface Dilemma applies here. People have "Bounded

Rationality." Each person sees only part of the entire system. The inconsistent mental pictures among people that are working on the system together can cause misunderstandings and cause delays in the project and even quality problems in the product. In addition, People can change jobs, move away, or retire. When system experts leave the organization, they take away a large portion of the system level knowledge. The replacement has to learn everything again from scratch. Unfortunately, the system level knowledge takes many years to learn by experience. Therefore, the frequent change of personnel for the system engineering position causes inefficiency in the organization, which is highly undesirable in today's competitive product market.

In the 1980s and 1990s, companies recognized the problem with the dispersion of expert knowledge in the companies, and started the Knowledge-based engineering (KBE) effort [Sferro (1999), Whitney (1999)]. KBE has not made large impact on the design of large complex products because the system level knowledge is simply not there to be programmed into software [Whitney (1999)]. The first step for KBE is to understand and manage knowledge at system level rather than hurry into software coding.

In addition, if we just record all the learning experiences in text or drawings, the documentation for a large complex system can grow to an enormous amount, which may be very difficult to find information for future reuse. We can offer help through a knowledge and document-browsing tool. To fundamentally solve the problem, a better framework on what knowledge to document and in what format is needed. Therefore, it is important to provide a good system level knowledge management strategy so that the organization knowledge is not only well documented, but also done in a way for easy retrieval.

In short, system level knowledge is poorly documented in companies. Relying on expert's experience to deal with large complex systems limits companies' ability to deliver quality product quickly to market. Good system level knowledge management can enable companies to take advantage of the Knowledge-based engineering tools to improve their system design and shorten design cycle. Therefore, managing system level knowledge is an

important research topic. We must also not forget to document system level knowledge for the purpose of retrieval.

1.2.3.2.3 Research Questions Under This Topic

First of all, this research is interested in collecting data similar to Figure 1-3 in other companies in order to see if the lack of documentation of system level knowledge is a common problem in different companies and different industries. Second, we must survey the existing literature to find out what has been done on managing system level knowledge already. Third, knowledge management inevitably run into the challenge of persuading engineers to document. Engineers working on large complex systems are usually busy with the project and have little interest in additional documentation. This research is interested in investigating ways to encourage engineers to document. The research questions are summarized below:

Q2-a. What has been done in managing system level knowledge?

Q2-b. Is there a better way to capture, store, and represent system level knowledge?

Q2-c. What are the best sources of information for predicting system interactions?

Q2-d. How companies are doing with managing system level knowledge?

Q2-e. How to encourage engineers to document system level knowledge? Make recommendations to the management.

1.3 Research Approach

In order to answer the above research questions, this thesis research developed a matrix transformation method, which allows us to obtain a Design Structure Matrix (DSM) at early phase of the design process from a Design Matrix (DM) used in Axiomatic Design [Suh (2000)]. The construction of a DM is feasible at early phase of the design process, because the inputs for the DM—product requirements and how the elements in the design concept meet the requirements—are available early on in the design process. Transforming the DM into a DSM allows us to predict system interactions before committing detailed design work. The prediction DSM captures the system interactions that are driven by design requirements.

System analysis carried out from the prediction DSM can facilitate us to have a requirement-driven design process.

In addition a framework for managing system level knowledge is proposed. The matrix transformation method fits in as part of the knowledge management framework, and enable the capturing of system interaction knowledge starting early in the design process. Both the matrix transformation method and the knowledge management framework will be detailed in Chapter 3.

In order to test the effectiveness of the above two proposals, and to investigate the research questions posed earlier, two case studies are carried out at CVC, and Johnson and Johnson Ortho Clinical Diagnostics division. From the two case studies, we found answers to some of the research questions and were able to ask better questions for future researches.

1.4 Thesis Structure and Chapter Outlines

1.4.1 Thesis Structure

This thesis concentrates on answering the research questions posed earlier. Here is a summary of all the questions:

Q1: How to predict system interactions early in the product development process?

- *Q1-a. What methods have been used in the past to capture system level interactions? What are the strengths and weaknesses of existing methods? Is DSM a good way to predict system level interactions?*
- *Q1-b. How to predict system interactions early? How to predict system interactions for new technology?*
- *Q1-c. If we can predict system interactions, how complete is the prediction?*

Q2: How to manage system level knowledge?

- *Q2-a. What has been done in managing system level knowledge?*
- *Q2-b. Is there a better way to capture, store, and represent system level knowledge?*

Q2-c. What are the best sources of information for predicting system interactions?

Q2-d. How companies are doing with managing system level knowledge?

Q2-e. *How to encourage engineers to document system level knowledge? Make recommendation to the management.*

The table below lists which questions each chapter intends to address. At the end of the thesis, not all questions have an answer. Some of the questions are re-written into better questions for future research.

Chapter Number	Research Questions Investigated
Chapter 2 Literature Review	Q1-a, Q2-a
Chapter 3 Research Method	Q1-b, Q2-b, c, d
Chapter 4 CVC Case Study	Q1-b, c; Q2-b, c, d, e
Chapter 5 Johnson and Johnson Case Study	Q1-b, c; Q2-b, c, d
Chapter 6 Status of Research Questions	All
Chapter 7 Conclusions and Future Work	----------

Table 1-1: Research Questions Discussed in Each Chapter

1.4.2 Chapter Outline

Chapter 2 reviews relevant literature, which sets the stage for the need of this thesis research. In this chapter, the existing methods in dealing with system interactions and the existing knowledge management techniques are studied. Their strength and limitations are discussed. The limitations of the existing methods bring the need to new approaches.

Chapter 3 introduces a matrix transformation method that enables us to obtain a DSM from the design requirements. In addition, this matrix transformation is also a part of the system level knowledge management tool proposed in this thesis. The assumptions used in this method are listed. Two case studies were designed to test this method in real engineering projects to find out its feasibility, strengths, and limitations.

Chapter 4 details the first case study conducted at CVC, a semiconductor manufacturing equipment producer. This case study shows the matrix transformation method in Chapter 3 works in a real engineering project. The outcome of the method is a very meaningful DSM, which can be used to help the planning of the system integration phase of the project. The method in Chapter 3 is also further refined for the selection of output variables based on the learning from CVC case study.

Chapter 5 concerns the second case study at Johnson and Johnson Ortho Clinical Diagnostics (JNJ OCD) division. OCD produces automated clinical chemistry systems that analyze patient body serums in hospitals. In this case study, the matrix transformation method introduced in Chapter 3 is tested again on a different product from a different industry. The method produced a useful DSM for the planning of the system integration work. The prediction DSM generated from the matrix transformation method matched the DSM's produced by the JNJ engineers based on their expert experiences. We learned more guidelines about effectively using the matrix transformation method. This case study also revealed that not all requirements can be decomposed and be used to predict system interactions. Hence, the prediction DSM is never a complete view of the system interactions. It must be kept as a live document to react to new knowledge about a system. This observation matches the emergent property of systems. Third, this case study also revealed which existing documents are the best sources for collecting system level knowledge for knowledge management.

Chapter 6 summarizes the learning from the CVC and JNJ case studies. The research questions posed in Chapter 1 are reviewed. What we learned from the case studies about each questions are listed. We found answers for some of the questions, and the rest remain questions. But because of better understanding about system interactions and system level knowledge, we are able to revise some of the questions and pose better ones for future researches.

Chapter 7 discusses how the findings in this thesis research can be applied to solve some of the issues in product development. It concludes the thesis with future research directions.

2 Literature Review

2.1 Systems Engineering

2.1.1 What is Systems Engineering

Systems engineering as a field of research does not have a very long history. Based on Hughes and Hughes (2000 p.1), the *Engineering Index* had no entry for "Systems Engineering" in 1964. By 1969, the number had jumped to eight pages of citation for "System Engineering." Systems thinking came into form during World War II in military realm. After WWII, its applications gradually entered the civil realm. The influence of systems thinking not only reached physicist, mathematicians, and engineers, but also management specialist and social scientists.

Since systems engineering is a fairly recent research field, in various literatures, the definitions of System Engineering are not the same. In 1962 when Hall wrote about this subject, he did not even attempt to give a clean-cut definition (1962 p.4). Thirty-six years later when Blanchard wrote his book on system engineering and management (1998 p. 12), he admitted that there is still variety of approaches. The readers can look through the literatures to see how each of them defined systems engineering. The literature review here combines the viewpoints of many authors and attempts to provide a comprehensive list of what systems engineering does:

1. Systems engineering is to recognize the system being designed and developed is a whole, not just parts [Blanchard (1998) p.13, Chestnut (1965) p. 8, Thome (1993) p. 23].
2. Systems engineering is about tradeoffs of different objectives. One must optimize the overall system rather than parts [Chestnut (1965) p.8, Stevens, et al. (1998) p.4, Westerman (2001) p.6].
3. Systems engineering must pay attention to the dynamics and the lifecycle of the system [Blanchard (1998) p. 13, Chestnut (1965) p.11, Sage and Amstrong (2000)

p.9]. This includes considering the "-ilities" of the system in addition to the function of the system during design. Examples of "-ilities" are reliability, maintainability, etc.

4. System engineering must also address the integration, verification and validation of the systems [Martin (1997) p.3, Stevens, et al. (1998) p.4]
5. Systems engineering must pay attention to not only the system within the boundary, but also the broader system outside the boundary [Chestnut (1965) p.19].
6. Systems engineering is about trying to model and simulate the system in order to predict the system behaviors, and try to control the emergent behaviors. The idea is that it is cheaper and quicker to calculate or measure a model or simulation than to build an actual system [Chestnut (1965) p.21, Thome (1993) p. 23, Stevens, et al. (1998)].
7. Systems engineering must understand the requirements on the system [Blanchard (1998) pp. 13, Chestnut (1965) p. 24, Martin (1997) p. 3, Stevens, et al. (1998) p. 4., Sage and Amstrong (2000) p. 10].
8. Systems engineering develops the architecture of the system and synthesizes parts of the system to meet the various objectives [Chestnut (1965) p. 25, Thome (1993) p. 23, Martin (1997) p. 3, Stevens, et al. (1998) p. 4, Westman (2001) p.6].
9. Systems engineering methods and techniques are independent to the system being studied, whether the system is mechanical, software, or electrical system [Thome (1993) p. 23, Stevens, et al. (1998) p.4].
10. Systems engineering is the management of the technical development of a system. System engineers must communicate across many groups involved in the development. System engineering is a teamwork effor [Blanchard (1998) p. 13, Hall (1962) p. 16, Martin (1997) p.3, Stevens, et al. (1998) p.4, Sage and Armstrong (2000) p.10.].

From the above list, we can see that the kind of work systems engineering does targets at the challenges in the design and development of large complex products (see Chapter 1). Not surprisingly, systems engineering methods are used widely for every phases of the product development process, especially for large complex products.

There are two basic categories of system engineering methods—management and engineering. Martin (1997 p.56) listed the subcategories of each category:

a. Management
   (1) Planning
   (2) Organization
   (3) Control
   (4) Direction
   (5) Integration
b. Engineering
   (1) Requirements Analysis
   (2) Functional Analysis (or Structured Decomposition)
   (3) Architecture Synthesis
   (4) System Analysis and Optimization
   (5) System Element Integration and Verification
   (6) Engineering Documentation

Since system engineering takes a holistic view of the development of large complex products, the results of applying systems engineering methods in either management or engineering category usually affect the decision in the other category. For instance, the N-Square diagram method [Grady (1993)], which is also called the Design Structure Matrix, belongs to the Functional Analysis subcategory, which belongs to the engineering category. The system study using the N-Square diagram may also contribute to the management decisions such as process planning and organization design.

Browsing through the system engineering literatures, we can find a long list of system engineering methods. This thesis focuses on the methods that deal with system interactions among the design variables in the product.

2.1.2 System Engineering Methods that Deal with System Interactions

This section reviews many of the commonly used system engineering methods that addresses the system interaction issues. The methods are briefly introduced, and the pros and cons of each method are examined.

2.1.2.1 Heuristics

The most basic method in dealing with system interactions is the use of heuristics. Rechtin (1996) wrote a book about the heuristics on designing system architecture, most of which concerns how to address issues related to system interfaces. Heuristics are very useful when there is no analytical method available. The common sense learned from past experiences are always good guidelines for future actions.

However, heuristics are usually used at higher level of abstraction during system engineering process than non-heuristic techniques [Martin (1997)]. The analytical capability of the heuristics is limited. The execution of the heuristics also highly depends on individual's understanding, experience, and subjective judgment. We need methods that are more objective. The heuristics do not help with the human cognitive limitation on dealing with large amounts of system interactions. Therefore, a method that can help to record and visualize the complex system interactions is needed.

2.1.2.2 Graphs and Digraphs

Graphs and diagraphs are common techniques used to capture and visualize system interactions. Most system engineering literatures mention these methods [Alexander (1964), Buede (2000) p.91, Sage (1981) p.12, Steward (1982)]. Figure 2-1 shows a diagraph, which is just a normal network graph with directions indicated on the braches.

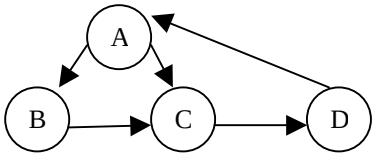


Figure 2-1: A Diagraph

Although good at visualizing system interactions, the graphs lose this advantage when a lot of system interactions have to be shown at once [Dong (1999), Sage (1981), Steward (1981)]. The graphs also cannot easily show indirect interactions such as the one between B and D in Figure 2-1. In addition, graphs, although they help people to visualize the system interactions, they cannot help to analyze the system. People must use trial and error to find the best ways to traverse the graph, which may be a very difficult task when the graph involves a large amount of system interactions.

2.1.2.3 Trees

Trees are special types of graphs (Figure 2-2). All the elements in a tree are arranged hierarchically. Therefore, a tree structure gives the system an order. Typically, trees are used to represent the hierarchical order among subsystem and components that form the system. Trees are also used to present requirements decomposition structure or organization structure.

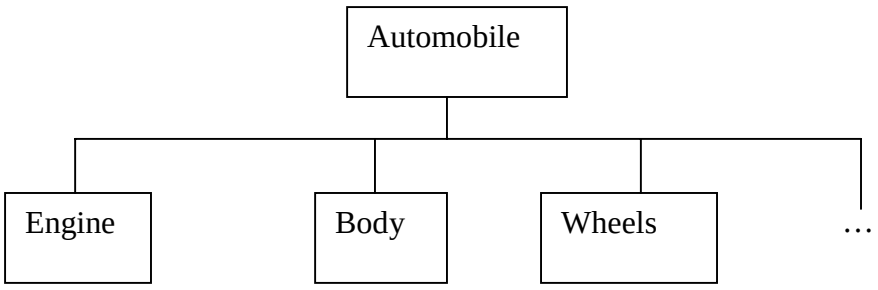


Figure 2-2: An Example of A Tree

However, trees cannot represent the interactions among the elements in a system because they do not allow connections between the leaf-elements and branch-elements. Therefore, the use of trees in helping dealing with complex system interactions is very limited.

2.1.2.4 IDEF

IDEF's roots began to form when the Air Force, in response to the identification of the need to improve manufacturing operations, established the Integrated Computer-Aided Manufacturing (ICAM) program in the mid-1970s. The requirement to model functions (processes), data, and dynamic (behavioral) elements of the manufacturing operations resulted in the initial selection of the Structured Analysis and Design Technique (SADT) method (SADT is a registered trademark of SofTech). SofTech's Doug Ross developed SADT in the early 1970s. A subset of SADT was the basis for the Air Force's ICAM language notation. A major development from the ICAM program was the Integrated DEFinition methodology or IDEF as it is now called [Wisnosky and Batteau (1990, pp. 8-11), Grady (1993 pp.251)].

Figure 2-3 and Figure 2-4 show the typical IDEF block and diagram. The blocks represent process steps. The diagram provides a way to characterize development and manufacturing process.

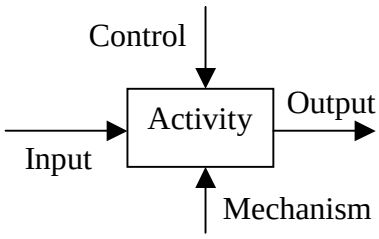


Figure 2-3: A Typical IDEF Block [Grady (2000 pp. 252)]

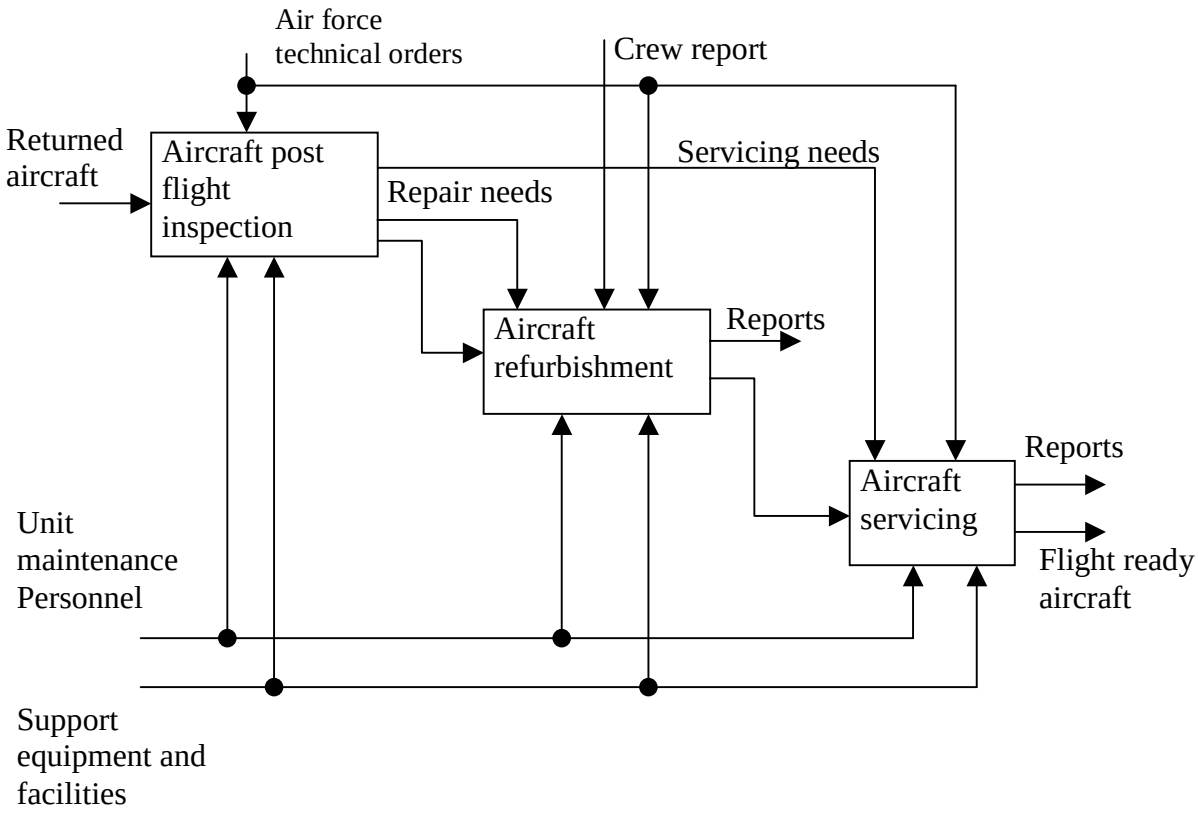


Figure 2-4: Typical IDEF Diagram [Grady (2000 pp. 252)]

IDEF diagrams are suitable for capturing business processes, but not the interactions among the components in a physical product. Like any graph, IDEF diagrams can get very cluttered with interconnecting lines when the system involves a lot of interactions. In addition, IDEF diagrams do not allow feedbacks between tasks. Rework and iterations exist in real projects, and IDEF is not able to capture them.

2.1.2.5 CPM, PERT, and Gantt chart

CPM, PERT, and Gantt chart are popular project management techniques. They are very much related to each other. Each technique is first briefly introduced here. Details about each technique can be found in most project management literatures [Weist and Levy (1990), Moder, et al. (1995), Steward (1981)]:

CPM is short for Critical Path Method. It is a technique developed by Remington-Rand and Dupont. It provides a mechanism for rapid assessment of alternatives and consequences so best actions can be implemented. CPM helps to schedule tasks in parallel. It provides an analytical way to project the completion time of a project based on the time it takes for each tasks to complete.

PERT is short for Program Evaluation and Review Technique. PERT is used to project completion dates with uncertainties factored into elemental activities. It uses the probability distribution of each tasks completion time to make better prediction.

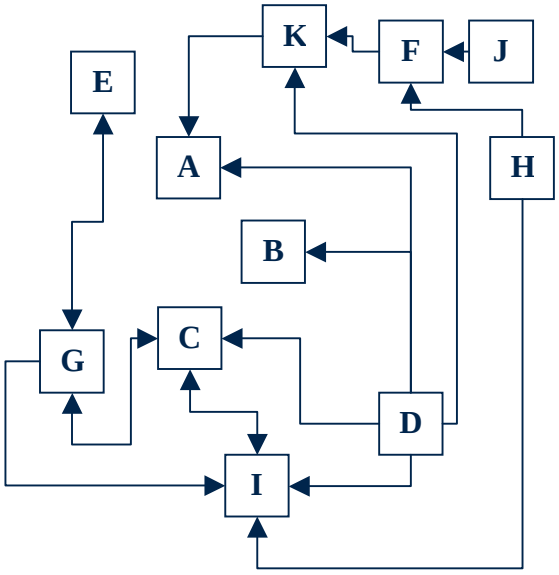
Gantt Chart shows the scheduling of tasks along time axis. It can show tasks that are on the critical path and that are not. Tasks that are not on the critical path have earliest and latest completion time. However, Gantt chart is the output of CPM or PERT. Itself is not an analysis tool.

Like IDEF, all of the above three techniques concern the interactions among the tasks in a project. They are not suitable for studying the interactions among the system elements in a product. The most severe weakness about the above three techniques is that they do not work with processes that include feedback loops. In other word, rework encountered in the actual design process cannot be reflected by these techniques.

2.1.2.6 Design Structure Matrix

Design Structure Matrix method (DSM) has a long history. Steward (1965, 1981) and Warfield (1973, 1976) are the two most important figures that brought the matrix representation of graphs into the area of system engineering and system management. Rogers (1989) developed a piece of software for DSM that was used by NASA projects. Whitney, Eppinger, and their students at MIT applied the method to the design and development of large complex products [Eppinger, et al. (1990)], and started a DSM community among the researchers and practitioners all over the world (MIT DSM website).

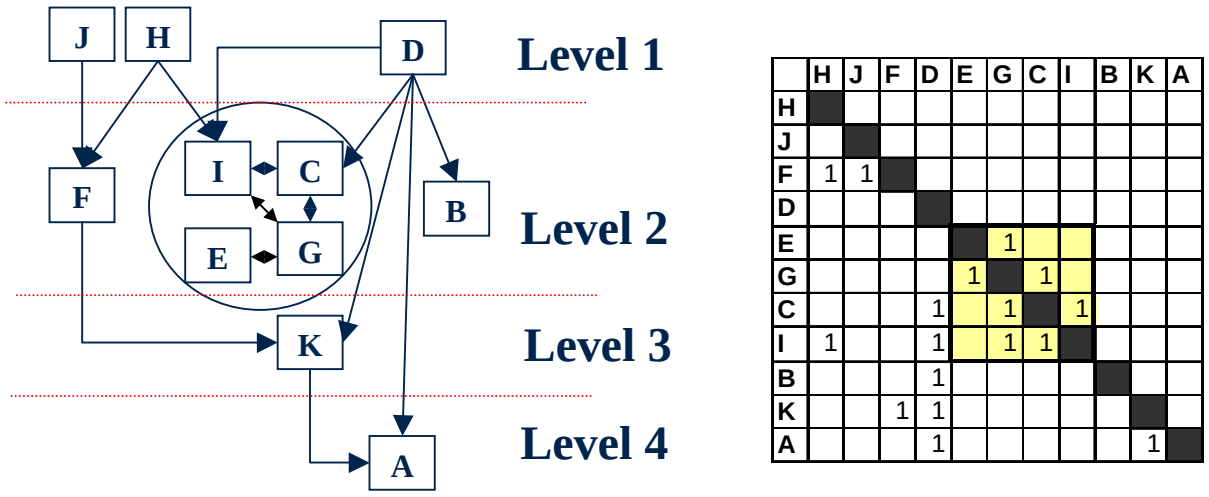
DSM is a matrix representation of the graphs. It has other names such as N-squared diagram, dependency matrix, etc. Figure 2-5 shows an example of the DSM. Each link in the graph is represented by a mark ("1" in this case) in the DSM. For example, in the graph we see an arrow pointing from *D* to *I*. This arrow is presented by a "1" in column *D* and row *I* in the DSM. Therefore, DSM can capture all of the information presented in a graph.



	A	B	C	D	E	F	G	H	I	J	K
A				1							1
B				1							
C				1			1		1		
D											
E							1				
F								1		1	
G			1		1						
H											
I			1	1			1	1			
J											
K				1		1					

Figure 2-5: An Example of a Design Structure Matrix and its Corresponding Graph

In addition, the DSM partitioning algorithm enables us to re-sequence the elements in the graph so that we can find the tasks that are in sequence, in parallel, and iterative [Steward (1965, 1981), Warfield (1973)]. The result of partitioning the DSM in Figure 2-5 is shown in Figure 2-6. The elements in the same level can be completed in parallel. Elements in different levels must be completed in sequence. Elements involved in an iteration block (such as *E*, *G*, *C*, and *I* in this case) must be done concurrently.



	H	J	F	D	E	G	C	I	B	K	A
H											
J											
F	1	1									
D											
E						1					
G					1		1				
C				1		1		1			
I	1			1		1	1				
B				1							
K			1	1							
A				1						1	

Figure 2-6: DSM after Partitioning

The partitioning result of any DSM is unique. It enables us to analytically discover the best approach to deal with complex systems. This technique can be used for both project management and the design of large complex products (see MIT DSM Website for references to papers). The matrix representation of system interactions is preferred over graphs when there are a large amount of system interactions [Dong (1999), Steward (1981)].

In addition to partitioning, many other techniques have also been developed to use DSM as a system analysis method. Steward (1981) developed the tearing method to identify the key elements that determine large iteration loops. Thebeau (2001) presented ways to cluster the elements in a system so as to aid the system architecture decision. Smith and Eppinger (1997) used Eigen value and Eigen vector concept to identify the convergence rate of a DSM, and the controlling system elements in the design iteration. Browning (1998, 1998b, and Eppinger 1998) and Carascosa (1998) developed models to use DSM to predict the probability of completing projects on time and on budget. Dong (1999), Bartowski (2000) and Glynn (2000) used DSM as a way to manage the knowledge about complex system design. Cho (2001, and Eppinger 2001) developed means to apply the analysis results from DSM back to PERT and Gantt charts to facilitate project management.

Therefore, as a system analysis tool, DSM provides many advantages over the previously introduced methods. Furthermore, DSM can also serve as a place to record the system emergent properties discovered throughout the product lifecycle, so that the system can be analyzed again using the DSM techniques and better decisions can be made according to the new situation. However, DSM has its own weakness. Two weaknesses of DSM method are discussed below. Both weaknesses originate from the traditional ways to construct a DSM.

First, current DSM analysis only improves the design of mature products. In previous DSM researches, in order to construct a DSM, researchers and practitioners interviewed people who were experienced with the system of interest. The experts told the researcher what the interactions were based on their experience. The researchers then constructed DSMs based on interview data. The DSMs were then analyzed using one or more of the techniques listed above. Recommendations were then made about the organization structure, product design, or project management.

However, the expert knowledge about the system interactions is difficult to obtain at early phase of the design process for a new product that has never been designed before. Let us borrow Henderson and Clark (1990)'s framework for the types of product innovation (Figure 2-7). The traditional DSM interview method works the best for incremental innovations because the expert knowledge about the system interactions and components can be reused, and DSMs can be easily constructed through interviews. Yet, for modular innovations, although the knowledge about the linkage between the module and the rest of the system is known, the design of the new module is not known. Therefore, DSM technique does not provide a lot of help with the design of the part of the system that is new. Furthermore, for architecture and radical innovations, the knowledge about system interactions is not available at early phase of the design process because past experience about the system is not reusable. Therefore, interviewing experts does not provide sufficient information about system interactions. From the author's personal experience, the experts just could not tell what the system interactions would be during concept development phase of a new product. Consequently, a sufficient DSM cannot be constructed until the product design is already in

the detailed-design phase when the engineers learn most of the system interactions (Figure 2-8). At that time, it is already too late for the DSM research results to have much impact on the current project. What about keeping the DSM results as lessons-learned? This lessons-learned would only be useful when the next generation of the product falls into incremental innovation. If the next product is modular, architectural, or radical innovation, little previous DSM study can be applied again.

Linkage between Core Concepts and Components	Core Concept: Reinforced	Core Concept: Overturned
Unchanged	**Incremental Innovation** (DSM)	**Modular Innovation** ?
Changed	**Architecture Innovation** ?	**Radical Innovation** ?

Figure 2-7 Four Types of Product Innovation [Henderson and Clark (1990) p.12]

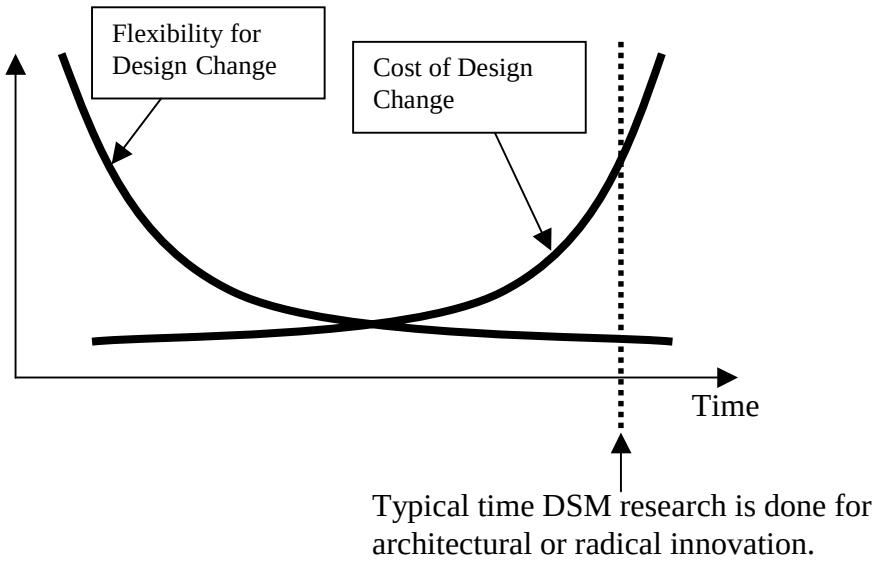


Figure 2-8 Typical Time A DSM Research is done

The second weakness of the DSM method is that it only improves the as-is process. Because the system interactions DSMs collected are based on expert experience, the DSM analysis only analyzes the system interactions that the experts think should happen. Most of the time, what the experts consider as system interactions is correct. However, the experts' subjective judgment based on how they always approached the design of a product may not be the best way to design the product (see the example later in section 4.3.3.2). In addition, what experts recall about what happened already may not always be accurate. Therefore, we need a method to objectively capture the interactions in the product system, and use those interactions to direct how people should communicate. If the method to capture the system interactions from a product perspective can be developed, this method will benefit not only products with architectural or radical innovations, but also mature product that experience incremental or modular innovations by learning how to come out of the box of how things are always done.

In short, DSM has been proven by many researches to be a very powerful tool for analyzing system interactions. Many additional analysis tools have been developed based on DSM to aid product design and project planning. However, the way DSM's are constructed limits the applicability of DSM techniques on the development of new products, where system analysis is most needed at early phase of the design process. Therefore, there is a need to construct DSM's at early phase of the design process by means other than using experts' experiences.

What engineers and managers think about at early phase of the design process are the design requirements and how design requirements can be met by the design concept. Both QFD (Quality Function Deployment) and Axiomatic Design's Design Matrix can be applied at early phase of the design process because they make use of the design requirements. The next section discusses QFD. The section following discusses the Design Matrix in Axiomatic Design.

2.1.2.7 QFD

Figure 2-9 shows the typical House of Quality used in QFD. A full scale QFD has four houses (Figure 2-10). Customer needs are traced through total system design requirements. System design requirements are translated into subsystem requirements then piece-part specifications. The piece-part specification is then translated into production process requirements, and finally production operation requirements [Clausing (1994)]. The top of the roof shows the relationship among the elements in the top row of each house.

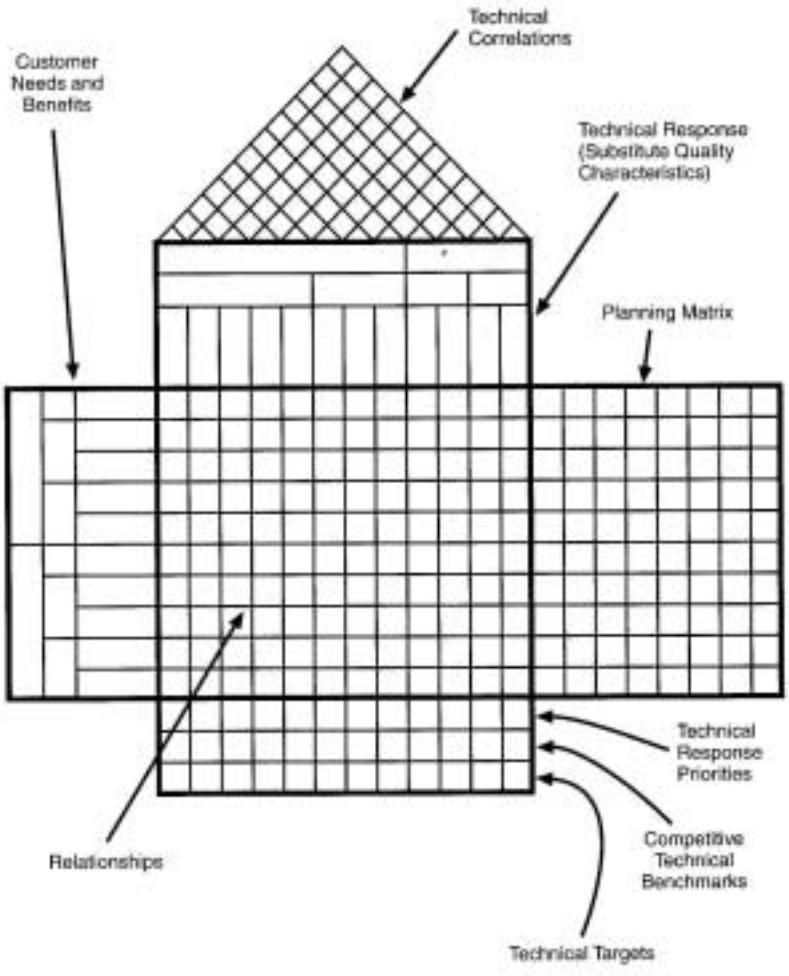


Figure 2-9: The QFD House of Quality [Cohen (1995) p.70]

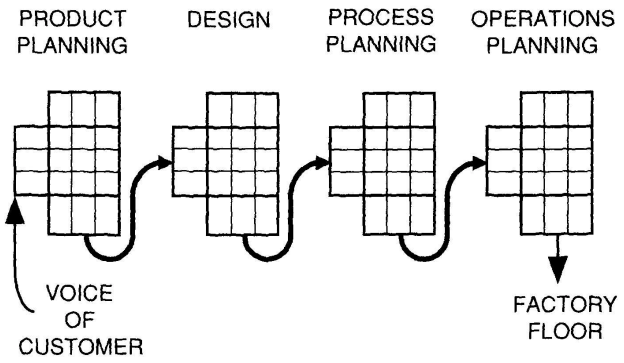


Figure 2-10: The Four Houses of QFD [Clausing (1994) p.68]

The advantage of QFD is that the matrices can be constructed at early phase of the design process because QFD matrices start with customer requirements. However, QFD is not quite the answer we are looking for to improve the weakness of DSM. First, the interactions captured by the roof of the QFD are still put in subjectively by experts. Therefore, the same problem as DSM exists for QFD. We cannot capture the interactions very well for radical and architectural innovations (Figure 2-7). Second, the top rows of each QFD matrix are the specifications rather than implementations. Therefore, we can say the QFD matrices stay in the requirements domain rather than cross over to the physical domain. The roof of each QFD matrix gives only the interactions among the requirements. We cannot infer the interactions among the components in the physical system from the roofs. Third, unlike DSM, QFD does not provide any analytical tools to deal with system couplings.

2.1.2.8 Axiomatic Design's Design Matrix

Axiomatic Design as a design theory involves many different aspects in its study [Suh (1990 and 2000)]. For the purpose of the discussion here, only the design matrix in the Axiomatic Design is briefly introduced here. The Axiomatic Design's Design Matrix (DM) relates requirements to design concept, and judges whether the system of the product is coupled using design matrix. A DM is a matrix that relates Functional Requirements (FR) to Design Parameters (DP). Functional Requirements are defined as "the minimum set of independent

requirements that characterize the functional needs of the product [Suh (2000)]." Design Parameters are "the key physical variables (or other equivalent terms in the case of software design, etc.) that characterize the design that satisfies the specified FRs [Suh (2000)]."

Figure 2-11 shows three Design Matrices (DM). The row headings are the Functional Requirements and the column headings are the Design Parameters. The marks in each matrix indicate the DP in the column heading contributes to the fulfillment of the FR in the row heading. When all of the marks in a DM are on the diagonal, the design is called uncoupled because adjusting any DP would affect only one FR. Therefore, the FR's in the design are independent of each other. When all of the marks in the DM are below diagonal, the design is called decoupled. In a decoupled design, although the change of a DP may affect more than one FR, we can adjust the DP's in the order of left to right. Then the FR's can be fulfilled in the order of top to bottom. Both uncoupled design and decoupled design maintain the independence of the FR's, and hence meet the first axiom in Axiomatic Design, although an uncoupled design is more desirable than a decoupled design. When not all marks in the DM fall on or below diagonal, such as the third DM in Figure 2-11, the FR's are no longer independent to each other. For instance, in the coupled design example in Figure 2-11, adjusting DP1 will affect FR2, which can cause DP2 to be adjusted. When DP2 is adjusted, DP has to be readjusted in order to fulfill FR1 again. Therefore, the design change between DP1 and DP2 are iterative in order to fulfill both FR1 and FR2, and FR1 and FR2 are no longer independent of each other.

	DP1	DP2	DP3
FR1	x		
FR2		x	
FR3			x

Uncoupled Design

	DP1	DP2	DP3
FR1	x		
FR2	x	x	
FR3	x	x	x

Decoupled Design

	DP1	DP2	DP3
FR1	x	x	
FR2	x	x	x
FR3	x	x	x

Coupled Design

Figure 2-11 Axiomatic Design Matrices

Axiomatic Design has many advantages over other system engineering methods. Two of the advantages that are relevant to this thesis are discussed here. First of all, Axiomatic Design Matrix prompts us to think about the system couplings in the physical form at early phase of the design process. This capability is exactly what DSM and other traditional system engineering analysis methods lack. The first Axiom prompts the design engineers to choose simple uncoupled design concept over complex interacting system designs in order to reduce problems associated with couplings in a system such as design rework, maintenance difficulty, etc. Therefore, Axiomatic Design takes the objective product-view in dealing with system interactions, unlike DSM, which relies on experts' subjective experience to capture system interactions.

Second, Axiomatic Design matrix enables us to think across two domains—the function and the form. Most of the existing system engineering methods, including QFD, only work in either domain alone. For instance, the DSM can be constructed to show the interactions among the requirements [Grady (1990) p.227] or that among the components in the hardware system. All requirements trace-ability tools including QFD deal with only the interactions among requirements. The Function Analysis System Technique (FAST) deals with only the functions, not the physical embodiment. However, based on psychology study of how people think when they design [Guindon (1990)], no one can think in only requirements domain or only physical domain. The thought process goes back and forth between the two domains. Therefore, the zigzagging process proposed by Axiomatic Design matches the natural thinking process of designers, and hence is very easy to apply during early phase of the design process.

However, Axiomatic Design also has many weak points. Four of them are relevant to this thesis research and will be discussed here. First, in order to avoid all the trouble that comes with system coupling, Axiomatic Design ask the designers to choose a design concept that makes the system uncoupled or decoupled (Figure 2-11). Although DM strives for ideal engineering design, the reality is that the ideal engineering design may not be feasible. From system architecture literatures, system design and decomposition must consider many aspects of the product development. Products are partitioned based on functional modules [Otto and

Wood (2000), Ulrich and Eppinger (2000)], geometric integration and precision [Ulrich and Eppinger (2000)], supply chain design [Ulrich and Eppinger (2000), Fine (1998), Parker (2000), Whitney (1993)], product family platform [Otto and Wood (2000), Gonzalez-Zugasti, et al. (1998 and 1999), Simpson (1998)], ease of assembly [Boothroyd and Dewhurst (1994)], cost [Gonzalez-Zugasti (1999)], etc. In addition, several searchers such as Rechtin (1991) and Ulrich and Eppinger (2000) mentioned partitioning the product for the ease of integration in their heuristic list. Therefore, products are never partitioned for the optimal of any single dimension, including system integration. The idea of designing the architecture so that we can have a completely decoupled or uncoupled design and will not have integration problems is naïve.

The second limitation of the Axiomatic Design DM is that it does not provide system analysis techniques to deal with the couplings in a design if the couplings cannot be avoided. Axiomatic Design insists that any good design should conform to the Axioms, and hence is either decoupled or uncoupled. Therefore, there would be no need for system analysis techniques such as the partitioning algorithm used for DSM. However, in reality, not only the ideal design in Axiomatic Design sense is hard to achieve, but it also takes time for a company to change the design of their current products. For instance, the already invested manufacturing facilities cannot be changed overnight for the new products. Axiomatic Design does not provide any tool to help with the transition period of a product design.

The third limitation of the DM is that it does not provide a clear way to deal with all types of design requirements. Axiomatic Design classifies requirements into two groups—the functional requirements and design constraints. Functional requirements can be decomposed through the construction of a Design Matrix using the zigzagging technique. However, constraints cannot be clearly decomposed like FR. Axiomatic Design advises the readers to keep in mind the constraints during decomposition. Tate suggests guidelines for constraint-decomposition (1999). He first classifies constraints into:

- Critical Performance Specifications
- Interface Constraints
- Global Constraints

- Project Constraints
- Feature Constraints

Then the general guidelines for decomposing constraints are:

- All Critical Performance Specifications will become lower level FR's.
- Interface Constraints will be converted into sub FR's.
- Global Object Constraints will not be refined into sub-FR's. They will remain constraints at lower level.
- Project Constraints will not be refined into sub-FR's.
- Project Constraints can be conditional.

All these guidelines seem good except that the classification of constraints is very foreign to design engineers. There exists a large body of requirements classification and management literatures already. If we could link the existing requirements classification to the FR and Constraints concepts, Axiomatic Design would be easier to implement in the practical world.

The fourth limitation is the most important weakness of Axiomatic Design. Axiomatic Design believes the effects of all requirements (FR and DP) on system interactions can be predicted in the DM, and hence the first Axiom states that an uncoupled or decoupled design matrix gives an uncoupled or decoupled design. This is rather a reductionism's view. Even from Tate's thesis (1999), we know not all requirements can be decomposed. Therefore, Axiomatic Design is not prepared to address the system interactions due to system emergent properties.

The first two weaknesses of Axiomatic Design—cannot address couplings and does not provide system analysis—are the strengths of Design Structure Matrix method. The third weakness of Axiomatic Design requires us to look further into requirements categorization in requirements management literatures. This way, we may get a better understanding on what the constraints in Axiomatic Design really are. In dealing with the last weakness, DSM is a suitable alternative for recording the system interactions generated by emergent system properties.

2.1.2.9 Requirements Classification

Looking through the requirements management literatures [Blanchard and Fabrycky (1981), Buede (2000), Hooks and Farry (2001), Gershenson et al. (1994), Grady (1993)], requirements categorizations fall into the following three categories:

1. Categorization based on the source of requirements
2. Categorization based on the subject of the requirements
3. Categorization based on the flexibility and tradability of the requirements

2.1.2.9.1 Categorization Based on the Source of Requirements

Gershenson et al. (1994), Buede (2000 p.122), and Martin (1997) categorize requirements based on the sources of the requirements. The purpose of this way of categorization is to make sure all stakeholders of the product design have inputs in the requirements for the product. The four major stakeholders are:

- The end user and the user context

  Typically, the requirements from the end users are collected in marketing analysis documents/customer needs analysis.
- The corporate stakeholders

  This category includes all departments downstream of the design phase, such as manufacturing, finance, service, distribution, etc. These departments set the internal design requirements for the product.
- The regulatory agency

  The government regulatory requirements are included. This also includes standards in for the design.
- The technical design group

  This is the technical group that is involved in the design. They set most of the technical requirements for the product.

2.1.2.9.2 Based on the Subject of the Requirements

Hooks and Farry (2001) and in their recent book on requirements have proposed a list of requirements. This categorization is based on the different aspects in a product's lifecycle that a design team has to consider. The list is as follows:

- Functional Requirements

  This type of requirements describes what function the product performs. For example: "The systems shall be able to track and manage the inventory of all on-board consumables."
- Performance Requirements

  This type of requirements describes how well a function has to be performed. For example: "The systems shall be capable of performing…within both of the following intervals: (1) 30 minutes from a cold startup at an ambient temperature of 20^{o}C; (2) 45 minutes from a cold startup at an ambient lab temperature of 15^{o}C." Some of the performance requirements go along with the functional requirements, and hence can be combined with functional requirements. Some of the performance requirements are system level behaviors, and can hardly be decomposed, such as this one.
- Classical Reliability

  This type of requirements describes how long should a product last before any failure. It can be measured by service call per machine per year, availability of the machine, mean-time-between-failure, etc.
- Analytical Reliability

  This type of reliability checks on the probability of malfunctioning of the machine. FMEA is used. Where analytical reliability is not guaranteed, monitoring devices and warning signs are placed.
- Maintainability/serviceability

  This type of requirements checks on the ease of performing maintenances and service. For example, "routine maintenance should be automated."
- Operational Environment

  This type of requirement states the environment the product has to work in, such as the temperature, pressure, etc.

- Operability

  This type of requirements states the ease of operation of the product. For instance, "There should be minimal operator involvement in programming tests." Sometimes, the operability requirements can be translated to functional requirements of the product. Sometime, it is a trial error process of understanding what the user needs.

- Safety

  This type of the requirements concerns this safety of the machine. For instance, fluid bottle caps and lines carrying fluids shall be designed to eliminate user contact with those liquids and to be simple to connect and replace." From this example, we can see sometimes safety requirements can also be translated into functional requirements of the product.

- Appearance

  Requirements on the appearance of the product.

- Packaging

  Requirements on the assembly, spatial arrangement of the product.

- Weight and Size

  Requirements on the weight and size of the product. For instance, "The analyzer should be no larger in size and weight than a XXX."

- Installation

  This contains the ease of installation and the reliability of the product at installation.

- Upgrading, expandability/configureability

  This requirements considers the future and the use environment of the product. For instance, "Future analyzer derivatives should be able to accommodate hybrid technologies and highly sensitive assays." Expandability and configureability requirements are similar to functional requirements. They can be seen as the functions the product must have for the future use.

- Transportation (including storage, loading, logistics)

  The requirements about the non-operational environment that the product has to withstand. For instance, "The packaged system and components shall withstand the following non-operational environment without degrading performance: Cold XXX, Heat XXX, Humidity XXX, Altitude XXX…"

- Manufacturing and assembly

  Requirements on how to make the product easy and cheap to manufacture and assembly. For example: “Slotted screw heads shall not be used.”
- Training

  Requirements on the training of users. For instance: “The analyzer should be easy to use so that training can be on- or off-site and should not take more than 4 days.”
- Retirement, disposal

  requirements on the retirement and disposal of the product. For instance, “ Packaging materials should be recyclable.”
- Cost

  Requirements on how much the product will cost the consumers.
- Timing, funding

  Requirements on the completion date and the budget of the project.
- Patents

  What patent to be aware of cannot be explicitly stated on the requirements. This type of requirements usually just state being aware of the patents in general.
- Policy and Procedure, Regulatory requirements
- Reuse of components
- Design Constraint

  Introduce fixed design implementation to narrow the choice of design concepts during synthesis. This design constraint is defined differently from that in Axiomatic Design. The constraints in Axiomatic Design are design inputs that cannot be decomposed as Functional Requirements using the Design Matrix.

Table 2-1 shows an attempt to relate the type of requirements and the contributing sources. The significance of constructing such a table is to help the design team to examine if all stakeholders have provided inputs, and if all subjects of the design for the product life cycle have been considered in the requirements generation phase.

ID	Requirement Type	End user	Corporate	Technical Team	Regulatory
1	Functional	X		X	
2	Performance	X		X	
3	Reliability (Classical and Analytical	X	X	X	
4	Maintainability	X	X	X	
5	Operational Environment	X			X
6	Operability	X		X	
7	Safety	X	X		X
8	Appearance	X	X		
9	Weight/Size	X		X	
10	Packaging		X		
11	Installation	X	X		
12	Upgrade/configurability	X	X	X	
13	Transportation	X	X		X
14	Manufacturing		X	X	
15	Training	X	X		
16	Retirement		X		X
17	Distribution		X		
18	Cost	X			
19	Timing and funding		X		
20	Patent		X	X	
21	Policy and Procedure		X		X
22	Reuse of Components		X	X	
23	Design Constraint	X	X	X	X

Table 2-1 Relating Product Design Stakeholders to Requirements Categories

2.1.2.9.3 Based on the Flexibility and Tradability

In many System Engineering documents, one may find requirements classification based on the flexibility of the requirements [Grady (1993), Stevens, et al. (1998), Sage and Rouse (1999)]. The categories are:

- Constraints—boundary conditions within which a designer must remain while satisfying the aggregate of the performance requirements for the item [Grady (1993) p. 355]. Constraints rule out certain possible design choices. For instance, a product

must function in the temperature between 0 and 32 Fahrenheit degree is a constraint on the design. In Pahl and Beitz (1995 p. 45), constraints include safety, ergonomics, production, quality, assembly, transport, operation, maintenance, recycling, and expenditure.

- Requirements—the range of acceptable measures for a successful design [Buede (2000) p.121]. For instance, if the customer wants a fast vehicle. The requirement on the speed of the vehicle can be set based on interpretation of how what fast means to a particular customer or customer group. The definition of requirement here is narrower than that used in this section (2.1.2.9).

- Goals—the desirable but not essential features of a product. Not achieving a goal does not mean the failure of the product design.

The flexibility increases from *Constraints* to *Goals*. Note the definition of constraints here is different from that used in Axiomatic Design. The constraints in Axiomatic Design are the requirements on the product that cannot be categorized as Functional Requirements [Suh (2000)]. Also, the definition of constraints used in this thesis (see section 2.1.2.9.2) is narrower than the constraint defined in this section.

Relating them to the stakeholders of product design to the flexibilities of the requirements, we can construct the table below (Table 2-2).

	End user	Corporation	Technical Team	Regulatory
Constraints	X	X		X
Requirements	X	X	X	X
Goals	X	X	X	

Table 2-2 Design Stakeholders vs. Flexibility of the Requirements

This table shows us the flexibility of each stakeholder in the design of the product. The technical team is the only one that has more flexibility. Technical teams must be aware of the constraints imposed by end user, corporation, and regulatory.

2.1.2.9.4 Based on the Design Approaches

The above three means of requirements categorization all focus on collecting a complete list of requirements, and identifying the priorities among the requirements. In fact, all of the requirements management techniques including requirement-traceability management remain in the domain of requirements. However, the reality is that one cannot work in the requirements domain only [Guindon (1990)]. Requirements and the selection of the physical form of the product must go hand-in-hand. For instance, if the high level requirement is "Fasten two plates." The high level form we can choose including gluing, welding, using clamps, using bolts and nuts, etc. Choosing gluing or clamping will result in very different requirements in the next level of decomposition.

Axiomatic Design's Design Matrix helps to relate the requirements to the physical design. Yet, Axiomatic Design does not have a good framework on how to deal with various types of requirements. Therefore, combining the two methods (Axiomatic Design and Requirements Management) may give us the advantages of both methods.

2.1.2.10 Summary on the Methods Dealing with System Interactions

Table 2-3 summarizes the system engineering methods reviewed in this section. DSM, DM, and requirements classification are complementary to one another. This thesis research intends to combine the strength of the above three methods in order to better deal with system interactions in the development of large complex systems.

	Heuristics	Graphs and Diagraphs	Trees	IDEF	CPM, PERT, Gantt chart	DSM	QFD	DM	Requirements Classification	This Thesis
Help to deal with system interactions	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**		**X**
Visualize system interactions		**X**	**X**	**X**	**X**	**X**	**X**			**X**
Show the Hierarchical Order of System Elements			**X**					**X**		**X**
Can be used to both understand the interactions in the physical system and manage tasks in a project		**X**	**X**			**X**				**X**
Sequence elements in the same hierarchical level				**X**	**X**	**X**				**X**
Provide analytical system analysis to predict project completion time					**X**	**X**				**X**
Allow feedback loops						**X**	**X**			**X**
Analytically identify the inevitable iterations and eliminate unnecessary iterations						**X**				**X**
Cluster the elements in the system for architectural design						**X**				**X**
Identify the rate of convergence in the feedback loops, and the controlling factors in the iteration						**X**				**X**
Can capture emergent system properties						**X**				**X**
Manage system interaction knowledge in complex system designs						**X**				**X**
Relate the requirements and the physical system design of the product								**X**		**X**
Can be constructed early in the design process							**X**	**X**	**X**	**X**
Can be applied to new products							**X**	**X**		**X**
Fits the design engineers' natural thinking process for concept generation								**X**		**X**
Prescribe system interactions rather than describe how things are always done								**X**		**X**
Prompt the engineers to compare the system complexity of different design concepts								**X**		**X**
Provide a complete understanding of all requirements									**X**	**X**

Table 2-3 Summary of the System Engineering Methods Concerning System Interactions

2.2 Design Theory

There exist many literatures on design theory [Finger and Dixon (1989), Tate and Norland (1995)]. The literature review here will sample a few of the representative schools and compare the differences between them. The discussions below focuses on the research question of how system interactions are treated by different design theories, and their pros and cons.

2.2.1 European Design Models

The two most representative European schools for design process are Hubka (1980), and Pahl and Beitz (1995).

Hubka proposes a phased design process (1980 p. 35). The design process starts with problem statement and the development of design specifications. Then function structures are built according to the specifications. Afterwards, the design concept iterates through various dimensions of the function structures until all functions are fulfilled. Then the design progresses into the preliminary layout stage, dimensional layout, and finally detailed assembly drawings. At this point, the design process is complete. Although design iterations are very much stressed by Hubka, the iteration in his design process occurs within each of the design phases. Iterations between different phases of the design process is not explicitly represented by Hubka's design model.

Pahl and Beitz (1995) is the most representative design literature for European design schools. They also propose a phased design process including the following three stages (p. 67):

- Conceptual design phase decomposes the functions. When the functions are decomposed, consider the design concept that could meet the functions.

- Embodiment phase considers the spatial relationship and the "–ilities" such as reliability, maintainability, etc. The system emergent properties are taken into consideration in this phase. The non-functional requirements are addressed here.
- Detailed design phase optimize the details of the design. Pahl and Beitz recognize that engineers may still come up with better design ideas in detailed design phase and the design can still change and evolve, as the understanding of the design and the system becomes more mature. This recognition of late design change reflects the acceptance of emergent property of the system in Pahl and Beitz design theory.

The design processes proposed by both above literatures are very similar. Both Hubka, and Pahl and Beitz suggest that function comes before form. Their design models regarding the relationship between function and form can be summarized as Figure 2-12. The interactions among functions are considered only in the function domain. The interactions among the forms are considered only in the Form domain. Neither schools of literature explicitly stated the relationship between the two types of system interactions.

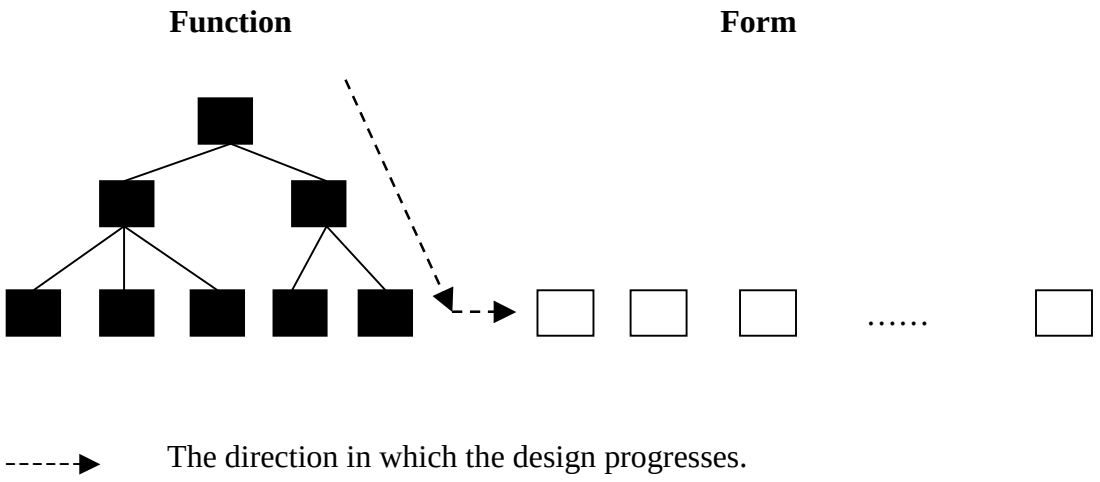


Figure 2-12: Sequential Model between Function and Form

In addition, both literatures stopped the design process at the generation of the CAD drawings. The collaboration among various departments in an organization such as the relationship between manufacturing activities and design are not stressed in these literatures.

2.2.2 Total Quality Deployment

Total Quality Deployment (TQM) is a design process promoting the concurrent collaboration among traditionally segregated departments in an organization during the design. QFD (Figure 2-9 and Figure 2-10) is the core method used in TQM to reflect the interrelationship among various parts of the organization, so as to achieve a holistic view of product design.

However, the relationship among the function and form in TQM is still the same as that in Figure 2-12. QFD's decompose the requirements from system level to subsystem level until piece-part level without representing the physical form of the design. Then the piece-part level specifications are translated into the manufacturing specifications and operations specifications. In addition, there is no explicit mention on how to deal with emergent properties of the system.

2.2.3 Axiomatic Design

Axiomatic Design proposes four domains in the design (Figure 2-13). Design Matrices are used to relate the domain to its adjacent domain. For instance, the design matrices in Figure 2-11 relate the functional domain to the physical domain.

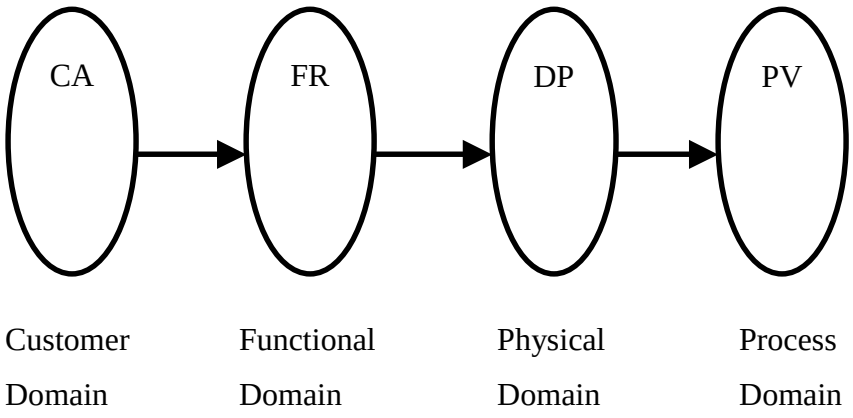
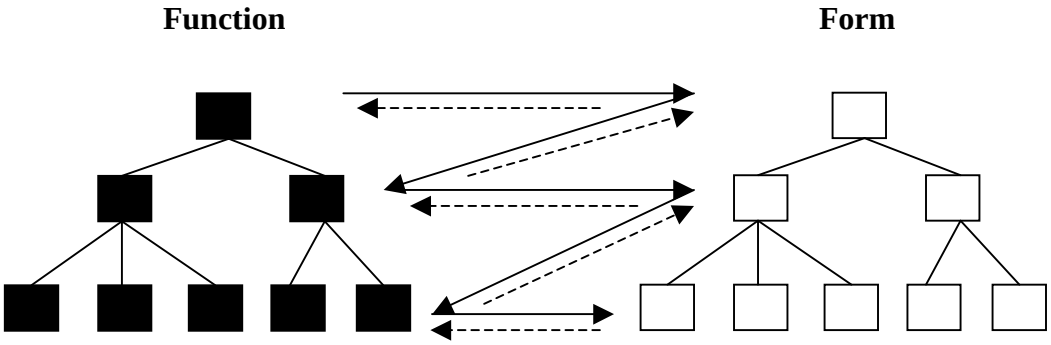


Figure 2-13: Axiomatic Design's Four Domains

The four domains of Axiomatic Design are very similar to the four houses in QFD (Figure 2-10). The main differences between the Design Matrix (DM) and the House of Quality (HOQ) matrices are the following:

- DM relates the requirements with the physical form. HOQ stay in the requirements domain.
- DM generates the requirements hierarchy through the zigzagging method between the function and the form (Figure 2-14). HOQ generates the hierarchy in the requirements without referencing to the form (Figure 2-12).



Decompose the functional requirements and relate the decomposition of the functions to that of the form through out the system hierarchy (zigzagging process in Axiomatic Design [Suh (2000)]).

When the lower level function cannot be met by the form chosen, the design engineers may trace back to the decision at higher level of the system and redesign the form at higher level.

Figure 2-14 Axiomatic Design Decomposition

Psychology experiment has shown that people cannot think only in the function or form domain [Guindon (1990)]. Therefore, Axiomatic Design provides a design process that is more natural to the thought process of the engineers than the previous design theories.

- However, Axiomatic Design does not consider the issue of system emergent properties. Axiomatic Design believes by decomposing the functional requirements

using the zigzagging process, we can predict the interactions among the DP's, and hence the couplings among the FR's. The first Axiom judges whether a system is uncoupled using the engineers' understanding on the relationship between the FR's and DP's at the beginning of the design process. What Axiomatic Design neglected is the system emergent properties, which may introduce more couplings as more details of the design become mature. Although neither the European design theories nor the Axiomatic Design provide any means to deal with the additional system interactions introduced by emergent properties, the failure of addressing the system emergent property in the design axioms reduced the credibility of Axiomatic Design theory.

2.2.4 Summary of Design Theories Reviewed

Table 2-4 compares the design theories reviewed above. Note the comparisons made in this section only focused on the metrics that are important to this thesis research. Many more strengths and limitations of the design theories are not listed in Table 2-4. This thesis will take Axiomatic Design as the starting point because it has the most advantages, and discover ways for the Axiomatic Design to deal with emergent properties in the design, so as to improve the capability of Axiomatic Design.

	Hubka, Pahl and Beitz	Total Quality Deployment	Axiomatic Design	This thesis
Provide a framework for design process	**X**	**X**	**X**	**X**
Address the emergent property of the system	**X**			**X**
Stress the collaboration among various departments in the organization		**X**	**X**	**X**
Indicate the relationship between the functions and the interactions among the components in the physical form			**X**	**X**

Table 2-4: Summary of the Comparison among Design Theories

2.3 Knowledge Management

2.3.1 System Level Knowledge Management in Literatures

In chapter 1 (section 1.2.3.2.1), we have defined the system level knowledge as the knowledge about the entire large complex system. It is the knowledge that must be learned through collaboration with other people, and hence is the knowledge that belongs to an organization rather than any one individual. Many knowledge management literatures exist. The relationship among them and where managing system level knowledge fits in are shown in Figure 2-15.

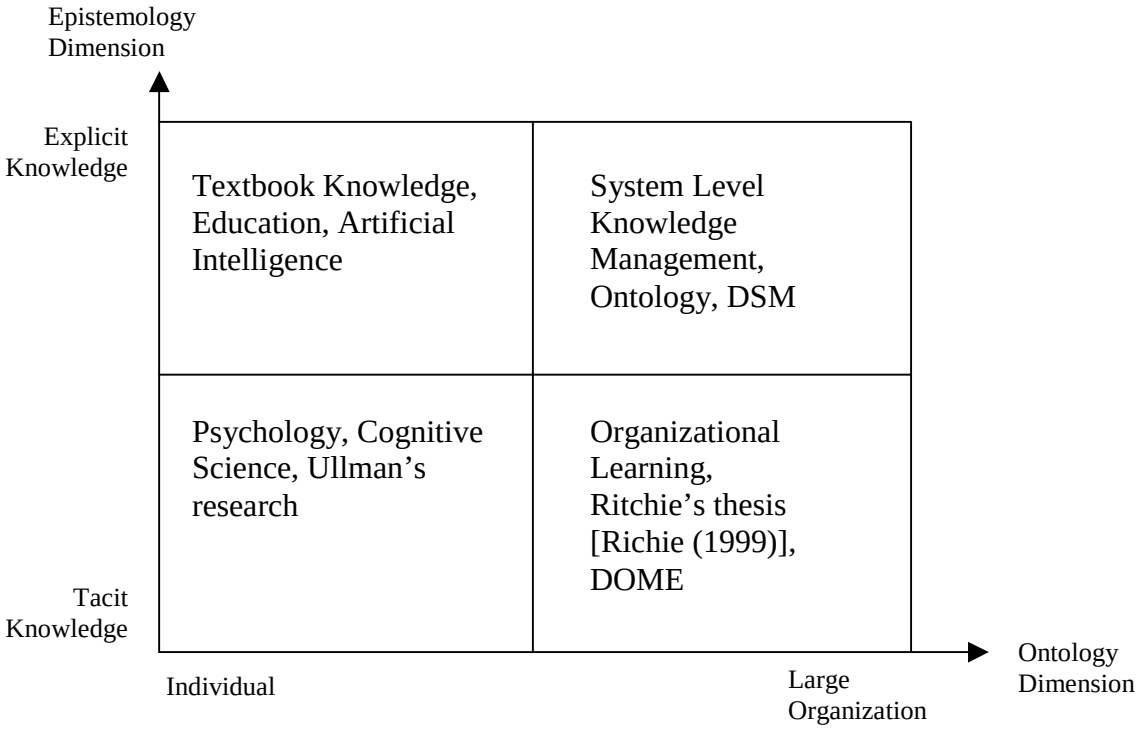


Figure 2-15: Classification of Knowledge Management Literatures

First, the definitions of epistemology and ontology are provided here. Epistemology is the branch of philosophy that studies knowledge. It attempts to answer the basic question: what

distinguishes true (adequate) knowledge from false (inadequate) knowledge? Practically, this question translates into issues of scientific methodology: how can one develop theories or models that are better than competing theories? It also forms one of the pillars of the new sciences of cognition, which developed from the information processing approach to psychology, and from artificial intelligence, as an attempt to develop computer programs that mimic a human's capacity to use knowledge in an intelligent way.

Ontology specifies the most fundamental categories of existence, the elementary substances or structures out of which the world is made. Ontology will thus analyze the most general and abstract concepts or distinctions that underlie every more specific description of any phenomenon in the world, e.g. time, space, matter, process, cause and effect, system. Recently, the term of "(formal) ontology" has been up taken by researchers in Artificial Intelligence to use it to designate the building blocks out of which models of the world are made. An agent (e.g. an autonomous robot) using a particular model will only be able to perceive that part of the world that his ontology is able to represent. In that way, ontology becomes the basic level of a knowledge representation scheme [Uschold and Gruninger (1996)].

In Figure 2-15, the epistemology dimension represents the understanding of the knowledge. Tacit knowledge is knowledge not being formalized. In engineering design situation, they are usually gathered through experience, and exists in forms of past design stories. Explicit knowledge is knowledge formalized. The knowledge we learn from school in the textbooks is the best example of explicit knowledge. In the engineering design situation, explicit knowledge may exist in the form of technical report, design rules and guidelines, etc. The ontology dimension in Figure 2-15 does not have the same meaning as the artificial intelligence community's definition. Rather, this dimension represents whether the learning and storage of the knowledge belong to individuals or organizations.

From Figure 2-15, we can see where past knowledge management literatures fit in. When the study is about how individuals learn and process information (the lower left quadrant in Figure 2-15), most of the research can be found in cognitive science and psychology

literatures. For instance, Guindon (1990) discovered that individual design engineers think in a highly non-linear and iterative fashion. Gick and Holyoak (1980) wrote about how people use analogy to solve unfamiliar problems. Genter (1983) proposed a framework for knowledge transfer. Ullman, et al. (1983, 1995), and Kuffner and Ullman (1991) have done extensive research using psychology techniques to understand what knowledge engineers need to design a product.

The upper left quadrant of Figure 2-15 is about how individuals learn and use explicit knowledge. Education is the field that studies this topic. In addition, the very popular Knowledge-based Engineering design applications using Artificial Intelligence Experts System also belong to this area. Artificial intelligence has been successful in dealing with design problems concerning small components and assemblies [Tong and Sriram (1992)]. The design rules must also be explicit in order to be programmed into the expert system. However, AI has not focused its effort on dealing with large complex systems [Whitney (1999), Dong (1999)]. One of the causes may be that the system level knowledge about the design of large complex systems is not well understood yet.

Now moving to the organization knowledge side. Look at the lower right quadrant in Figure 2-15 first. Many of the organization learning researches [Nonaka and Takeuchi (1995), Ritchi (1999)] concern facilitating the exchange and capturing of tacit knowledge within organizations. Since the unit in an organization is the individual human beings, the research concerning the individual's knowledge and learning discussed above are also relevant to the organization learning research. DOME—the MIT design knowledge-sharing-and-simulation platform developed by Professor D. Wallace and his students—helps the organizations to overcome the widespread of tacit knowledge [Abrahamson (1999 and 2000), Senin, et al. (2000)]. Yet, being a successful system engineering technology enabler, DOME does not provide answers as what knowledge needs to be communicated. Instead, it provides a free marketplace for design information exchange. Hence, DOME is a very suitable tool to deal with the tacit knowledge about system interactions.

Last but not least is the upper right quadrant in Figure 2-15. This quadrant represent the definition of managing system level knowledge in this thesis—explicitly capture the understanding about system interfaces and facilitate the reuse of this knowledge. So far in literatures, only Design Structure Matrix (DSM) has been used to capture the explicit system level knowledge. The author's master degree thesis first showed this capability of DSM [Dong (2000)]. Bartowski (2001), and Glynn and Pelland (2000) at Pratt and Whitney applied DSM to capture the system interaction knowledge for a module-centered business. Thebeau (2001) uses the process of building DSM as the process of capturing system level knowledge. Therefore, we can conclude that DSM is a suitable method for managing system level knowledge.

In addition to representing the system level knowledge using DSM, research work in ontology is also relevant to the upper right quadrant in Figure 2-15. Ontology helps to unify the terms used in a domain so as to improve the communication among people or software agents [Uschold and Gruninger (1996), Noy and Hafner (1997)]. In order to set up a framework to enable knowledge sharing at system level, ontology is needed [Borse, et al. (1996)]. However, ontology is different from classification. Ontology concerns the terms used to communicate. Classification concerns the similarity/difference between two entities. This thesis is interested in understanding what types of system level knowledge there are. Therefore, knowledge classification is the focus rather than ontology.

At the end of this section, it is important to realize that the four quadrants in Figure 2-15 are not independent and separated. Individuals are the units of organizations. Explicit knowledge comes from tacit knowledge. The research in each quadrant benefits the understanding in other quadrants. The intention of Figure 2-15 is to show what work has been done in the literature and what the focuses were. Figure 2-15 suggests there has not been a lot of methods and tools developed for managing system level knowledge. Therefore, more work is needed in this area.

2.3.2 System Level Knowledge Classification

In order to understand what system level knowledge is, a classification for system level knowledge is needed. There exist many ways to classify knowledge. Various classifications have different basis and serve various purposes. For instance, library subject indexing serves as a way for easy knowledge browsing and retrieval. Epistemology classifies knowledge in order to measure the level of adequacy of a piece of knowledge (tacit vs. explicit). Prasanna (2000) has a comprehensive review of various knowledge classifications.

However, only a few pieces of work have been done on the classification of knowledge used by design engineers. Ullman, et al. (1983) studies the types of knowledge individual designers requests so that all knowledge types can be incorporated in the intelligent CAD design tool. Hutton and Klein (1999) classified the knowledge based on what makes an engineer the expert in a decision making process. Patil (2000) classified the design knowledge into *What*, *How*, *Who*, and *Why* based on how each type of knowledge is addressed by existing IT design tools, so that we can identify which area of knowledge the existing IT methods and tools are inadequate.

This thesis intends to classify system level knowledge for the ease of capturing, storage, and retrieval. The classification of system level knowledge is different from the classification of an individual human being's knowledge, because system level knowledge is organizational knowledge. System level knowledge is also more than the engineering knowledge about the dynamical physical system design. It involves also the human aspect in the organization such as who had a particular piece of information. Only Patil (2000)'s work had relevance to the goal of this thesis. Therefore, this thesis research will propose a system level knowledge management ontology built upon the basis of Patil's work.

2.4 Progress Made Regarding Research Questions

This chapter searches past research with the goal of answering two research questions in Chapter 1: Q1-a and Q2-a. The answers to these two research questions are summarized as follows.

Q1-a: What methods have been used in the past to capture system level interactions? What are the strengths and weaknesses of existing methods? Is DSM a good way to predict system level interactions?

A review of system engineering methods and design theories showed that many existing methods could capture system level interactions. Table 2-3 summarizes the strengths and limitations of various systems engineering methods. Among the methods surveyed, Design Structure Matrix (DSM) is most suitable for not only capturing system level interactions, but also providing analysis to the system interactions to aid the design and management of the system. However, DSM is hard to use at early phase of the design process or for new product development using the current interview method to construct a DSM.

Axiomatic Design's Design Matrix (DM) provides many of the capabilities DSM lacks, including being able to apply at early phase of the design, can be applied to new products, prescribing system interactions rather than capturing the experts subjective opinion, allowing engineers to compare the system complexity of various design concepts, fitting into the natural thinking process of design engineers, etc. In addition, Table 2-4 shows Axiomatic Design is one of the most advanced design theory except for its negligence of emergent properties of systems. Yet, DM's weaknesses such as not being able to deal with iterations in the system and not being able to reflect emergent properties of the system are strengths of DSM. DM is also lack to solid techniques to deal with various types of design inputs (requirements) other than functional requirements (FR). Requirements management literatures on the other hand provide great resources for understanding various types of requirements.

Therefore, the review of literature suggests that we should find ways to combine the strengths of DSM, DM, and requirements classification so that we can have a method to predict and analyze system interactions at early stage of the design process.

Q2-a: What has been done in managing system level knowledge?

Figure 2-15 shows the various types of existing literatures concerning knowledge management. Very little can be found about managing system level knowledge except for some of the recent DSM studies. DSM has been shown to be a good method to capture system level knowledge. Very little work also has been done regarding classification of system level design knowledge. This thesis will start from Patil's knowledge classification to propose a classification for system level knowledge.

2.5 Summary

This chapter reviewed existing literatures in the field of system engineering techniques, Design theory, and knowledge management. The current practices, their strengths and weakness are compared in Table 2-3, Table 2-4, and Figure 2-15. The following conclusions were made:

1. DSM and DM complement each other's capability in dealing with the interactions in complex systems. Combing the two methods will enable us to predict and analyze system interaction at early phase of the design process, so as to avoid costly rework and delay later.
2. Existing requirements classification should be used so that we can take into consideration all of the important design inputs for a product.
3. Axiomatic Design is one of the most advanced design theory among all of the existing ones. However, Axiomatic Design neglected the system emergent properties. DSM enhances Axiomatic Design by providing the capability to capture system emergent properties.
4. Little work has been done in managing system level knowledge. DSM has been shown to be a good way to capture system level knowledge.
5. Managing system level knowledge requires the ontology of system level knowledge. Very little previous work has been done in the area of classifying system level knowledge.

3 Research Method

The main research goals of this thesis established in Chapter 1 are the follows:

- Obtaining and managing system interactions at early stage of the design process
- Managing system level knowledge

Literature search in Chapter 2 reveals that little work has been done in managing system level knowledge. Therefore, Section 3.1 in this Chapter introduces a framework proposed by this thesis research to manage system level knowledge.

Literature search also reveals that the Design Structure Matrix method can help to manage system interactions but is difficult to apply at early phase of the design process, while the Axiomatic Design's Design Matrix can predict system interactions early on. Section 3.2 in this chapter proposes a matrix transformation method to transfer a DM into a DSM so that we can take advantage of both methods to achieve the objective of both obtaining and managing system interactions at early stage of the design process.

A large portion of this thesis research is to test the knowledge management framework and the matrix transformation method using case studies conducted in real engineering companies. Section 3.3 introduces the two case studies used in this thesis research to test the framework and method explained in Section 3.1 and 3.2.

In the end, Section 3.4 and 3.5 concludes the chapter by summarizing the progress made on answering the research questions listed in Chapter 1.

3.1 A Framework for Managing System Level Knowledge

3.1.1 Requirements on the System Level Knowledge Management Framework

As discussed in Chapter 1, system level knowledge in this thesis includes four major parts:

1. What the system components are
2. How system components interface with each other to achieve the functions
3. Who has the knowledge about each system component
4. Where to find the documented knowledge about each system component

The system level knowledge management framework must incorporate all four above aspects. In addition, since the system level knowledge also has the organization management aspect, we cannot ignore the project management types of knowledge such as scheduling, budgeting, etc.

In addition, the framework must allow the engineers to capture knowledge throughout the design process. The reason is that our knowledge about the system grows as the development process is carried out. The emergent properties of a system are usually learned during system integration, verification and validation, or even field service. The framework must service as a basis to record the continuous learning experience, and keep the system-level knowledge in the organization. Therefore, the system level knowledge management framework should reflect various stages in the design process.

Furthermore, a system level knowledge management framework must provide incentives for documentation. Knowledge management is good for an organization in the long term, but usually adds work to the already very busy engineers and managers. The most common complaint I get from talking to the engineers about knowledge management is that they are so busy dealing with design problems that they do not have the time to document. In order to encourage the engineers to spend time to document the latest learning about the system emergent properties, the knowledge management tools and methods must provide not only long-term benefits to the organization, but also short-term benefits to the engineers and managers who are involved with the system design and development. Or else the method must be designed to capture the knowledge in background of the design activities automatically. The research in this thesis takes the former approach.

3.1.2 Proposed System Level Knowledge Management Framework

In light of the above requirements, a system level knowledge management framework is proposed (Figure 3-1). This framework is based on the Axiomatic Design theory (Figure 2-13) with additional modification. Each of the polygons represents a domain of knowledge in the design. The first domain--Customer Needs and Enterprise Strategy—is only Customer Needs in Axiomatic Design. The Enterprise Strategy is added here to include the effects of internal stakeholders in a company. The second domain—Functional Requirements and Constraints--represents the design inputs. The third domain—Design Parameters—represents the design itself. The fourth domain—process variables—represents the manufacturing process design. The second, third, and fourth domains maintain the same meaning as they were in Axiomatic Design. In addition, the fifth domain—Product—is added here in order to address the emergent properties of the system that are learned through integration, verification, and field service.

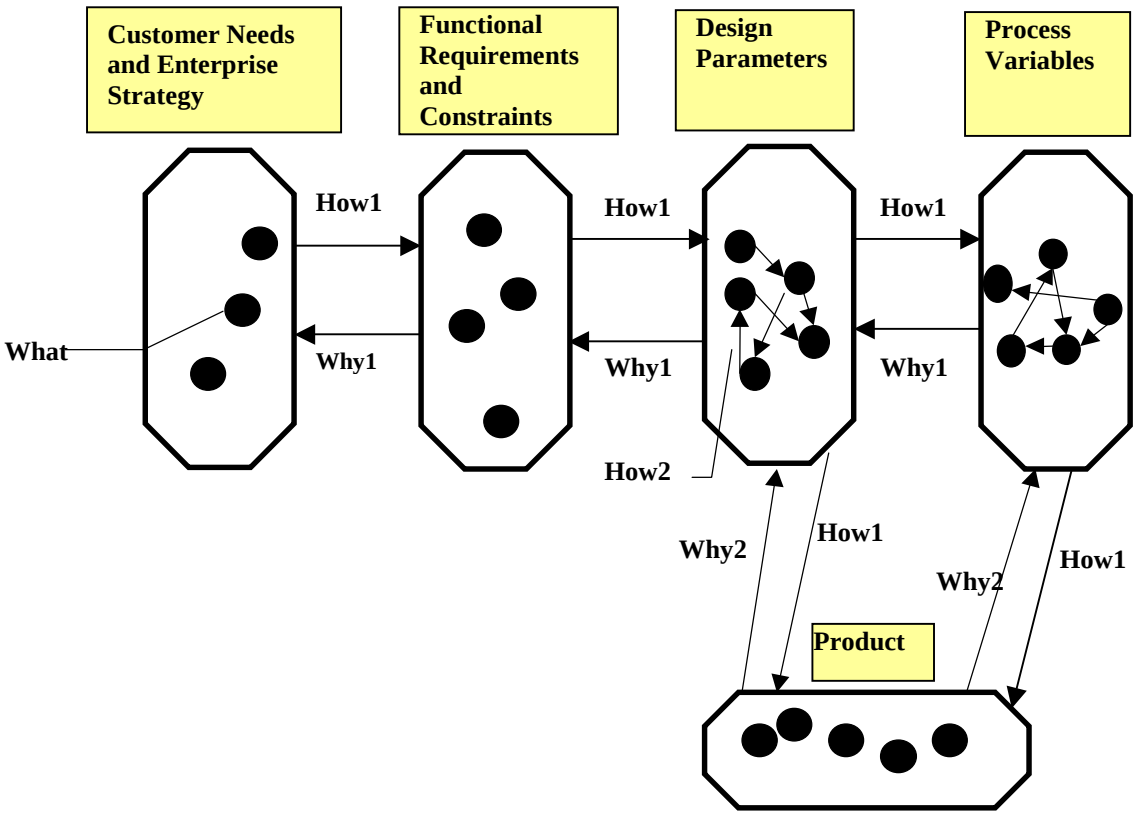


Figure 3-1: A Framework for Managing System Level Knowledge

The types of knowledge involved in each domain and between domains are listed below. Note that not all knowledge types below are shown in the graph in Figure 3-1. The knowledge classification below is an attempt to set up system knowledge management ontology.

What: the facts in design. It can be what the customer requirements and enterprise strategies are, what the requirements and constraints are, what the physical design solutions are, what the manufacturing processes are, and what the product is. In Figure 3-1, the "What's" are the dark dots in each domain.

How1: The first type of "how"—"How1"—concerns the solution to requirements, or the requirement flow-down. This type of "How" is indicated in Figure 3-1 by the arrows going from left to right or top to bottom. For example, how a functional requirement is fulfilled by the DP solutions belongs to this type of how.

How2: The second type of how deals with the interface among the elements (What's) in each domain. For example, the interactions among the design parameters belong to this category of how. How2 type of knowledge exists in each domain in Figure 3-1. This type of knowledge may also concern the sensitivity of certain elements to the output and the dynamic relation of the elements to the total output.

When: When type of knowledge concerns the sequence of completing design tasks. This knowledge can be inferred from the knowledge regarding system interactions (How2) using the partitioning technique in the Design Structure Matrix method.

Where: The location of information and hardware.

Who: This is a subset of Where. It refers to the person who owns particular pieces of information.

Which: Contains the logic for choosing among alternatives. This concept is applicable to both DP's and PV's.

Why1: This type of Why corresponds to How1. In Figure 3-1, the arrows for Why1 goes in parallel but opposite directions of the arrows for How1. Why1 type of knowledge records the trace-ability of design intent and design requirements.

Why2: This Why explains the emergent system behavior that occurs during system integration and perhaps field service. It is a different type of knowledge from the

Why1. Why1 traces the intended design rationales. Why2 records the unintended system behavior.

Rules: These can be categorized as conclusions from past experiences. They may be backed up by Why's, but they may just be heuristics. They may be viewed as common senses.

3.1.3 How the Requirements on System Level Knowledge Management Methods are fulfilled by This Framework

The above framework fulfills the requirements for a system level knowledge management framework (see Section 3.1.1). It addresses all four aspects of the system level knowledge listed in Chapter 1. First, the "What" type of knowledge contains the knowledge about system components. Second, the "How1", "How2", "Why1", and "Why2" types of knowledge address the interconnections among the system components. The "When" knowledge derived from "How" knowledge aids the project management effort. Fourth, the "Where" and "Who" knowledge addresses the issue of finding the source of the knowledge and easy browsing of the knowledge database. In addition, the "Which" and "Rules" types of knowledge further prompt the engineers to learn from past lessons and deduce knowledge from experiences for future use. The rules summarized from experiences can be used in the Knowledge-based Engineering applications, and transfer tacit knowledge to explicit knowledge. In reference to Figure 2-15, this framework is capable of addressing system level knowledge in large organizations, and tries to record the system level knowledge in an explicit fashion by proposing knowledge categories. Therefore, it falls into the upper righter quadrant in Figure 2-15.

The framework proposed here (Figure 3-1) follows the Axiomatic Design process [Suh (2000)]. The comparisons in Table 2-4 show that Axiomatic Design has one the most advanced design process. Also, the "product" domain in this knowledge management framework (Figure 3-1) addresses the negligence of emergent properties in Axiomatic Design. Therefore, this framework can be used to produce a live document of the system level knowledge starting at the beginning of the product development process and throughout

the product lifecycle. The "Why2" type of knowledge collects the learning about the system emergent properties. The end product of using this framework is a complete documentation of a project kept in an organization, which can serve easily as a lessons-learned for future activities because of the knowledge browsing capability of this framework.

This framework provides incentives for documentation from two perspectives. First, the "How1" and "Why1" types of knowledge keep track of the requirements flow down. System verification and validation plans thus can be made from these types of information. Design changes can also be evaluated by relating the effects of change back to customer requirements and enterprise strategy. Second, if a method is developed to link the "How1" and "How2" types of knowledge, then the documentation of requirements flow down can be automatically turned into the knowledge regarding the system interactions (this method is presented in Section 3.2). This "How2" knowledge can thus use the DSM partitioning techniques to help project planning and organization structuring. Therefore, documenting system level knowledge using the framework proposed here can not only benefit the organization in the long term, but also provide short-term benefits to the engineers who are documenting.

In summary, the proposed framework for knowledge management meets all the requirements for such a framework. Yet, this framework is just a hypothesis. This hypothesis needs to be tested on real engineering cases and will be revised based on our learning from cases. In addition, the method to transfer "How1" knowledge to "How2" knowledge needs to be developed. The "How1" type of knowledge can be recorded using the Axiomatic Design's Design Matrix (DM). The "How2" knowledge can be recorded using a Design Structure Matrix (DSM). In the next section, a matrix transformation method for converting a DSM from a DM is introduced, which enables us to transfer the "How1" knowledge into "How2" knowledge.

3.2 The Matrix Transformation Method

3.2.1 Motivation for Obtaining a Design Structure Matrix from a Design Matrix

The motivation for developing a transformation method between Design Matrix (DM) and Design Structure Matrix (DSM) really comes from the strength and limitations of both methods. In the Literature Review chapter of this thesis, DM and DSM are discussed in details (Table 2-3). Here is a summary of what they do well and what they don't.

The strength of DSM method is its system analysis capability. Once a DSM is built to describe the system interactions, there exist the method to partition the matrix so that unnecessary iterations in the system can be avoided. A piece of software has been written to transfer the partition result into GANTT Chart for project planning [Cho (2001)]. The DSM clustering algorithms can identify the most tightly coupled elements in the system and suggest system architecture. The DSM simulation models based on Markov chain can predict the likelihood of a project to complete on time and on budget. In short, DSM analysis tools provide us plenty of ways to manage system couplings.

The limitation of DSM method lies on the way a DSM is constructed. To build a DSM, we need detailed knowledge about the interconnections within a system. This detailed knowledge is usually only available during detailed design phase or for a mature product. Thus, traditionally a DSM is built by interviewing experienced engineers during the detailed design phase of a project. The results of DSM analysis, although usually reveal means for improvement, come too late in the development process after all the important decisions about a system has already been made. Therefore, the traditional DSM method is only helpful for mature products with little change from their last generation. In addition, the DSM construction relies on people's subjective judgment about where system interactions should occur. The way a DSM is constructed lacks objective knowledge on what system interactions need to occur and why. Therefore, improvements made based on the subjective expert knowledge may not be the optimal solution for a system.

The strength of the DM in Axiomatic Design is that it provides a way to look at system interactions from requirements perspective. Requirements are the ultimate goals of product design activities. DM provides a way of identifying system interactions based on how requirements are fulfilled through a design concept. Therefore, the construction of DM's prompts a design engineer to think of ways to reduce system couplings upfront. In addition, requirements are available at early stage of the design process. Thus the knowledge of system interactions can be obtained from early on in the design process when the most important decisions about a system are made.

There are two limitations about the Axiomatic DM. First, the DM is constructed under the guideline of Axiom 1. Couplings and iterations in a system are not allowed. Consequently, DM is also unable to deal with couplings if they exist in a system. From earlier discussions in Chapter 2, we know that couplings are not easy to avoid for real large complex engineering systems. Therefore, DM although strives for the ideal solution, is incapable of dealing with the real world situations. The second limitation of the DM is that it assumes all requirements can be decomposed. This is rather a reductionism's view. Hence DM cannot address the interactions within a system due to system emergent properties.

	Design Matrix (DM)	**Design Structure Matrix (DSM)**
Can be applied at early stage of the design process	Yes	No
Can deal with system couplings	No	Yes
Can capture the emergent properties of a system	No	Yes
Does not rely on subjective understanding of the system interactions from experts	Yes	No
Have system analysis tools	No	Yes

Table 3-1: Comparison between DM and DSM

Table 3-1 summarizes the above discussions on DM and DSM. Compare DM and DSM, they complement each other in many aspects. Both methods are concerned with system interactions. Both methods try to provide solutions to improve system designs. If we could combine the strengths of these two matrices, we will have a way to:

- Reduce the amount of system coupling using good system design concept.
- Forecast system interactions before detailed design phase.
- When system couplings cannot be avoided for the reason of cost, technology maturity, etc., apply system analysis tools to manage system iterations, so that the project can go through the system interactions more efficiently.
- Capture system emergent properties as the design work carries out. Reflect the emergent properties back to the requirements they affect.

Now the question left is how to obtain a DSM from a DM. The inspiration of finding the answer comes from solving systems of linear equations using substitution.

3.2.2 A Look at Solving Systems of Linear Equations

The inspiration of transferring a DM into a DSM comes from linear algebra—how to solve a system of liner equations using substitution. Equation (1) and (2) show an example of such a problem:

$$3 * X_1 + 5 * X_2 = 6 \qquad (1)$$

$$2 * X_1 - X_2 = 4 \qquad (2)$$

Solve for X_1 and X_2.

There are many ways to solve this system of linear equations. The method we are about to use here is substitution. From Equation (1), we get:

$$X_1 = 2 - 5/3 * X_2 \qquad (3)$$

From equation (2), we get:

$$X_2 = 4 - 2 * X_1 \qquad (4)$$

Substitute (3) into (4), we get:

$$X_2 = 4 - 2 * (2 - 5/3 * X_2)$$

Therefore,

$$X_2 = 0$$

Substitute into equation (3), then:

$$X_1 = 2$$

The answer to the question is:

$$X_1 = 2 \text{ and } X_2 = 0$$

The process of writing X_1 in terms of X_2 and X_2 in terms of X_1 is the process of discovering the relationship between X_1 and X_2. From equation (3) and (4), we can conclude that X_1 and X_2 are coupled. If we were to solve these two equations numerically, we will have to iterate several times before we can get the actual answers. Methods of numerically solving systems of linear equations can be found in mathematics books [Strang (1986) and Steward (1962)].

3.2.3 The Three Steps to Transfer a DM into a DSM

The Design Matrix has many similarities to a system of linear equations. Let's take the Design Matrix below as an example:

	DP1	DP2	DP3
FR1	X	O	X
FR2	X	X	O
FR3	O	X	X

There are three Functional Requirements (FR) and three Design Parameters (DP). The "X" in the matrix represents the corresponding DP affects the fulfillment of the corresponding FR. The "O" means no relationship exists between the corresponding DP and FR.

Each row of the DM can be seen as a linear equation in a system of linear equations. Thus the DM above can be translated into:

$$FR1 = a_{11} * DP1 + a_{13} * DP3 \quad (5)$$

$$FR2 = a_{21} * DP1 + a_{22} * DP2 \quad (6)$$

$$FR3 = a_{32} * DP2 + a_{33} * DP3 \quad (7)$$

Where a_{ij} are coefficients. Solve for DP1, DP2, and DP3.

We can thus apply the substitution method to this set of equations.

DP3 = f(FR1, DP1) from equation (5)
DP1 = f(FR2, DP2) from equation (6)
DP2 = f(FR3, DP3) from equation (7)

According to mathematical definitions, DP3, DP1, and DP2 here are called the *Output Variables* of equation (5), (6), and (7), respectively.

Therefore, we can represent the relationship among the three design variables in the DSM below:

	DP1	**DP2**	**DP3**
DP1	X	X	O
DP2	O	X	X
DP3	X	O	X

Through the above procedures, we started with a DM and arrived at a DSM. If we compare the DM and the resulting DSM, we can observe that the DSM is the DM permuted by rows to move the output variables to the diagonal position. This rule holds true for all transformations in this nature [Steward (1962)].

Therefore, the above matrix transformation procedure can be summarized into three steps:

Step 1: Construct an Axiomatic Design Matrix.

	DP1	**DP2**	**DP3**
FR1	X	O	X
FR2	X	X	O
FR3	O	X	X

Step 2: Choose the output variables in each row (circled out in the matrix below).

	DP1	**DP2**	**DP3**
FR1	X	O	(X)
FR2	(X)	X	O
FR3	O	(X)	X

Therefore, we have:

DP3 = f(FR1, DP1) from row 1

DP1 = f(FR2, DP2) from row 2

DP2 = f(FR3, DP3) from row 3

Step 3: construct the final DSM by permute the rows of the DSM to move the output variables to the diagonal position, or by using the relationship in the equations in Step 2.

	DP1	DP2	DP3
DP1	X	X	O
DP2	O	X	X
DP3	X	O	X

We may also get a DSM for the Functional Requirements (FR) by permuting the columns of the DM instead of the rows, and then transpose the DSM. Without transposing the DSM, the information flow direction is from row headings to the column headings.

	FR1	FR2	FR3
FR1	X	O	X
FR2	X	X	O
FR3	O	X	X

3.2.4 The Choice of Output Variables

The three steps of matrix transformation seem quite straightforward. Yet, a careful observer may discover that the choice of the output variables in Step 2 is not unique. In the example in the last section, the output variables can also be chosen as the elements on the diagonal. The comparison of the both choices is shown in Figure 3-2. By choosing different output variables, we go through the iteration in different directions. Yet the iteration is involves the same system elements and the same interactions (without taking into account the direction of interactions).

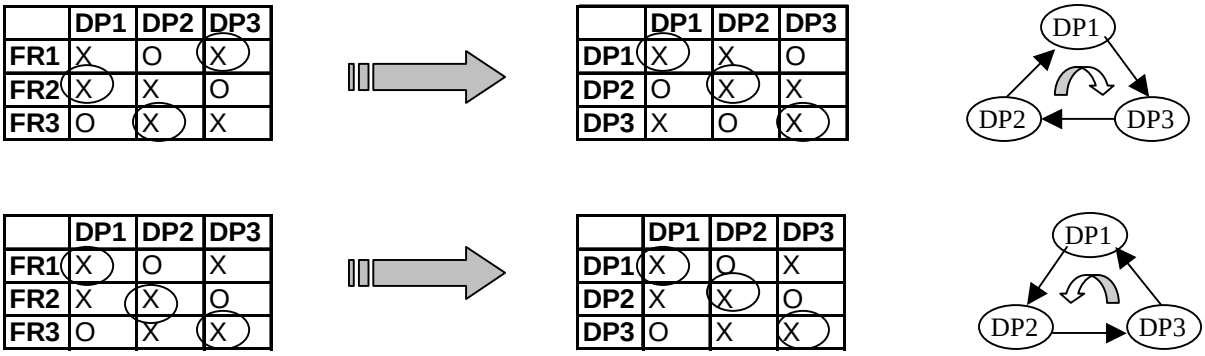


Figure 3-2: The Choice of Output Variables

In addition, there can only be one output variable in each row and column of the DSM [Steward (1962)]. Therefore, the choice of output variables is unique when the system interactions do not involve coupling, but rather are sequential or uncoupled. In this case, the output variables are always the diagonal elements. Only among the system elements that are coupled, the choice of system elements is not unique. Figure 3-3 shows this point. The choice of the output variable for the first two rows is unique because DP1 and DP2 are related in sequence rather than coupling. DP1 must be solved first before DP2 can. DP3, 4, and 5 have more than one output variable choices because they are coupled.

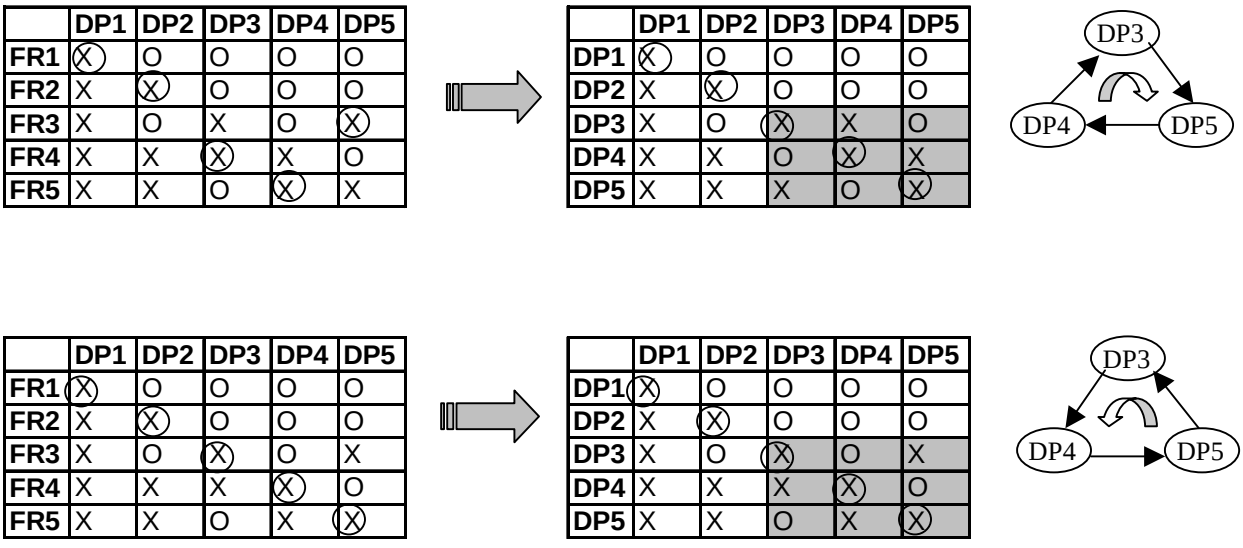


Figure 3-3: The Choice of Output Variables for Elements Not Involved in System Iterations

The choice of output variables does not matter very much in mathematics. Because the iteration is still the same iteration, the Eigen value is the same no matter which direction we take to go through the iteration. Therefore, there is little written about which output variable to choose in mathematics literatures [Steward (1962)]. However, when this technique is applied to a product development situation, different choices of output variables give different work procedures. What does this mean in a real engineering design project? Does one choice of output variables make the design process converge faster? These questions are answered in this thesis research from observations made in the case studies (see later Section 4.3.6).

3.2.5 Assumptions Used in the Matrix Transformation Method

Two assumptions are used in the matrix transformation method. Later on in the thesis, these assumptions are examined, and the limitations of the matrix transformation method are discussed.

Assumption 1: It is assumed that the Design Matrix (DM) is always constructed based on the Axiomatic Design method [Suh (2000)], and hence the DM is always square.

According to Axiomatic Design, a DM is constructed using the following steps:

1. Take a Functional Requirement (FR) and identify the Design Parameters that address the FR. Add a new row in the DM for the FR and a new column in the DM for the corresponding DP. Put a mark on the corresponding diagonal of the DP.
2. Identify other existing DP's that have side effects on the FR, and put off-diagonal marks in the DM.

The first consequence of this assumption is that the resulting DM from the above two steps is a square matrix, because each FR is identified with a DP. The consequence of this assumption is that the diagonal marks in the DM always carry more weight/importance than

the non-diagonal marks. For the FR-DP pair represented by the diagonal marks reflects the main reason for a DP to exist. The FR-DP pair represented by the off-diagonal marks shows only the side effects a DP has on another FR.

The second consequence of this assumption is that not all requirements for a product can be called Functional Requirements. A requirement like "Allow the passengers to enter/leave the vehicle" is a FR because we can identify a DP, such as "a door", for this FR. However, a requirement like "Meet XXX reliability measure" is not a FR because all parts of the system are important to this requirement, and no one dominant DP can be identified. Putting a reliability requirement into the DM will make the matrix have more rows than columns. Therefore, reliability requirement cannot be directly put into a DM like other FR's. Axiomatic Design calls these non-FR types of requirements "Constraints", and does not provide a systematic way to reflect the effects of constraints in the DM. Hence, with this assumption, the matrix transformation method introduced in this chapter carries the danger not to be able to represent all of the system interactions introduced by requirements.

Assumption 2: The second assumption of this method is also originated from Axiomatic Design theory. Constructing a DM to predict system interactions implies that all of the important requirements about a system can be decomposed. This is rather a reductionism's view. From the discussion in Chapter 1, we know systems have emergent behavior. Not all requirements can be decomposed clearly at the early phase of the design process. For instance, a machine may produce unexpected noise after all the components are assembled. The noise is then an emergent behavior of the system. The system interactions that contribute to meeting the noise requirement cannot be captured in a DM. Therefore, the reductionism's view Axiomatic Design takes may limit the capability of the matrix transformation method.

In short, both assumptions for this method reveal that employing the Axiomatic Design Matrix may limit our capability of predicting all of the system interactions. Yet, the matrix transformation method for obtaining a DSM from a DM is still worth trying for two reasons. First of all, people currently make decisions at early phase of the design with very limited

knowledge about the system. People cannot predict the emergent behavior of the system anyway. This matrix transformation method at least provides a framework to put everyone's mental model about the system on paper and reach a common understanding about the system. Plus, the system analysis methods and tools associated with DSM method can be used to help make better decisions at early phase of the design process. Second, prediction does not have to be perfect. People make judgments without complete information anyway. If the matrix transformation method can give more insights to the system, it will help with better decisions at the most important phase of the design process.

3.3 Design of the Two Case Studies

Two case studies were designed to test the method of obtaining a DSM from a DM in order to predict system interactions at early phase of the product development process. The knowledge management framework is also examined in one of the case studies.

3.3.1 CVC Case Study

The first case study was carried out at CVC between May and August 2000. CVC was a semiconductor manufacturing equipment producer located in Rochester, New York. CVC has been acquired by VEECO in spring 2000 and the merger complete in the fall of 2000. Besides the author, another Master's degree student Guru Prasanna [Prasanna (2000)] went to CVC as well in order to investigate whether the framework of managing system level knowledge proposed by the author was feasible.

The objectives of this case study corresponding to the research questions raised for this thesis (see Chapter 1) are the followings:

- Test the framework for managing system level knowledge—Guru Prasanna and Qi Dong(Q2-b)
- Discover the situation of documenting system level knowledge at CVC—Qi Dong (Q2-c, d)

- Testing the matrix transformation method—Qi Dong (Q1-b, c, and Q2-e)

Note the "Qm-n" types of symbols in the parentheses are the same notation used in Chapter 1 for each research questions.

3.3.2 Johnson and Johnson Ortho Clinic Diagnostics Case Study

The second case study was carried out at Johnson and Johnson's Ortho Clinical Diagnostics (JNJ OCD) between May and August 2001. Only the author conducted this case study. The research questions investigated in this case study include:

- Testing the matrix transformation method (Q1-b)
- Testing the correctness of the prediction DSM (Q1-b)
- Discover the completeness of the prediction (Q1-c)
- Discover the situation of documenting system level knowledge (Q2-c, d)

Again, note the "Qm-n" types of symbols in the parentheses are the same notation used in Chapter 1 for each research questions.

3.3.3 Strengths and Limitations on Learning from Case Studies

Case studies are necessary steps for product development research. Any method and tools development in research must be tested on real product development cases to discover their feasibilities and limitations. The system level knowledge management framework and the DM to DSM matrix transformation method developed in this chapter are no exceptions. Case studies are the time these methods are tested in real life.

The limitation of case studies is that it is hard to generalize the conclusions from a case study, because each case is a specific example. Something worked for one case may not work for another case. We need many cases with similar set up and controlled conditions to generalize a conclusion. Therefore, the readers of this thesis must keep in mind that the conclusions drawn from the two case studies may only be specific to these case studies.

Yet, the Scientific Method [Wilson (1952)] suggests that no hypothesis can be proven right. We can only find counter examples to prove them wrong. The hypothesis that stood correct through many cases are elevated to become law and theories. The methods proposed here to manage system level knowledge and to obtain a DSM from a DM can only be proven wrong from cases. We do not know how many product development examples we need to make a general conclusion. The value of the case studies are to find out whether the methods can stand strong for at least two cases, and the future directions for this research.

3.4 Progress Made Regarding Research Questions

In this chapter, the research questions addressed include Q1-b, Q2-b, c, and d.

Q1-b: How to predict system interactions early? How to predict system interactions for new technology?

The DM-DSM matrix transformation method introduced in this chapter is a way to predict system interactions early in the design process. The DM can be constructed during the concept development phase, because the information needed to construct a DM is available at that time. The requirements on the system are developed during concept generation. How requirements are met by the design parameters is decided at the same time too. DM collects this information and hence is very easy to construct during early phase of the design process. Once a DM is obtained, the matrix transformation method enables us to get a DSM from the DM. This DSM contains the prediction on system interactions before the detailed design work starts. This technique works for early phase of the design, and also works for new technologies.

The matrix transformation method will need to be tested on real engineering projects. The selection of the output set is still a question that needs further investigation in later chapters.

Q2-b: Is there a better way to capture, store, and represent system level knowledge?

A system level knowledge management framework is introduced in this chapter. This framework is based on Axiomatic Design's four domains for product development. It also added the fifth domain to capture the learning about the emergent properties of a system. Eight types of system level knowledge were proposed. They are *What, How, When, Where, Who, Which, Why, and Rules*. This framework may enable companies to manage system level knowledge throughout the design and development process.

How well this framework work needs to be tested. Whether the proposed categories of system level knowledge are complete also needs investigation through case studies.

Q2-c. What are the best sources of information for predicting system interactions?

If the knowledge management framework is used, the best source of system interactions will be the requirements decomposition document. This claim needs also to be tested out in case studies.

Q2-d: How companies are doing with managing system level knowledge?

From the author's Master degree thesis, it is known that system level knowledge is not well managed at Ford Motor Company. This thesis will look at two more companies and compare them with Ford Motor Company.

3.5 Summary

This chapter first introduced a system level knowledge management framework. This framework is based on Axiomatic Design's four domains for product development. It also added the fifth domain to capture the learning about the emergent properties of a system. Eight types of system level knowledge were proposed. They are *What, How, When, Where, Who, Which, Why, and Rules.*

Next, a method to obtain a Design Structure Matrix (DSM) from a Design Matrix (DM) is introduced. This method enables us to take advantage of the strengths of both matrices and

reduce the limitation of each method. This matrix transformation method will enable us to predict system interactions from early on in the design phase, so that we can plan the project and organize the people better in order to avoid unnecessary rework. This transformation method also enables us to systematically trace requirements throughout the system. The resulting matrices can serve as a life document throughout the product life cycle to capture the learning about the system level knowledge. Thus we can be less dependent on human experts, and keep the system level knowledge in the organization.

These two methods will be tested on two case studies—CVC, and Johnson and Johnson. Each case study serves to answer some of the research questions for this thesis. The details of each case study are introduced in the next two chapters.

4 CVC Case Study

4.1 The Research Setting

4.1.1 About the Company CVC

CVC was a semiconductor manufacturing equipment producer. Its headquarters were located in Rochester, NY. In spring 2000, CVC was acquired by VEECO. The merger started in May 2001 and went through August 2001. CVC now is a subdivision of VEECO. When this case study was set up in January 2001, the acquisition of the company was not known yet. When the case study was carried out during summer 2000, CVC was undergoing a transition from the merger. The author is very thankful to many engineers and managers at CVC who provided great help and support to this research work even though they were going through a lot of changes themselves.

Before the merger, CVC's core competency was the data storage process equipment. It was in fact the market leader in this area. CVC produced cluster machines that can manufacture thin film magnetic heads for data storage purposes. The cluster machines consist of a central wafer handling system and many wafer processing modules around it. The processing modules can be physical vapor deposition modules, ion-beam deposition modules, ion-beam etching modules, etc. When a wafer enters the system, the deposition and etching processes it has to go through (called the "recipe") is already programmed into the control system. A robot in the Central Wafer Handler will take the wafer from its storage area and put it into the first processing chamber. After the first step, the robot will take the wafer out of the first processing chamber and place it into the second module, and so on, until the entire recipe is completed. Figure 4-1 shows a typical cluster machine CVC produced, and now is listed as VEECO products (http://www.veeco.com). For the throughput efficiency, usually more than one wafer or even more than one kind of recipe are processed simultaneously. Therefore, the logistics of the central wafer handler is important to the capability of this manufacturing equipment.

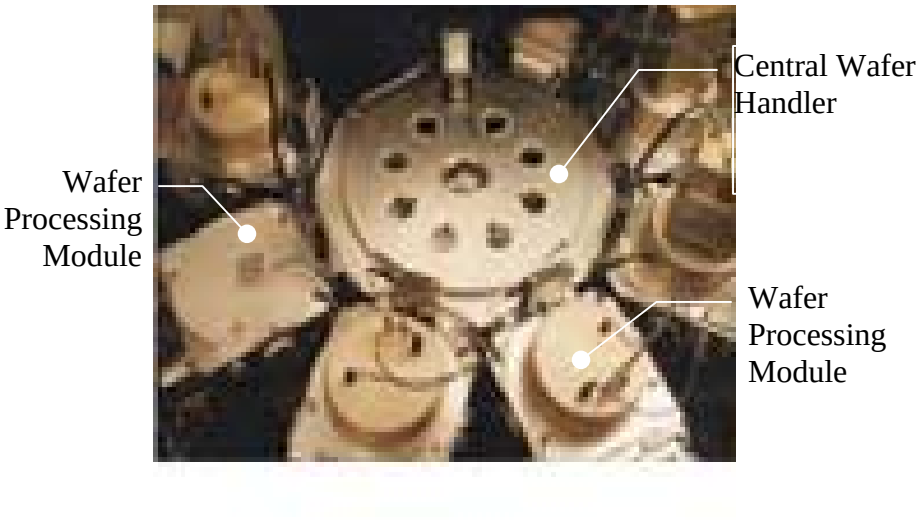


Figure 4-1: A Typical CVC Cluster Machine

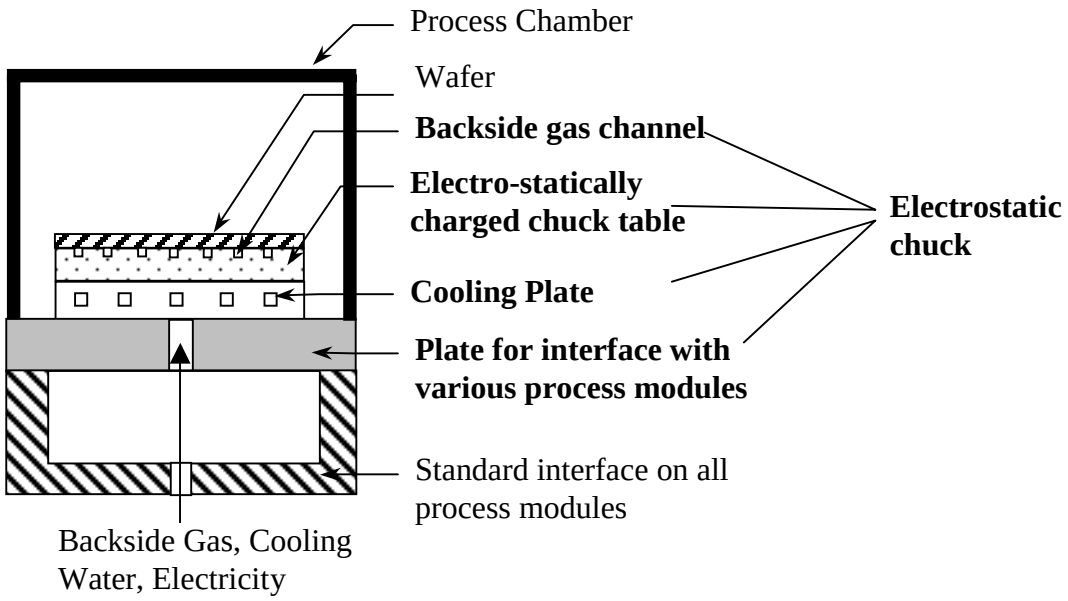
Before the merge, CVC was a company with about 400 employees. It had acquired several smaller companies and laboratories over the past years. With about 200 employees located in Rochester, NY, CVC had the rest of the employees in Virginia, Texas, California, etc. The differences in working culture and the difficulties of communication among these sites were increasing challenges for CVC. CVC was interested in DSM as a way to structure collaborations among different locations in order to facilitate the system engineering effort of their projects.

4.1.2 Case Study Description

4.1.2.1 The Product

This research case study concerns the system integration project for an Electrostatic Chuck. The Electrostatic Chuck—ESC (Figure 4-2) is used in various wafer process modules in the cluster machines (Figure 4-1). When in use, ESC loads the chuck table and the wafer with opposite static charges, and the wafer is held down on the chuck table by the electro-static force. After the wafer is processed, the chuck table and wafer are discharged, and the wafer is de-clamped. Contrary to a conventional mechanical clamp, ESC does not exert contact

force on the processing side of the wafer. Hence, ESC is particularly suitable for processing wafers plated with brittle materials. ESC also contains backside gas channel and cooling system designed to maintain wafer temperature during processing. Each ESC has an interface plate that is designed so that the ESC can be assembled in all wafer-processing modules CVC is selling in the market. Based on Henderson and Clark's (1990) four types of product innovation (see Figure 2-7), the ESC project belongs to the modular innovation. The core concept of the wafer chuck design is changed. The architecture interface between the chuck and the cluster machines however remain the same.



Note the bold characters indicate the elements that are actually in the ESC. The regular characters indicate components that are not part of the ESC.

Figure 4-2: Schematic of the Electro-static Chuck

4.1.2.2 Case Study Objectives

4.1.2.2.1 CVC's Objectives

CVC recognized the market need to replace the mechanical chucks in many of their existing process modules with ESC. By April 2000, the advanced research group in Dallas, Texas completed the technical feasibility study of the ESC as a stand-alone component.

Recommendations were made to the Rochester design-engineering group regarding the design specifications of ESC. The Rochester engineering department must work out the details to integrate the ESC concept into existing modules.

The system integration of the ESC faced two challenges. First, the integration had to be completed within six months so that the ESC feature could reach the market before the competitors did. Second, due to the geographical distance between the Texas group and the Rochester group, the communication across organization boundaries had been historically problematic. The advanced research engineers in Texas sometimes miss the details of the production units during testing. The product-engineering department in Rochester also could send the wrong information to Texas.

Therefore, CVC's main objective in this case study was to use the matrix transformation method to obtain a DSM before the integration work was really carried out, so that mistakes could be prevented, and they could complete the project in time. CVC's second objective was to use this project to discover how to better set up a communication channel among the various dispersed divisions all over the nation.

4.1.2.2.2 Thesis Research Objectives

The objectives of the author's research matched very well with CVC's interests. The matrix transformation method was to be tested to see if it works in a real engineering project. If the method worked, then the result would be a DSM that predicted what would occur during the system integration phase of the ESC project. In addition, the author was interested in the existing situation of system level knowledge documentation in CVC, and to see if the system level knowledge management framework was applicable in an industry setting. These two research goals matched with CVC's agenda for understanding what could be done to improve the communication among their organizations.

The research objectives of this case study are summarized here again. The parentheses indicate which research question is related to which objective.

- Test the framework for managing system level knowledge—Guru Prasanna and Qi Dong (Q2-b)
- Discover the situation of documenting system level knowledge at CVC—Qi Dong (Q2-c, d)
- Test the matrix transformation method—Qi Dong (Q1-b, c, Q2-e)

4.1.2.3 Case Study Scope

This case study is interested in the system interfaces of the ESC design. The details of the ESC design itself and the details of the design of the rest of the cluster machine are not the focus.

4.2 Data Gathering Process

4.2.1 Sources of Inputs

The information sources for this case study include two types—the documents and the people. The documents include:

1. ESC design specifications
2. The technical report from the advanced research group in Texas
3. ESC preliminary design drawings

The people who provided information about the project are:

1. Matthew Coon, CVC Rochester System Engineer, ESC project manager before July 15, 2000.
2. Shawn Chen, CVC Garland, TX advanced project development engineer.
3. Steve Buckner, CVC Rochester software and controls engineers. He is the primary software developer for the ESC project.
4. Gary Denton, CVC Rochester software and controls project manager.
5. Blake Reese, CVC Rochester Electrical Engineer.
6. Judd Prozeller, CVC Rochester Vice President of Quality.

Due to the merger of CVC and VEECO, CVC experienced substantial restructuring. By the end of the summer, many of the people listed above were no longer CVC employees. Consequently, the progress of the ESC project as well as the information gathering process of this research project was affected. The information gathering stage for this project took 1.5 months, while it could have taken less than a month in a normal condition.

4.2.2 Constructing a DSM from Requirements Using the Matrix Conversion Process

The following steps were taken in order to apply the DM to DSM transformation in this case study.

1. Understood the ESC project background, and the progress made in the design. Matthew Coon was the main contact for this information in January 2000. Shawn Chen was the main contact for this information in March 2000.

2. Constructed the Design Matrix (DM) for the ESC system integration, starting in May 2000. This step can be split into two sub-steps:

    2.1. First, the highest-level design specifications for the ESC project were obtained from the project document.

    2.2. Next, the author used the zig-zagging method in the Axiomatic Design [Suh (2000)] to decompose the requirements and to relate them to the Design Parameters (DP). CVC did not have lower level design requirements documents. Therefore, the process of constructing the DM was also the process that CVC engineers decided the subsystem requirements for the ESC. Mathew Coon and Shawn Chen helped to construct hardware subsystem requirements. Steve Buckner constructed the software requirements.

3. Derived the DSM from the DM using the three steps for matrix transformation, where the diagonal elements were selected as output variables. Other possible selections of output variables were also tried.

4. Verified the DSM with the Advanced Product Development engineer Shawn Chen, the software engineer Steve Buckner, the software manager Gary Denton, and the Electric Engineer Blake Reese. The above individuals revised the DSM obtained from DM.

5. Compared the DSM revised by the individuals listed in Step 3 with that derived from the Design Matrix.

6. Partitioned the Design Structure Matrix for the planning of the ESC integration.

7. Constructed the function decomposition diagram of the ESC requirements and the architecture decomposition of the ESC.

4.2.3 Testing the System Level Knowledge Management Framework

A Master's degree student from MIT—Guru Prasanna—came to CVC during the same period. Guru was assigned to review the following documents:

- CVC Metal Organic Chemical Vapor Deposition (MOCVD) module chuck design documents
- CVC ESC project design documents
- CVC ESC design DSM (see Figure 4-8)
- Ford's design documents for throttle body
- Ford's throttle body design DSM's [Dong (1999)]

His goal was to see if all of the documented knowledge in the above cases could fit into the categories in the proposed system level knowledge management framework (See Chapter 3). The details of Guru's work are in his thesis [Prasanna (2000)]. As a short summary, Guru read each line of the CVC and Ford design documents, as well as the DSM's built by the author for the CVC ESC project and Ford throttle body. Lines of text in the documents were used as units of counting. The types of knowledge in each line were counted. For

ambiguous sentences, Guru discussed with the author of this thesis to decide if new categories were needed or if the definition of the existing categories should be broadened.

4.3 Results and Analysis

4.3.1 The Design Matrix for ESC Integration Project

The Design Matrix for the ESC system integration is shown in Figure 4-3. The size of the DM is too large to show all the details on one page. Therefore, the first FR and its decomposition are shown in Figure 4-4 with more visible details. Note the DM includes six levels of decomposition. In order to demonstrate the decomposition structure of the FR and DP, again the FR1 is used as an example. The decomposition structure of the FR1 and DP1 are shown in Figure 4-5 in a tree structure. By following the Axiomatic Design's zigzagging process, the function flow structure and the product architecture tree are generated. The DM in Figure 4-3 includes all six levels of decomposition. The interactions that occur among lower-level elements are also reflected by the interactions among higher-level elements. We can also hide other levels and only show one level at a time. In Figure 4-6 only the first level of FR and DP are shown. We can do the same for other levels of FR and DP.

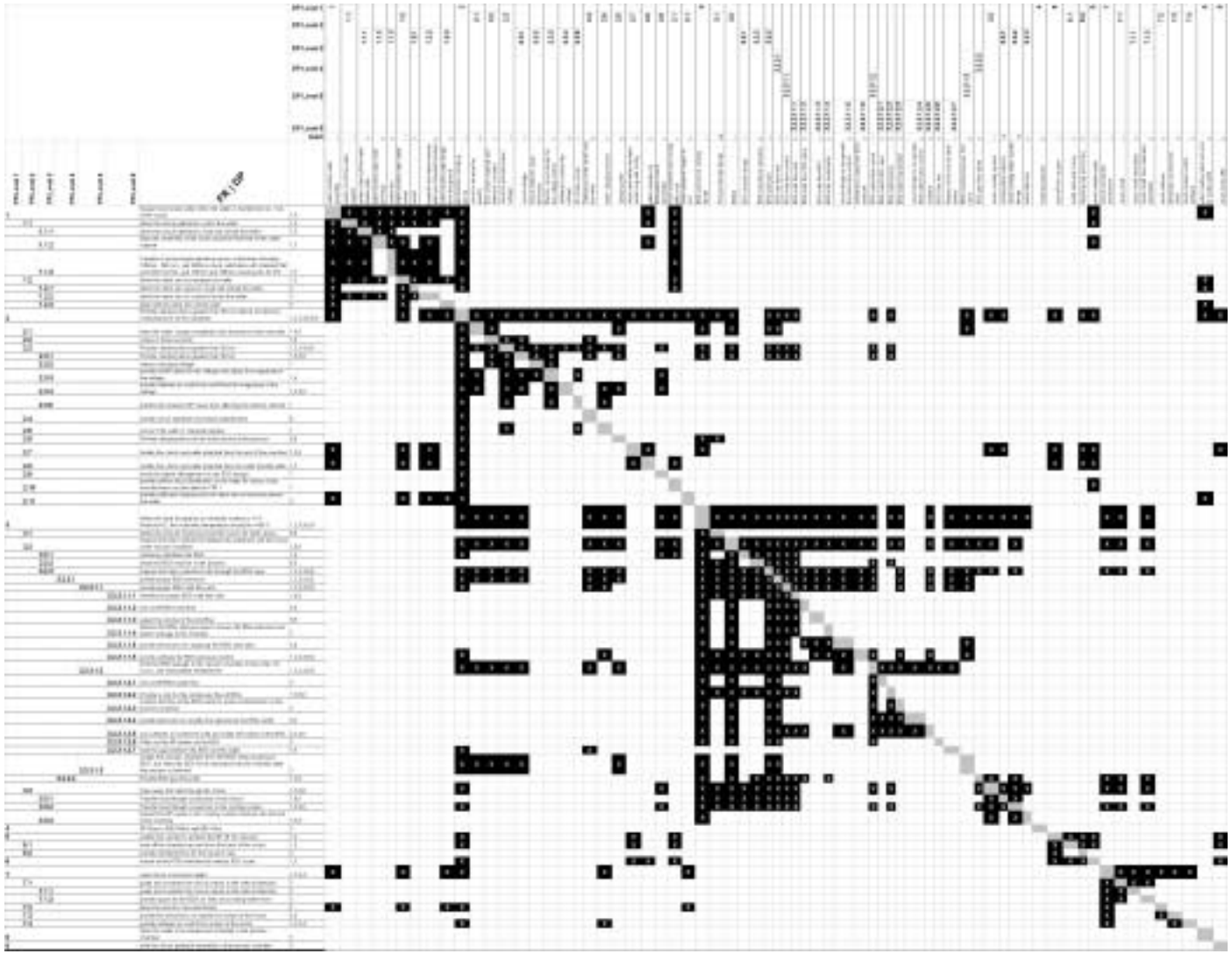
In addition, each DP can be assigned to a person or a group (see the "team" assignment in the DM's in Figure 4-3 and Figure 4-4). Then looking across the row, we can easily identify the groups that are responsible for each functional requirement. The FR's that are assigned to more than one team deserve special attention because they may cause issues in system integrations.

Furthermore, the sources of each FR can be identified and recorded in the DM (Figure 4-4). This information can be valuable at time when certain requirements are in question. The same thing can be done for DP's. In addition, the reason for each mark in the DM to exist can also be recorded in a database. The database used in this case study employed a simple EXCEL worksheet (Figure 4-7). This database proved to be helpful to keep track of the rationales behind each interaction in the DM. When disagreement rose among engineers who

provide information for this research, this database helped to keep the understanding in the team consistent.

Based on the Axiomatic Design Axiom I [Suh (2000)], a good design should have its DM as a diagonal matrix (uncoupled design) or lower triangular matrix (decoupled design). The Design Matrix in Figure 4-3 is a coupled design and hence is not acceptable by Axiomatic Design standards. According to Axiomatic Design, CVC should redesign the rest of the cluster machine and the electrostatic chuck so that the entire system is not coupled.

However, the real situation is that CVC had been selling the cluster machines for many years. They are pressured for delivery time to beat the competitors. CVC does not have time to redesign everything, and it cannot convince its customers to invest in new cluster machines just because electrostatic chucks are installed in the process modules. CVC had to work with the less ideal situation—to live with the existing design of the system and make sure the ESC can be integrated into the existing machines. Therefore, in this case study, it is unrealistic to push for the Axiomatic Design's Axioms. The matrix transformation method introduced in Chapter 3 is going to use the power of the DSM method to deal with the inevitable system couplings in this design.



Teams: 1—Supplier, 2—Software, 3—Mechanical Team, 4—Electrical Team, 5—R&D, 6—Marketing, 7—Process engineering, 8—Operation procedure

Figure 4-3: The ESC System Integration Design Matrix

				DP Level 1	DP ID	1								
				DP Level 2			1.1				1.2			
				DP Level 3				1.1.1	1.1.2	1.1.3		1.2.1	1.2.2	1.2.3
				team	team	1	1	3	3	3	1	3	3	3
FR Level 1	FR Level 2	FR Level 3	FR / DP		source of FR	wafer transfer plate assembly	center cut of the wafer support	center cutout of the wafer trasfer plate	wafer transfer plate shaft bushings	cutout lips	wafer transfer plate radial cutout	cutout	standoff and wafer transfer plate pin (standard location)	wafer transfer plate height from the ground
1			Support and locate wafer when the wafer is transferred into / out-of the chuck	1,3	Matt Coon from the drawings		X	X	X	X	X	X	X	X
	1.1		allow the chuck pedestal to catch the wafer	1,3	Matt Coon from the drawings	X		X	X	X	X	X	X	
		1.1.1	allow the chuck pedestal to load and unload the wafer	1,3	Matt Coon from the drawings	X	X		X	X				
		1.1.2	align the centerline of the chuck pedestal with that of the wafer support	1,3	Matt Coon from the drawings	X	X	X		X	X		X	
		1.1.3	Capable of processing/supporting various substrates including: 100mm, 150 mm, and 200mm round substrates with standard flat and notch for SA, and 125mm and 150mm round pucks for DS.	1,3	White Paper Specifications (2)	X	X	X			X	X	X	
	1.2		allow the robot arm to transport the wafer	1,3	Matt Coon from the drawings	X	X	X	X			X	X	X
		1.2.1	allow the robot arm space to load and unload the wafer	3	Matt Coon from the drawings	X					X			
		1.2.2	allow the robot arm to correctly locate the wafer	3	Matt Coon from the drawings	X	X	X	X		X	X		
		1.2.3	align with the robot arm motion path	3	me	X					X			

Teams: 1—Supplier, 2—Software, 3—Mechanical Team, 4—Electrical Team, 5—R&D, 6—Marketing, 7—Process engineering, 8—Operation procedure

Figure 4-4: ESC DM with only the Decomposition of FR1

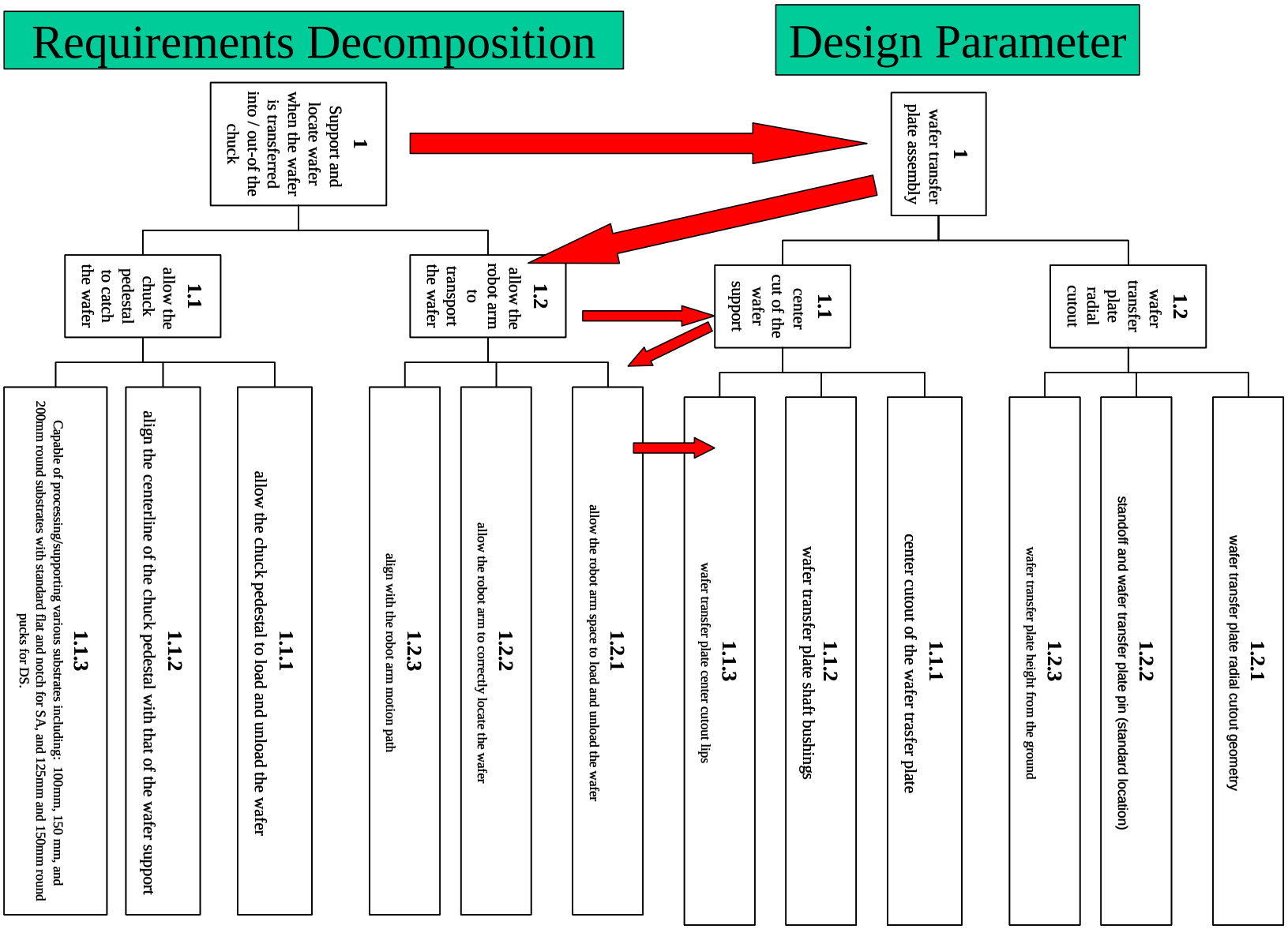


Figure 4-5: ESC Functional Requirement 1 Decomposition Diagram

		DP Level 1	1	2	3	4	5	6	7	8	9
FR Level 1	FR / DP	Team	wafer transfer plate assembly	electrostatic clamping device	BSG and chuck cooling design	matching network	seal off the vacuum	chuck adapter plate	Chuck movement mechanism	wafer loading robot arm and its motion path	chuck body (36)
1	Support and locate wafer when the wafer is transferred into / out-of the chuck	1,3		X				X		X	
2	Provide clamping force greater than 30 torr without mechanical clamping force on the substrate.	1,2,3,4,5,6,8	X		X		X	X		X	X
3	When the heat dissipation at substrate surface is >=3 Watts/cm^2, the substrate temperature should be <=60 C	1,2,3,4,5,6		X					X		
4	RF Bias to 600 Watts and 600 Volts	3									
5	enable the system to achieve the 5E-10 Torr vacuum	1,3		X				X			X
6	mount on the PVD machines for various ESC sizes	1,3		X			X				X
7	raise chuck to process height	1,2,3,4	X	X							
8	allow the wafer to be loaded and unloaded in the process chamber	3									
9	hold the chuck pedestal assembly in the process chamber	3									

Figure 4-6: ESC DM with Only the Highest Level of Decoomposition

FR ID	FR	DP ID	DP	why	source
2.2	clamp in three seconds	2.2	ESC chuck material and dielectric constant	Dielectric constant of the chuck material contributes mainly to how fast the chuck can clamp.	Shawn Chen 6/27/00
2.3.2	reduce the clamping voltage	2.3.2	Chuck dielectric layer thickness	Thinner dielectric layer can provide higher clamping force with the same voltage	TARAPDG0927991 page 3
2.3.2	reduce the clamping voltage	2.2	ESC chuck material and dielectric constant	The material of choice will influence the thickness of the dielectric layer.	TARAPDG0927991 page 3
2.3	Provide clamping force greater than 30 torr	2.2	ESC chuck material and dielectric constant	clamping force is a function of the voltage/thickness*dielectric material constant	Shawn Chen

Figure 4-7: Database to Record the Rationale behind Each Interactions in the DM

4.3.2 The DSM Obtained from the DM Using the Matrix Transformation Method

After obtaining the DM, the method of converting DM into a DSM was used. First of all, the DSM could be constructed for either one level of decomposition or the entire DM including all levels. The method proposed in Chapter 3 works for either choice. The purpose of obtaining a DSM was to analyze the system interactions, such as applying the DSM partitioning algorithm to identify sequence and iterations in the system, and the partitioning algorithms must be applied to a single level of the system. Otherwise, the coupling identified is useless, because higher-level system elements are always coupled with its lower level elements after decomposition. Therefore, a DSM was constructed for each level of the DM decomposition. The most useful was the lowest level of the DM decomposition, because the lowest level of DP's represented the system elements engineers actually worked on individually or as small groups.

The second decision to be made was the choice of the output variables during the matrix transformation. In Chapter 3, it was demonstrated that although the choice of output variables is not unique for systems involving iterations, the choice for elements that are not involved in iterations is unique. The choice of elements inside the system couplings is not unique, but does not affect the identification of the variables involved in the iteration blocks. Therefore, the diagonal elements in the DM were chosen as the output variables for now. Other choices of output variables and their implications will be discussed in the next section.

A DSM was then obtained by selecting the diagonal elements of each row as the output variable set. The DSM was partitioned using the partition algorithm in Warfield (1973). Two major iteration blocks were identified (Figure 4-8).

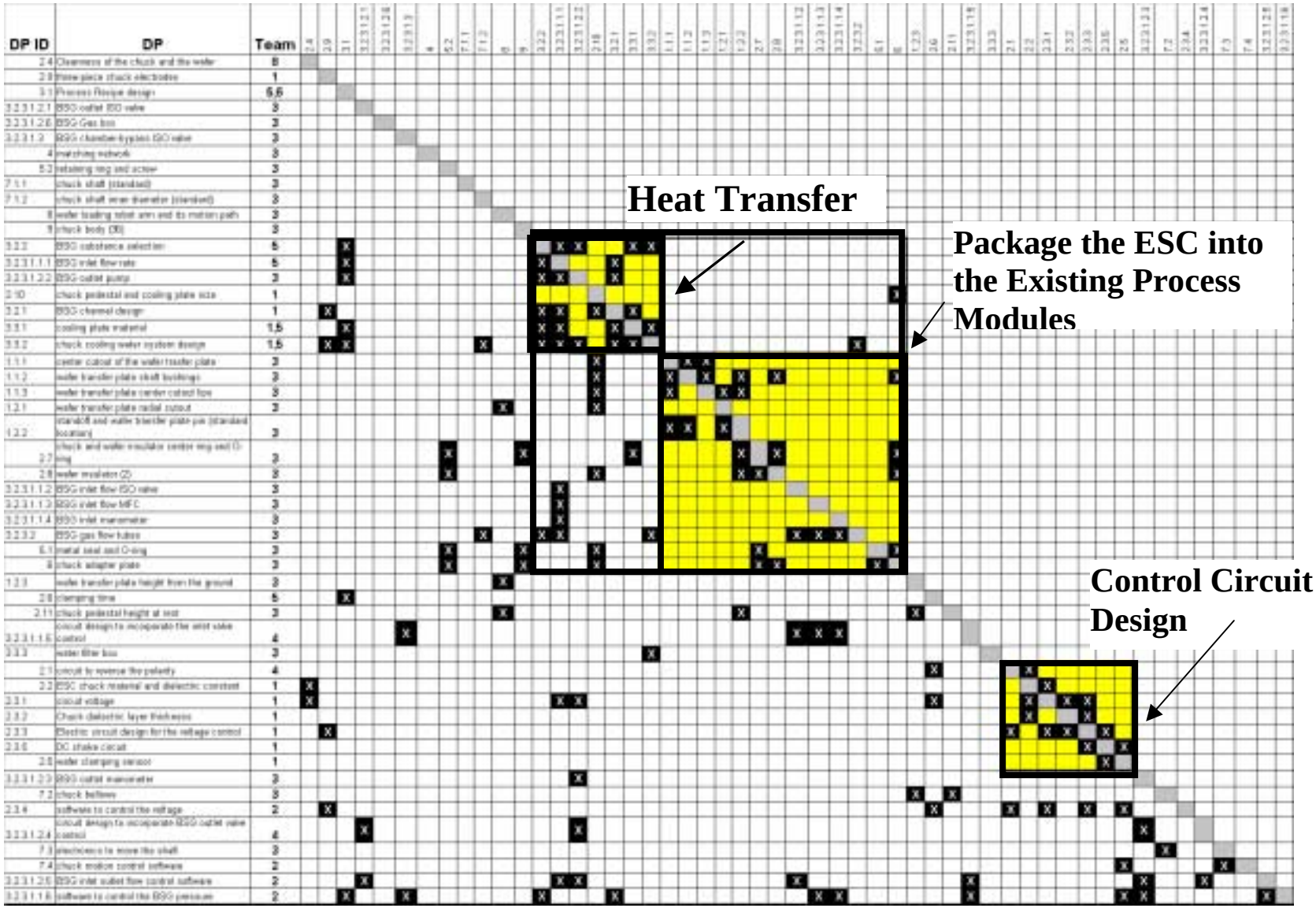


Figure 4-8: DSM of the Lowest Level Elements for ESC Integration

4.3.3 Benefits of the DSM Obtained from DM

4.3.3.1 Prediction of System Interactions

The first benefit of the DSM in Figure 4-8 is that the it serves as a prediction of what was going to happen in the ESC system integration work. Further reading the DP's that are in the DSM iterations, we can identify that the first iteration block involves two issues. The first issue must be resolved is taking the heat away from the wafer and maintain the wafer at a certain temperature during the process. The second issue involves the packaging of the ESC into the existing modules. The second iteration is about the circuit design. Therefore, before the integration phase started, the CVC engineers already were informed about the issues that might occur during system integration. They could be much more prepared for their work and reduce the amount of unnecessary rework.

4.3.3.2 A Requirement-driven Process

The second benefit that such a prediction DSM brings is a prescribed design process that is based on the product requirements and the product design itself, rather than people's subjective decision on what information needs to be exchanged among design teams. For instance, in order to fulfill the requirement of taking the heat away from the wafer during processing and maintaining the wafer temperature at a certain level, the Backside Gas (BSG), the chuck material, and the chuck cooling channel design form a heat transfer circuit (Figure 4-9). Given a certain goal of heat transfer rate, the above three elements can trade off against one another to achieve the same goal and at the same time lower the cost of the design. For instance, we can choose an expensive substance to use in the chuck cooling channels so that the chuck material does not have to be expensive. Or we can design the BSG substance so that the BSG conducts heat well. Then the requirements on the chuck cooling channel substance will not need to be high. Expressed in a heat transfer equation, the relationship among the three DP's are:

$$\left(L_{BSG}/k_{BSG} + L_{chuck}/k_{chuck\ material} + 1/h_{cooling\ fluid}\right) * \left(T_{wafer} - T_{cooling\ channel\ substance}\right) = dq/dt$$

Where

L_{BSG} is the thickness of the Back-side Gas;

k_{BSG} is the heat conduction coefficient of the Back-side Gas;

L_{chuck} is the thickness of the Chuck material;

$k_{chuck\ material}$ is the heat conduction coefficient of the chuck material;

$h_{cooling\ fluid}$ is the heat convection coefficient of the cooling fluid;

T is the temperature;

dq/dt is the heat transfer rate per unit area.

Clearly, the heat transfer equation shows the three DP's are coupled in the design in order to fulfill a common requirement.

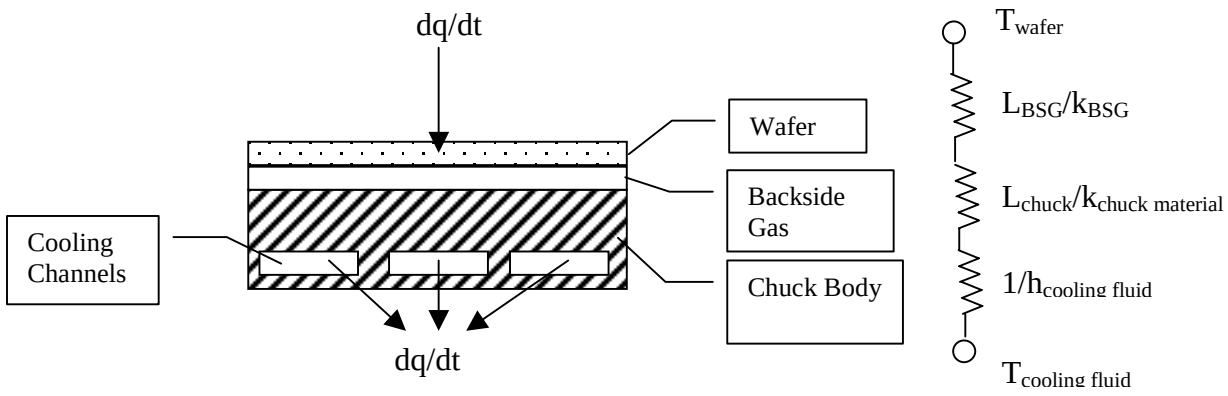


Figure 4-9: Wafer Cooling Heat Transfer Circuit

The DSM in Figure 4-8 captures this coupling. The heat transfer problem existed in every chuck CVC designed. Therefore, CVC engineers had prior experience about this design problem. When the DSM in Figure 4-8 was presented, the CVC engineers pointed out that they always only individually optimized the three elements in the heat transfer circuit, and hoped that the cost and performance requirements would be met when the system came together. The couplings among the three DP's were not taken advantage of. CVC engineers

might have always over designed the heat transfer circuit without knowing they could do better. Figure 4-10 shows the comparison of the two approaches. From this example, we can see the DSM obtained from DM took a product view, and showed the engineers ways to improve the way they always did their design.

The DSM from the DM:

BSG substance selection		X	X			X	X
BSG inlet flow rate	X				X		
BSG outlet pump	X	X			X		
chuck pedestal and cooling plate size							
BSG channel design	X	X		X		X	
cooling plate material	X	X			X		X
chuck cooling water system design	X	X	X		X	X	

The DSM that shows how CVC engineers approached the design in the past:

BSG substance selection		X	X			O	O
BSG inlet flow rate	X				X		
BSG outlet pump	X	X			X		
Chuck pedestal andn cooling plate size							
BSG channel design	X	X		X		O	
Cooling plate material	O	X			O		X
Chuck cooling water system design	O	X	X		X	X	

Note: highlighted rows indicate the three heat transfer elements involved in the heat transfer circuit in Figure 4-9. The row and column headings are the same in both DSM's.

X: interaction exists between the row heading and the column headings.

O: interaction exists in only the DSM from DM, not the DSM based on CVC engineers' approach to the design.

Figure 4-10 The Heat Transfer Design Problem

Another interesting observation is that the DSM obtained from the DM (the first DSM in Figure 4-10), although contains more complicated system couplings, represents a more scientific approach to the design of the system by reminding the engineers of system trade-offs. The second DSM in Figure 4-10 although look simpler, is a less optimal approach to

the system design. Therefore, more coupling in the DSM did not equal to a worse design approach in this example.

4.3.3.3 Manage the Communication Across Organization Boundaries

One of the proven benefits of DSM [Dong (1999)] is that it can show how people in different organizations should interact with one another in order to manage the design issues regarding the system interfaces. In the DM and the DSM (Figure 4-4 and Figure 4-8) for the ESC project, the teams that were responsible for each Design Parameters were indicated in the matrices. Therefore, the DM and DSM could become maps to guide the communications across organization boundaries. For CVC's situation of geographically dispersed design teams, DSM can be especially useful.

4.3.3.4 Summary of the Benefits

In summary, the DSM obtained from the DM provided the CVC engineers a prediction to the system interactions. Applying the partitioning algorithm, we were able to identify the iteration areas of the system before we start the actual integration work. This would be helpful for managing the ESC system integration project. Second, the DSM constructed from DM prescribed system interactions. Some of the prescribed interactions helped the CVC engineers to see beyond how things were always done, and to find the underlying structure of the design of the system. Third, the DSM with work assignments indicated is especially helpful for CVC to overcome the challenges in managing the communication across organization boundaries.

4.3.4 When to Stop the Decomposition in the DM and DSM Construction

Rechtin (1991) has once said "system is unbounded." The system decomposition of the requirements and the design parameters can go on forever using the tree structure. Too much detail is not necessary for the purpose of planning. Where is the limit of decomposition for the Design Matrix? The rules of thumb used in the CVC case study are as follows:

1. Select the level of system of interest. Within a system, the parameters that contribute to the system-level interaction should be listed. The interaction within each system component can be ignored. In this case study, the interactions within ESC that do not concern ESC system interfaces were not considered.
2. The items that are usually assigned to one person should not be further decomposed unless a critical detail about the system level interaction will be missed otherwise.
3. Too much detail never hurts to gain the insights. Details can always be eliminated if after the DSM partitioning, they seem not to reveal additional insights to the nature of the system level interaction. The trade-off with details is the time spent on the analysis. The less detailed the DM and DSM are, the less time is needed for analysis. Huge DSM involving all detailed system elements take a long time to partition and to analyze.

In order to test whether the ESC chuck DM and DSM has taken proper level of details, two checks were performed. The first check was that the author interviewed the engineers on the project including Shawn Chen, Steve Buckner, Gary Denton, and Blake Reese. These engineers not only reviewed the DSM, but also judged whether the level of detail was appropriate. After reviewing, all of the above engineers agreed the DSM has captured the appropriate level of details for the system interface for the design parameters that are in their responsibility.

The second check performed was on the DSM itself. Since the design parameters in the DM had a tree structure (Figure 4-5), DSM's can be constructed for each level of the tree. The deepest branch of the tree in this case study has six levels. Some branches have fewer levels than six. The highest level in the tree is given an ID using an integer such as "1". The second level follows its super level by a dot and an integer, such as "1.1" and "1.2". The subsequent levels are denoted following the same rules (see Figure 4.5).

By only considering the interactions between the leaf elements on each branch of the decomposition tree in the DM (see Figure 4-3), the DSM in Figure 4-8 can be constructed. By only considering the interactions among the elements one level higher than the leaf

elements, the DSM in Figure 4-11 is constructed. Figure 4-12 is the DSM with nodes two level up from the leaves, and so on (Figure 4-13, Figure 4-14, and Figure 4-15). Note for branches with fewer levels than six, when the top level is reached, the top level elements stayed in the subsequent DSM's. For instance, FR and DP 7 have only three levels in the tree. Therefore, starting at the DSM in Figure 4-12, the DP 7 stayed in each of the higher level DSM's (Figure 4-13 through Figure 4-15).

Comparing the partitioned DSM's at various decomposition levels of the design parameter tree, we can make the following observations:

- First, the details of the design parameters included in the leaf-level DSM (Figure 4-8) are necessary. Compare the DSM for the leaf-level (Figure 4-8) to the DSM at the fifth level (Figure 4-11) and fourth level (Figure 4-12), certain tasks in the super-category of 2, 3, 5, and 7 can be done earlier without getting involved in the iterations in the leaf-level DSM. Therefore, the details contained in the leaf level DSM help for the planning and completing tasks in parallel.

- Second, compare the DSM's that are two levels or more beyond the leaf-level (Figure 4-13, Figure 4-14, and Figure 4-15), they all have the same topology. Only one iteration block is identified. It seems like everything interacts with everything else in this project, which is not far from what people expect anyway without the aid of DSM tools. Added details towards the leaf level show there are actually two iteration blocks. The first one involves the packaging of the chuck into the PVD machine, and the second one involves the circuit design for the voltage control. Therefore, the additional details enable us to take advantage of the DSM and let the interactions behind the design parameters to help us to plan out the design tasks.

The above observations hold true for the CVC ESC case study. For other cases, the decision of when to stop the decomposition may be different. In any case, this decision is a management judgment. The lessons learned from this case study can be used for future cases.

DP ID	DP	2.4	2.9	3.1	3.2.3.1.3	4	7.1	8	9	2.6	1.1	1.2	2.1	2.2	2.3	2.5	2.7	2.8	2.1	3.2.1	3.2.2	3.2.3.1.1	3.2.3.1.2	3.2.3.2	3.3	5	6	2.11	7.2	7.3	7.4
2.4	cleanness of the chuck and the wafer																														
2.9	three-piece chuck electrodes																														
3.1	process recipe design																														
3.2.3.1.3	BSG chamber bypass ISO valve																														
4	matching network																														
7.1	chuck shaft																														
8	wafer loading robot arm and its motion path																														
9	chuck body																														
2.6	clamping time			X																											
1.1	center cutout of the wafer support											X						X	X								X				
1.2	wafer transfer plate radial cutout							X			X								X												
2.1	circuit to reverse the polarity									X				X																	
2.2	ESC chuck material and dielectric constant	X													X																
2.3	circuit to provide proper voltage	X	X							X			X	X		X						X	X								
2.5	wafer clamping sensor														X																
2.7	chuck and wafer insultaor center ring and O0ring								X			X						X							X	X	X				
2.8	wafer insulator (2)											X					X		X							X	X				
2.10	chuck pedestal and cooling plate size																										X				
3.2.1	BSG channel design		X																X		X	X									
3.2.2	BSG substance selection			X																		X									
3.2.3.1.1	BSG inlet flow control			X	X											X				X	X		X								
3.2.3.1.2	BSG outlet flow control			X																X	X	X									
3.2.3.2	BSG gas flow tubes						X														X	X			X						
3.3	chuck cooling system		X	X			X													X				X							
5	seal off the vacuum								X								X		X								X				
6	chuck adapter plate								X								X	X	X							X					
2.11	chuck pedestal height at rest							X				X																			
7.2	chuck bellows											X																X			
7.3	electronics to move the shaft																												X		
7.4	chuck motion control software															X														X	

Figure 4-11: DSM at the Fifth Level of Decomposition (one level higher than the leaf level)

DP ID	DP	3.1	4	8	9	1	2	3.2.1	3.2.2	3.2.3.1	3.2.3.2	3.3	5	6	7
3.1	process recipe design														
4	matching network														
8	wafer robot arm and its motion path														
9	chuck body														
1	wafer transfer plate assembly			X			X							X	
2	electrocstatic clamping device	X		X	X	X				X		X	X	X	
3.2.1	BSG channel design						X		X	X					
3.2.2	BSG csubstance selection	X								X					
3.2.3.1	BSG net flowrate	X					X	X	X						
3.2.3.2	BSG gas flow tubes								X	X		X			X
3.3	chuck cooling system	X					X	X			X				X
5	seal off the vacuum				X		X							X	
6	chuck adapter plate				X		X						X		
7	chuck movement mechanism					X	X								

Figure 4-12: DSM at the Fourth Level of Decomposition (two levels higher than the leaf level)

DP ID	DP	3.1	4	8	9	1	2	3.2.1	3.2.2	3.2.3	3.3	5	6	7
3.1	process recipe design													
4	matching network													
8	wafer robot arm and its motion path													
9	chuck body													
1	wafer transfer plate assembly			X			X						X	
2	electrocstatic clamping device	X		X	X	X				X	X	X	X	
3.2.1	BSG channel design						X		X	X				
3.2.2	BSG csubstance selection	X								X				
3.2.3	BSG pressure	X					X	X	X		X			X
3.3	chuck cooling system	X					X	X		X				X
5	seal off the vacuum				X		X						X	
6	chuck adapter plate				X		X					X		
7	chuck movement mechanism					X	X							

Figure 4-13: DSM at the Third Level of Decomposition (three levels higher than the leaf level)

DP ID	DP	3.1	4	8	9	1	2	3.2.1	3.2.2	3.2.3	3.3	5	6	7
3.1	process recipe design													
4	matching network													
8	wafer robot arm and its motion path													
9	chuck body													
1	wafer transfer plate assembly			X			X						X	
2	electrocstatic clamping device	X		X	X	X				X	X	X	X	
3.2.1	BSG channel design						X		X	X				
3.2.2	BSG csubstance selection	X								X				
3.2.3	BSG pressure	X					X	X	X		X			X
3.3	chuck cooling system	X					X	X		X				X
5	seal off the vacuum				X		X						X	
6	chuck adapter plate				X		X					X		
7	chuck movement mechanism					X	X							

Figure 4-14: DSM at the Second Level of Decomposition (four levels higher than the leaf level)

DP ID	DP	3.1	4	8	9	1	2	3.2	3.3	5	6	7
3.1	process resipe design											
4	matching network											
8	wafer loading robot arm and its motion path											
9	chuck body											
1	wafer transfer plate assembly			X			X				X	
2	electrostatic clamping device	X		X	X	X		X	X	X	X	
3.2	EBSG	X					X		X			X
3.3	chuck body cooling system	X					X	X				X
5	seal off the vacuum				X		X				X	
6	chuck adapter plate				X		X			X		
7	chuck movement mechanism					X	X					

Figure 4-15: DSM at the First Level of Decomposition (the highest Level in the tree)

4.3.5 The Validity of the DSM Obtained from Matrix Transformation Method

4.3.5.1 Reviews from the CVC Engineering Experts

In order to verify the correctness of the DSM constructed from DM, the DSM was presented to 5 technical experts at CVC. These experts included one person from each of the functional groups that were preparing the integration process of the ESC—the advanced technology department, the mechanical engineering group, the electrical engineering group, the software and controls group, and the systems engineering group. Each expert reviewed the interactions in the DSM and agreed that most of the interactions captured were correct and reasonable. Only very few modifications were proposed. However, these proposed modifications were caused by the missing information during the construction of the DM, not the technique of converting DM to DSM. In the end, all of the proposed modification on the DSM could be correctly incorporated into the original DM. Therefore the case study demonstrated that the matrix conversion method gives us a valid DSM.

A better validation of the prediction DSM would be to compare it with what actually happened later in the ESC system integration project. However, as stated earlier, the company CVC was going through a major merger process, and the personnel was changing rapidly. By the end of this case study, most of the CVC employees on this project left the company. A new project manager was assigned to continue this project. Many of the initiatives in the company changed, and it became difficult to follow up with the ESC integration project after the author left the company in August. Therefore, the DSM obtained from the DM was not validated against what happened later on. Yet, in one occasion, the new project manager told the author about a design problem that occurred during their ESC integration work. The author went back to the prediction DSM and found the iteration was predicted in the DSM.

Nevertheless, although the DSM was not verified in a strict sense, all CVC engineers who were assigned to this project agreed with the DSM obtained from the DM. If the people on the team had not left the company because of the merger, they would have used the DSM to

start planning their integration effort. For a matrix transformation technique that has never tried out before, this case study gave an encouraging result.

4.3.5.2 DSM System Interaction Density

The DSM system interaction density is an idea started by my advisor Dr. D. E. Whitney at MIT. Dr. Whitney measured all of the past DSM's made by different researchers and calculated the interaction density:

System Interaction Density = Total Number of Off-diagonal Marks / Total Number of Rows

He observed that all of the past DSM's had an interaction density ratio of about 6, whether it was a small product or a large product. Therefore, the interaction density ratio of 6 has been a hypothesis to examine whether a DSM had captured enough information about system interactions. In this case study, the interaction density ratio is also used to test whether the DSM's built from requirements and from expert experiences had the same ratio.

In this case study, the system interaction density ratio is:

System Interaction Density = 146 / 52 = 2.8

This value is much lower than the typical number 6. There may be many possible causes for this low value. For instance, the product may be relatively simple. Or the system has been optimized to reduce the system couplings through many generations of products. Although the number 6 is a hypothesis, we may still speculate that the system interaction density ratio is so low in this case study because the DSM was built at early stage of the design process, and not so much was know yet about the system interactions. Therefore, the question about the validity of the DSM built from DM is again raised here. Unfortunately, due to the CVC's condition, we were unable to well follow up with what happened actually later on. It would have been very helpful to following up closely with what happened later on in the actual system integration. Would the experts know more about the system interactions and so the

interaction density increased? We can only hope a future research project would be fortunate enough to have a condition to answer this question.

4.3.6 The Choice of Output Variables

In Chapter 3, one of the unsolved questions for the matrix transformation method was how to select the output variables in the DM before transferring the DM into a DSM. From mathematical examples, we know the choice of output variables is unique for system elements (DP's in a DM) that are unrelated or in sequence. The choice of the output variables is the diagonal elements. The choice of output variables is not unique for system elements that are in a coupling.

The DSM in Figure 4-8 was obtained by choosing all diagonal elements (both in and out of iteration blocks) as the output variables. The resulting DSM was meaningful and useful for CVC engineers. Therefore, the diagonal elements must be a correct choice for output variables. To test the result of choosing non-diagonal elements in the iteration blocks as output variables, two different approaches were taken. The first approach—the Logical Approach--was to create the DSM by choosing non-diagonal elements in the iteration blocks, then see if the resulting DSM gives a logical design process. The second approach—the Mathematical Approach--was to assign sensitivity values in the DM, and see whether choosing non-diagonal elements make the system iteration diverge or converge faster. Both approaches are described below.

4.3.6.1 The Logical Approach

Take an iteration block from the ESC DSM as an example. The FR's and DP's involved are:

FR2.3.2 = Reduce clamping voltage
FR2.3.3 = Provide on/off and magnitude control for the voltage
FR2.3.5 = Prevent the chamber RF power from affecting the electric network

Reducing the dielectric layer's thickness in the chuck can reduce the voltage needed to clamp wafers. An electric circuit can be designed to control the voltage magnitude that is applied to the wafer. A DC choke circuit can be designed to prevent the RF power from affecting the electric network. Therefore, the corresponding DP's for the above three FR's are:

DP2.3.2 = Chuck dielectric layer thickness
DP2.3.3 = Electric circuit for the voltage control
DP2.3.5 = DC choke circuit

In addition, the above three DP's also have side effects on other FR's. Part of the voltage used to clamp the wafers can be dissipated in the circuit (DP2.3.3). Therefore, DP2.3.3 has side effect on FR2.3.2. The chuck dielectric layer thickness determines how much voltage is needed to clamp the wafer and hence influences the magnitude to be controlled in the circuit. Therefore, DP 2.3.2 affects FR2.3.3. In addition, the DC choke circuit is designed to filter out the RF power so that the magnitude control on the voltage can be stable. The DC choke circuit design must know the voltage to be controlled. The voltage regulation circuit must also know the capability of the choke circuit. Therefore, FR/DP 2.3.3 and FR/DP 2.3.5 affect each other. Using the Axiomatic Design method to construct the Design Matrix, we get:

	DP2.3.2	**DP2.3.3**	**DP2.3.5**
FR2.3.2	X	X	
FR2.3.3	X	X	X
FR2.3.5		X	X

This is a coupled design, and the choice of output variable is not unique. The first choice we can try is to choose the diagonal elements as output variables:

	DP2.3.2	**DP2.3.3**	**DP2.3.5**
FR2.3.2	(X)	X	
FR2.3.3	X	(X)	X
FR2.3.5		X	(X)

By choosing these output variables, we are saying:

DP2.3.2 = f (FR2.3.2, DP2.3.3)

DP2.3.3 = f (FR2.3.3, DP2.3.2, DP2.3.5)

DP2.3.5 = f (FR2.3.5, DP2.3.3)

In other words,

- Given the information about DP2.3.3 (Electric circuit for the voltage control), design DP2.3.2 (Chuck dielectric layer thickness) to fulfill the requirement FR2.3.2 (Reduce clamping voltage).
- Given the information about DP 2.3.2 (Chuck dielectric layer thickness) and DP2.3.5 (DC choke circuit), design DP2.3.3 (Electric circuit for the voltage control) to fulfill the requirement FR2.3.3 (Provide on/off and magnitude control for the voltage).
- Given the information about DP2.3.3 (Electric circuit for the voltage control), design DP2.3.5 (DC choke circuit) to fulfill the requirement FR2.3.5 (Prevent the chamber RF power from affecting the electric network).

The above interpretation of the design process as a consequence of choosing the diagonal elements as output variables makes perfect sense. Hence, the resulting DSM below represents a design process that is perfectly executable.

	DP2.3.2	**DP2.3.3**	**DP2.3.5**
DP2.3.2	(X)	X	
DP2.3.3	X	(X)	X
DP2.3.5		X	(X)

We may also choose non-diagonal elements as output variables in this DM. Below is one of the possible choices:

	DP2.3.2	**DP2.3.3**	**DP2.3.5**
FR2.3.2	(X)	X	
FR2.3.3	X	X	(X)
FR2.3.5		(X)	X

Therefore, we get:

DP2.3.2 = f (FR2.3.2, DP2.3.3)

DP2.3.5 = f (FR2.3.3, DP2.3.2, DP2.3.3)

DP2.3.3 = f (FR2.3.5, DP2.3.5)

In other words,

- Given the information about DP2.3.3 (Electric circuit for the voltage Control), design DP2.3.2 (Chuck dielectric layer thickness) to fulfill the requirement FR2.3.2 (Reduce clamping voltage).
- Given the information about DP2.3.2 (Chuck dielectric layer thickness), and DP2.3.3 (Electric Circuit for the Voltage Control), design DP2.3.5 (DC choke circuit) to fulfill the requirement FR2.3.3 (Provide on/off and magnitude control for the voltage).
- Given the information about DP2.3.5 (DC choke circuit), design DP2.3.3 (Electric circuit for the voltage Control) to fulfill the requirement FR2.3.5 (Prevent the chamber RF power from affecting the electric network).

Among the above three bullet points, the first one sounds very reasonable, because the choice of output variable in the first row is still the diagonal element. The second and third bullet points do not sound correct. Following these design procedures, both DP2.3.3 and DP2.3.5 will be designed for their side effects rather than the main FR they are designed for at first place. As a result, the DSM from this choice of output variables does not represent a logical design process:

	DP2.3.2	**DP2.3.3**	**DP2.3.5**
DP2.3.2	X	X	
DP2.3.3		X	X
DP2.3.5	X	X	X

In summary, choosing non-diagonal elements as the output variables leads to a non-executable design process. It is like designing a component not for its main purpose, but rather for its side effects. Choosing diagonal elements as the output variables gives us a

feasible design process through the resulting DSM. Therefore, from a logic point of view, the diagonal elements are the correct choice for output variables in a DM.

4.3.6.2 The Mathematical Approach

In this approach, our goal is to see the effect of output variable choice on the convergence of system iterations. First, we can assign sensitivity values to each entry in the Design Matrix. These sensitivity values indicate one unit of change in the DP would cause so much percentage of change in order to achieve the FR value.

We can again use the same example from the last section to demonstrate this point.

	DP2.3.2	**DP2.3.3**	**DP2.3.5**
FR2.3.2	0.75	0.5	0
FR2.3.3	0.5	0.75	0.2
FR2.3.5	0	0.2	0.9

Shawn Chen, the advanced research and development engineer for the ESC project, gave the values in the above DM. The values in each row or column do not have to add up to one because they are sensitivity values for each individual FR-DP relationship. These values can be interpreted as:

$\Delta FR2.3.2^* = 0.75 * \Delta DP2.3.2^* + 0.5 * \Delta DP2.3.3^*$

$\Delta FR2.3.3^* = 0.5* \Delta DP2.3.2^* + 0.75 * \Delta DP2.3.3^* + 0.2* \Delta DP2.3.5^*$

$\Delta FR2.3.5^* = 0.2 * \Delta DP2.3.3^* + 0.9 * \Delta DP2.3.5^*$Where ΔFR^* and ΔDP^* means a unit of change from the nominal values of FR and DP.

Also note the diagonal elements in the DM should always have the highest value in a row, because they should be the major contributors to the corresponding FR's. This observation is also a result of following the Axiomatic Design guidelines to construct the DM.

From the above DM, if we select the diagonal elements as the output variables, we get:

	DP2.3.2	**DP2.3.3**	**DP2.3.5**
FR2.3.2	(0.75)	0.5	0
FR2.3.3	0.5	(0.75)	0.2
FR2.3.5	0	0.2	(0.9)

Therefore,

$\Delta DP2.3.2^* = (1/0.75) * \Delta FR2.3.2^* - (0.5/0.75) * \Delta DP2.3.3^*$

$\Delta DP2.3.3^* = (1/0.75) * \Delta FR2.3.3^* - (0.5/0.75) * \Delta DP2.3.2^* - (0.2/0.75) * \Delta DP2.3.5^*$

$\Delta DP2.3.5^* = (1/0.9) * \Delta FR2.3.5^* - (0.2/0.9) * \Delta DP2.3.3^*$

The corresponding DSM is then:

	DP2.3.2	**DP2.3.3**	**DP2.3.5**
DP2.3.2	0	0.5/0.75	0
DP2.3.3	0.5/0.75	0	0.2/0.75
DP2.3.5	0	0.2/0.9	0

According to Strang (1986), and Smith and Eppinger (1997), when the DSM interactions contain the sensitivity values, the Eigen values of the DSM tell the convergence of the iterations. When the absolute value of the largest Eigen values is less than 1, the iteration converges.

The absolute value of the largest Eigen value for the above DSM is 0.71. Therefore, choosing the diagonal elements as output variables makes the design iteration converge.

Use the same Design Matrix but choose a different set of output variables such as below. This choice of output variables is the same as the second choice of output variables in the logic proof section.

	DP2.3.2	DP2.3.3	DP2.3.5
FR2.3.2	(0.75)	0.5	0
FR2.3.3	0.5	0.75	(0.2)
FR2.3.5	0	(0.2)	0.9

The resulting DSM is:

	DP2.3.2	DP2.3.3	DP2.3.5
DP2.3.2	0	0.5/0.75	0
DP2.3.3	0	0	0.9/0.2
DP2.3.5	0.5/0.2	0.75/0.2	0

The absolute value of the largest Eigen value for this DSM is 3.864. Therefore, the iteration does not converge if we choose the non-diagonal elements as output variables.

Although the above example is only one special case, the same applies to all general cases in design when sensitivity values are assigned. In summary, because the diagonal elements always have higher sensitivity values for the corresponding FR, only choosing the diagonal elements as the output variables can make the iteration converge.

4.3.6.3 Summary for Both Approaches

In the above two sections, we have seen that only by choosing the diagonal elements in the Design Matrix, could we get a meaningful DSM to aid the design process planning. In addition, only the diagonal elements make the design iterations in the DSM converge.

Both conclusions originate from how we built the Design Matrix. The Axiomatic Design DM is built with the rule that the diagonal elements are the major design parameter for the corresponding FR. Therefore, only by choosing the diagonal elements, could we get a meaningful DSM to aid the planning of the design process.

4.3.6.4 Implication on the Conversion between DM and DSM

The observation on the choice of output variables has two important implications on the conversion between DM and DSM:

1. The interactions seen in the DM are the same interactions that will be seen in the DSM. In other words, except for the differences in the row headings, DM and its DSM should have the same look.

2. When an interaction in the system between two DP's is identified, that interaction can be reflected directly in the DSM. Consequently, a mark at the same position will appear in the DM. Therefore, the matrix transformation between DM and DSM is a two-way process. This implication enables us to capture and document the emergent properties in the system using DSM, and reflect the impact of emergent properties quickly back to the desired functions in the DM. This capability of the matrix transformation method remedies the DM's incapability of dealing with emergent properties of the systems.

4.3.7 Not All Design Requirements are Decomposed

Not all design requirements in the CVC case study are decomposed in the Design Matrix. Several specifications cannot be decomposed down to specific subsystem(s) in the product. Below are some examples of such requirements:

- The micro-contamination delta level is less than XX particles of size YYY and larger for a ZZZ substrate surface.
- Has a life time longer than XX years based on YY shifts per day, ZZ days per week operation at KK substrate per hour.
- Mean Time between Cleaning is greater than XX week based on YY days per week and ZZ hours per day.

Note the values in these requirements are replaced by symbols for proprietary reasons. The CVC quality-engineering department developed the above specifications. The Vice President of Quality at CVC and other engineers on the ESC project agreed that these specifications were usually not explicitly decomposed into the design.

These requirements fit into the concept of Constraints in Axiomatic Design theory. Axiomatic Design Constraints are system level performance measures that cannot be decomposed down to subsystem levels [Suh (2000), Tate (1999)]. Axiomatic Design believes that if the people who decompose the systems keep these constraints in mind, the effects of the constraints on the system interactions will show in Design Matrices. This is easy said hard to do. Interviews with the CVC design engineers reveals that they really cannot map those system level performance requirement into the Design Matrix. In their design, CVC engineers approached these requirements by trial-and-error. They tried to reuse components understood from the past design and hope their knowledge about past design will help to predict what happens in the new design. How well the un-decomposed design requirements are met is not understood until the prototype-testing phase. If the requirements are not met, CVC engineers will redesign the product based on what goes wrong in the testing.

Therefore, the Axiomatic Design DM can never take all of the inputs to a design. Consequently, the interactions in the DM are not the complete set of system interactions. Therefore, using the matrix transformation method to convert a DM into a DSM only gives us part of all system interactions. This observation is a part of the answer for research question Q1-c.

4.3.8 Managing System Level Knowledge at CVC

The existence of indecomposable requirements indicates that there exist system interactions to be learned later on in the design process. Therefore, it is important to have a knowledge management framework to capture the new learning and relate back to how requirements, ultimately customer needs, are fulfilled. The effort of managing system level knowledge

started at the beginning of the design process by constructing DM and converting it into DSM's can be continued throughout the entire product lifecycle. In the end, we will capture all the important learning about the system/product we design. Meanwhile, collecting the knowledge learned about the reliability of individual components also enable the statistical analysis of the system level performance measure. Through the accumulation of knowledge about the design and the statistical analysis, the testing length and the iteration back to design may be reduced in the future.

When this case study was carried out, CVC did not have a good history in documenting their design knowledge. Preliminary design logbooks were kept for each project, yet the information in the logbooks were ordered chronically. It was difficult to browse past design books to find answer to current design problems. Therefore, CVC heavily relied on the knowledge of the experts in the company.

Before the integration work of the ESC project started, the only documents were the system design specification, the preliminary design drawings, and the ESC technical feasibility report. In order to construct the DM and then transfer the DM into a DSM, many CVC engineering experts were consulted. Among all the interactions captured by the final DSM at the leaf level of the system decomposition (Figure 4-8), 2.9% of the interactions could be identified from documents. The rest 97.1% of the system interactions resided in experts' minds. Figure 4-16 below shows this situation.

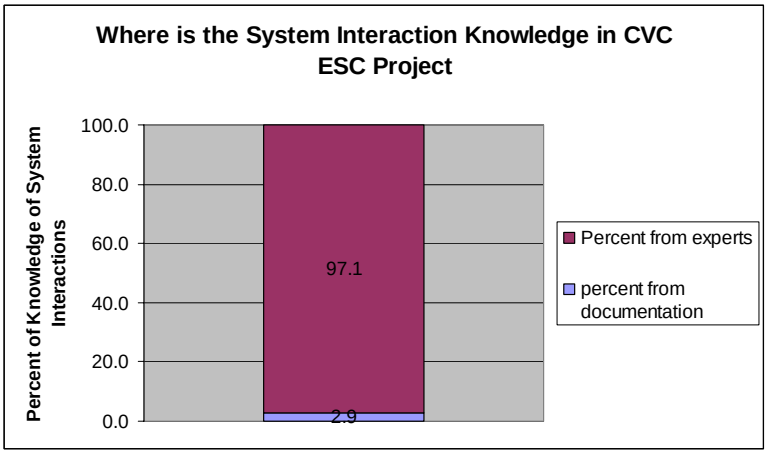


Figure 4-16: How well CVC Documents System Level Knowledge in the ESC Project

Therefore, the best source of system level knowledge at CVC was the engineering experts. Compare to Ford Motor Company (Figure 1-3), CVC is quite behind. A good knowledge management system should help CVC to better keep the organization learning within the company.

4.3.9 Results from Testing the Knowledge Management Framework

As mentioned earlier, Guru Prasanna—a MIT Master degree student—went to CVC to help test the knowledge management framework proposed in Chapter 3. Guru reviewed two project documents at CVC—the Metal Organic Chemical Vapor Deposition (MOCVD) project and the ESC integration project. The MOCVD project was a finished project. All of the design documents were already archived. The ESC project was an on-going project, and the design document was incomplete. Yet, the ESC case study had the DSM built by the author from the requirements, which captured some of the system interaction knowledge undocumented by CVC. In addition, Guru also reviewed the design documents for Ford throttle body design from the author's master's degree thesis. Ford throttle body design was a completed project, but the DSM's built for the author's Master's degree research capture some of the system interactions knowledge undocumented by Ford [Dong (1999)]. The goals for Guru's study were:

- Test the knowledge types proposed by the knowledge management framework on already completed project documents. Find out whether the knowledge types at least cover all of the knowledge being documented on paper.
- Use the framework to find out what types of knowledge are better documented at CVC and what are not. Discuss with the company means to improve in the area where documentation is lack.
- Compare the CVC's practice with that of Ford Motor Company.

Guru's study is documented in his thesis [Prasana (2000)]. The details will not be repeated here. The major findings from his thesis include:

- The categories of system level knowledge proposed in Chapter 3 were able to include all of the knowledge documented in all three cases—Ford throttle body design, CVC MOCVD project, and CVC ESC project. Therefore, these categories of system level knowledge may be sufficient for other cases too.
- The system level knowledge management framework proposed in Figure 3-1 was able to serve as a basis to measure what types of knowledge were well documented and what were not. This learning was helpful to point out to CVC what kind of improvements they need in the future for documentation. For instance, the "How" types of knowledge was not well documented in the ESC project, but was found with abundance in the DSM made for ESC project. Hence, DSM was recommended to CVC to document interaction types of knowledge.
- The knowledge management framework also served as basis for comparison among companies and projects. Guru compared Ford throttle body project with the CVC MOCVD and ESC projects using each categories of knowledge. Different projects do well in different categories. The framework provided more insights to the strengths and weaknesses of each company.

Nevertheless, all the comparisons Guru's thesis made were on the existing documents companies had and the DSM from the author's case studies. The testing on the knowledge management framework is far from conclusive for the following reasons:

- From Figure 3-1 and Figure 4-16, we know neither company did very well in documenting system level knowledge. Therefore, the design document only tested the knowledge management framework against a small amount of design knowledge.
- Although the DSM's in both the Ford and CVC case studies captured a lot of the system interaction information, we still lack the information on marketing report, manufacturing processes, and service and field support. This framework has been tested only in the FR and DP domains rather than in all four domains of the framework (see Figure 3-1).

Therefore, a lot more work is needed to further test and improve the knowledge management framework. However, the preliminary results from Prasanna (2000) are encouraging for future researches in this direction.

4.4 Progress Made Regarding the Research Questions

In this case study, the following observations were made about the research questions:

Q1-b. How to predict system interactions early? How to predict system interactions for new technology?

The CVC ESC case study tested the feasibility of the matrix transformation method in a real engineering project. The matrix transformation method produced a meaningful DSM from a DM. The CVC engineers accepted the partition results from the prediction DSM and planned to use it for the actual ESC system integration project.

Three benefits were observed in this case study regarding obtaining the DSM from the DM. First, the DSM was obtained before the ESC chuck integration project. The DSM analysis provided the CVC engineers insights about the system interfaces and where system iterations existed so that they could be prepared to address those system interface issues. Second, the DSM obtained from the DM predicted system interactions based on the system requirements. The DSM prescribed a design process that is different from what people think how things are always done. It provided refreshing ideas on how to improve the design based on the underlying structure of the system. Third, the DSM with team assignment associated with the DP's provided a way for CVC to manage the communication across the boundaries of geographically very dispersed design teams.

In addition, the answer to the choice of output variables was found from this case study. The selection of output variables must be the diagonal elements in the DM. Otherwise, the DSM interactions in the iteration blocks do not make sense in the design situation and do not converge. This answer is due to how Design Matrices are constructed based on the rules in Axiomatic Design.

The implication of this finding is that the DM and its corresponding DSM should look the same except for the headings of the rows. Therefore, when system emergent properties are discovered later on in the design, the relevant system interactions can be captured by the DSM and then reflected in the DM by putting a mark at the same location in the DM as that in the DSM. Therefore, we can remedy the DM's incapability of dealing with emergent properties of the system.

Unfortunately, due to the organization change at CVC, most of the people who started this ESC project left the company. The system interaction predictions made by the DSM was not verified during the actual integration process. The goal of the next case study is to set up an experiment where the prediction can actually be verified.

In summary, the results obtained from this case study about the matrix transformation method are very encouraging. The next case study shall focus on the following three aspects:

1. Is this matrix transformation method transferable to another product and another company?
2. How well does the prediction DSM matches with what the actual system interactions are?

Q1-c. If we can predict system interactions, how complete is the prediction?

This case study revealed that some requirements could not be easily decomposed at early phase of the design. These requirements measured system emergent properties. The system interactions that contributed to these requirements could not be predicted at early phase of the design. We must wait until prototyping and testing to understand the system tradeoffs that are important to meeting these requirements. Therefore, the interactions predicted by DSM transferred from the Design Matrix can never be the complete set of system interactions.

In addition, the system interaction density ratio of the resulting DSM was very low for the DSM at the lowest level of the system hierarchy. Although a valid DSM must have about 6

marks per row is a still a hypothesis, a low interaction density ratio gives a warning on the completeness of the interactions the DSM has captured.

However, the decision-making at early phase of the design process involves risks and uncertainties anyway. We may not need a prediction that is complete. If the prediction can reveal the essential information about the system, that may just be enough for project planning. Unfortunately again, the merger with VEECO interrupted the ESC project. We need another case study to find out whether this incomplete prediction is useful.

Q2-b. Is there a better way to capture, store, and represent system level knowledge?
The case study revealed that CVC lacks documentation for the system level knowledge. It heavily relied on expert engineers to carry the learning from the beginning to the end of the project and from projects to projects. The DM-to-DSM matrix transformation method provided a way to start documenting the system level knowledge from early on in the design process. When the ESC project was assigned to a new team after the merger, the DSM was passed on as part of the design document to the new project team so that they could catch up with what had been understood about system interactions. Therefore, the matrix transformation method was a better way to capture, store, and represent system level knowledge at CVC. However, this thesis cannot generalize this observation for all other companies. More case studies in different industries are needed.

In addition, Guru Prasanna [(2000)] tested the system level knowledge management framework using CVC's documents on MOCVD project and the ESC project, and Ford throttle design documents. Guru also reviewed the DSM's built by the author for Ford throttle body and CVC ESC project. Guru found the categories of system level knowledge proposed by this framework captured the documented knowledge in all three projects. Guru also found the framework was an effective way to identify what types of knowledge were recorded better/worse in a company, so that the company could focus the improvement effort on target. However, Guru's research focused on documented design knowledge in companies. His study did not look at all five domains of the design in the proposed

framework (see Figure 3-1). Therefore, we still cannot make conclusive statement about the effectiveness of the framework in general.

In summary, the findings in this case study were very encouraging regarding this question, but the data observed were incomplete to make any conclusive statement regarding this question. However, the observations did not provide any negative examples either.

Q2-c. What are the best sources of information for predicting system interactions?

In this case, since CVC does not have much design documents, the best source for predicting system interactions were the design requirements document, and the engineering experts who knows how requirements were decomposed into subsystems. CVC may be a typical example for small companies, but not all companies. Therefore, more observations are needed from another case study.

Q2-d. How companies are doing with managing system level knowledge?

CVC does a poor job in managing system level knowledge. The percentage of system level knowledge documented in the ESC project was only 2.9% (Figure 4-16), even compare to Ford's data (Figure 1-3). CVC heavily relied on engineering experts to understand and manage the system level interactions. This observation again confirmed the need for a good system level knowledge management framework, in order to keep the system level knowledge in the organization.

Q2-e. How to encourage engineers to document system level knowledge?

The matrix transformation method enabled the CVC engineers to obtain a DSM from documenting the requirements decomposition and trace-ability in a DM. The resulting DM and DSM were excellent design documentation for future reference. Furthermore, the prediction DSM can be used to help project planning immediately. The prediction DSM reveals interactions that are driven by product requirements. The interactions revealed the underlying structure of the system, rather than based on people's subjective judgment.

The effect of documenting system level knowledge was not only long-term, but also immediate using the matrix transformation technique. Therefore, the matrix transformation technique provides incentives for engineers to document system level knowledge.

4.5 Summary

This chapter detailed the case study carried out at CVC. The project was the system integration for the Electro-static Chuck (ESC). CVC's objectives were:

- Use the matrix transformation method to obtain a DSM before the integration work was carried out, so that mistakes could be prevented, and they could complete the project in time.
- Discover how to better set up a communication channel among the various dispersed divisions all over the nation.

The objectives for this thesis research in the CVC case study were:

- Test the framework for managing system level knowledge—Guru Prasanna and Qi Dong(Q2-b)
- Discover the situation of documenting system level knowledge at CVC—Qi Dong (Q2-c, d)
- Test the matrix transformation method—Qi (Q1-b, c, Q2-e)

The matrix transformation method introduced in Chapter 3 was applied. A DM was generated capturing the requirements decomposition and how the design concept fulfilled the requirements. This DM was transformed into a DSM predicting system interfaces involved in the ESC integration. The CVC engineers agreed that the prediction DSM captured very meaningful and useful system interactions, proving the matrix transformation method was feasible for real engineering cases.

For CVC, this case study fulfilled their objectives:

- The prediction DSM made from DM was partitioned. The results of partitioning identified the major system iteration area, and helped the project managers to plan their integration effort better.
- As a geographically very dispersed company, the team assignments indicated in the DSM helped to facilitate the communication across the organization boundaries.
- The current situation of system level knowledge management at CVC was studied. CVC was very behind. The knowledge management framework proposed in Chapter 3 was suggested to CVC to improve their current practice.

This case study helped us to understand the research questions better. We found partial answers to some of the questions, and raised more questions for further research. Regarding research questions Q1b:

- The matrix transformation method introduced in Chapter 3 was feasible in real engineering projects. Therefore, it enabled us to predict system interactions before the actual system integration of the ESC project started.
- The DSM analysis provided the CVC engineers insights about the system interfaces and where system iterations existed. CVC engineers could be prepared to address those system interface issues and avoid unnecessary rework.
- The prediction DSM prescribed a design process that is different from what people think how things are always done. It provided refreshing ideas on how to improve the design based on the underlying structure of the system.
- The DSM with team assignment associated with the DP's provided a way for CVC to manage the communication across the boundaries of geographically very dispersed design teams.
- The case study showed us the diagonal elements in the DM are the correct choices of output variables in the matrix transformation method.
- From the choice of output variables, we can conclude that the DM and its corresponding DSM look the same except for the row headings. System emergent properties can be

captured by the DSM and then reflected in the DM. This remedies the DM's lack of capability in dealing with system emergent properties.

- The correctness of the prediction DSM was not verified by what actually happened later in the ESC project due to the organization changes from CVC's merger. Thus, the following questions were posed:
    - Is this matrix transformation method transferable to another product and another company?
    - How well does the prediction DSM matches with what the actual system interactions are?

Regarding Question Q1-c, the CVC case study showed that not all requirements could be decomposed using the Axiomatic Design's Design Matrix. Therefore, the DSM converted from DM does not provide a complete view of the system interactions. The system interactions associated with emergent properties are to be learned later. This discovery brings the question of how good is the prediction in the DSM transformed from DM.

Regarding Question Q2-b, it was found that the matrix transformation method is a good way to let people start documenting the requirements decomposition and flow-down structure from early on in the project. Guru tested the knowledge management framework proposed in Chapter 3 on CVC's documented design knowledge. No adverse examples were found, and the knowledge categories worked for CVC's design documents. Yet, the conclusion based on one case study is not conclusive. More research cases are needed.

Regarding Question Q2-c, the requirements document and the experienced engineers were the main sources of information for predicting system interactions. Regarding Question Q2-d, CVC was not good at managing system level knowledge. Less than 3 percent of the system level knowledge in the ESC project was documented. Regarding Q2-e, the matrix transforming method not only is a way to document system level knowledge, but also produce a prediction DSM. Analyzing the system structure using DSM techniques can provide immediate insights on the system of the product and hence help the project planning.

Therefore, documentation is no longer a burden, but rather a tool to improve the current design process.

In summary, the CVC case study provided many encouraging results and observations, which were enough reason to continue the investigation on these research questions using another case study in another industry.

5 Johnson and Johnson Case Study

5.1 The Research Setting

5.1.1 About Johnson and Johnson Ortho-Clinical Diagnostics (JNJ OCD)

Ortho-clinical Diagnostics (OCD) is a company within Johnson and Johnson (JNJ). It produces high-value diagnostic products and services for the global health care community. The major products of OCD includes:

Transfusion Medicine

- Blood Screening – development and commercialization of instrument systems and reagents that screen blood for AIDS and Hepatitis, aimed at ensuring the safety of the world's blood supply.
- Immunohematology – Ortho-Clinical Diagnostics is the worldwide leader in the marketing and development of instrumentation and reagent systems that enable blood typing, aimed at ensuring patient-donor compatibility in blood transfusions.

Clinical Laboratories

- Clinical Chemistry – patented dry-slide technology and systems for use in stat and random access in-vitro diagnostic testing. The Company offers a broad menu covering basic metabolites, classical chemistries, special chemistries, proteins, toxicology and therapeutic drug monitoring tests.
- Immunodiagnostics – enhanced chemiluminescence technology and systems offering immunoassay testing capabilities across menu categories of thyroid function, reproductive endocrinology, cardiology, anemia, metabolism, oncology and infectious diseases.

More information about OCD and its products is at the website: http://www.orthoclinical.com.

5.1.2 Case Study Description

5.1.2.1 The Product and Process

This case study was conducted in the OASIS program at Johnson and Johnson Ortho-Clinical Diagnostics (JNJ OCD). The OASIS program develops the OASIS analyzer, which is an automated clinical chemistry system combining both the Thin Film technology and the Wet Chemistry technology within the same unit. Figure 5-1 is a picture of the OASIS model in the showroom. The OASIS program started in February 2000. The product is planned to be ready for manufacturing in February 2003. The development of the analyzer involves engineers and scientists in various technical disciplines, including Marketing, Mechanical Engineering, Electrical engineering, Computer Science and Software Engineering, Clinical Assay Chemistry, Quality Assurance, manufacturing, etc. At the peak of the development process, the core group contains about 120 people.

Figure 5-1: OASIS Analyzer Model

The development of the OASIS analyzer is an architecture innovation (see Figure 2-7) for JNJ OCD. OCD owned the thin film technology for many years. Many of their existing chemistry analyzers use this technology. Wet chemistry technology is new for OCD but also has existed in the market for quite a few years. OCD is purchasing the wet chemistry technology from a vendor company. Therefore, the core technology is not new for OASIS. Yet, integrating the two technologies into one analyzer is a new architecture problem for OASIS. Therefore, the OASIS development is an architectural innovation.

The author has been involved with the OASIS program since summer 2000. During summer 2000, the OASIS program was going through the concept design and feasibility analysis phase. At the same time, the company was trying to adapt to new systems design methods such as Six-Sigma and Robust Design. A lot of uncertainties were involved in the program. Although it would have been an excellent time to test the author's matrix conversion method, the author was unable to find solid support and sponsor for this research in the OASIS program.

Beginning January 2001, the OASIS program moved into the detail design phase. Under the effort of the systems engineering group, requirements were documented and kept up to date. When the author returned to JNJ OCD in January 2001, OCD managers were finally able to make a solid connection between this thesis research work and the OASIS program.

Although the program had moved into detailed design in year 2001, the systems integration team lead by Mark Raymond would still like to learn what the system interfaces were and how they could prevent serious problems and rework that might happen during system integration and verification. The planned system integration phase would start in November of 2001. Therefore, applying the author's matrix conversion method in the summer of 2001 would help forecast the system interfaces and improve the OASIS system integration work. Although the author missed the opportunity to use this method in the concept selection phase, this method would still be able to help predicting system interfaces and have impact on a real product program.

5.1.2.2 Case Study Scope

Systems can be viewed in different layers of hierarchy and different amount of details. The scope of this case study was chosen under two conditions. First, the amount of information should be enough to provide insights of the system, but not too much for a three-month-one-person summer research project. Second, the research should try to mesh with the current systems engineering effort in the OASIS program so that the results can be beneficial to JNJ OCD, as well as can be compared to the reality.

This case study limits its scope to predicting only the interactions among the subsystems of the OASIS analyzer. Interactions among the components within subsystems were not captured. Figure 5-2 shows the subsystems in OASIS as well as their abbreviations. This architecture is adopted from the OASIS architecture document written by Dee (2001a). As stated earlier, this research missed the opportunity to have influence in the concept selection phase. Therefore, this case study would have to stick with the chosen design concept and predict the system interfaces within this particular architecture. If this method were applied during the concept selection phase, the construction of the Design Matrix would have had influence on the choice of the design concept.

Figure 5-3 shows where some of the major subsystems are in the analyzer. Due to the limited detail of this schematic, some of the small metering systems are not shown. In addition, the *Delivery* types of subsystems listed in Figure 5-2 are about the manufacturing and supply chain design. They are not hardware designs and cannot be shown on this schematic.

This choice of system scope was suitable for this case study because it fulfilled the two requirements mentioned earlier. First, there were total 32 subsystems involved, which was a manageable number to build a DM and a DSM for the research purpose. Second, the OASIS systems engineering and systems integration group were only interested in the system level interactions. These two groups were the direct sponsors of this research. Choosing to look at the interactions among subsystems aligned with the goal of the systems engineering group at OASIS.

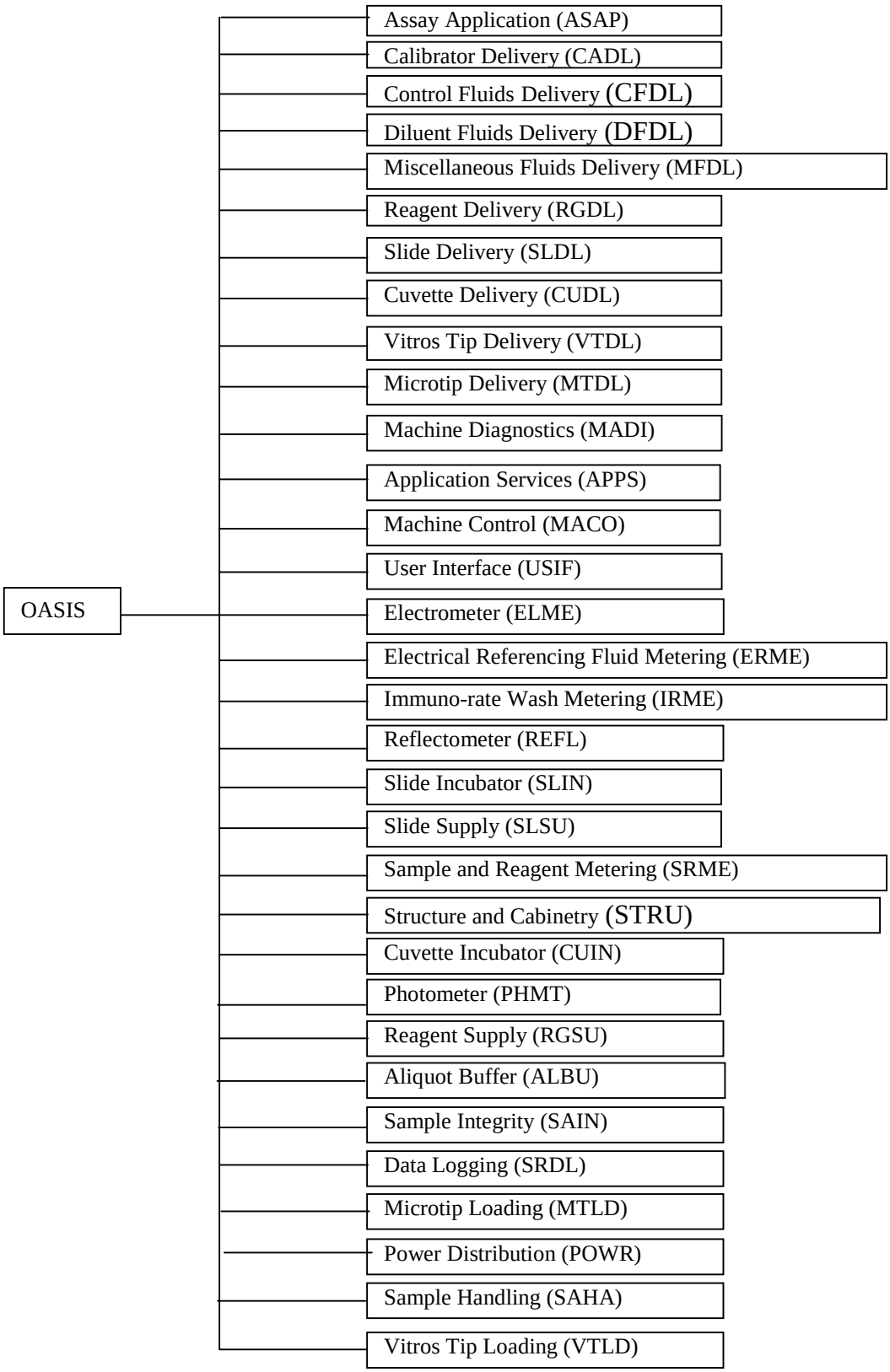


Figure 5-2: OASIS Subsystems

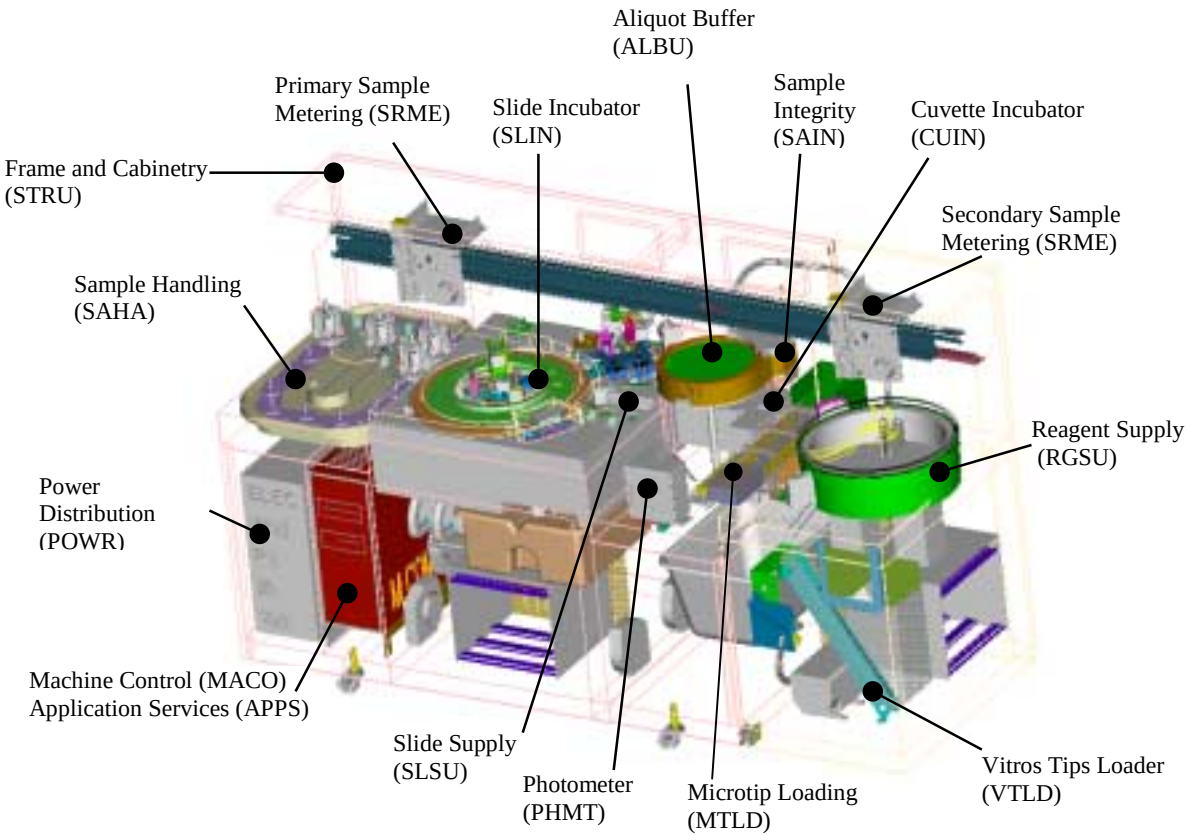


Figure 5-3: Location of Major Subsystems

5.1.2.3 Case Study Objectives

5.1.2.3.1 Research Objectives

Learning from the unanswered research questions in CVC case study, the main objectives of this case study are the following. The parentheses at the end of each objective indicate which research question was related to which objective.

1. Use the OASIS Product Requirements Document (PRD), Subsystem Requirements Document (SSRD), and other supporting documents to build a DSM capturing the

interactions among the subsystems using the DM-DSM matrix conversion method. Find out if the matrix conversion method is transferable to a different case study (Q1-b).

2. Investigate how closely the DSM derived from requirements predicts the reality by comparing it with the DSM produced by expert engineers and scientists using the traditional DSM building method (Q1-c).

3. Understand whether the matrix conversion method can be used for all types of requirements or just Functional Requirements. Observe what types of requirements drive the system interactions (Q1-c).

4. Observe the sources of information in identifying system interactions. How much system interaction knowledge is captured in documents and how much is in people's heads? Benchmark the result with the same data from Ford and CVC (Q2-c, d).

5.1.2.3.2 JNJ OCD's Objectives

The objectives of JNJ OCD in this case study include the follows:

1. Predict system interface problems that may happen during integration and verification.
2. Aid the system integration manager's work on planning and managing OASIS subsystem interfaces.

5.1.2.4 Chapter Outline

This chapter is long. The following list shows the relationship between each section in the rest of the chapter and the research objectives:

5.2 Data Gathering Process	JNJ Case Research Objective 1, Research Question Q1-b
5.3.1 How Realistic is the Prediction DSM from Requirements	JNJ Case Research Objective 2, Research Question Q1-b, c
5.3.2 The Requirements Decomposition Process vs. Various Types of Requirements	JNJ Case Research Objective 3, Research Question Q1-c
5.3.3The Sources of System Level Knowledge	JNJ Case Research Objective 4, Research Question Q2c, d

5.2 Data Gathering Process

This case study used six OCD system design documents to gather system interaction information. Some of the design documents could be used to construct DM, while the others could only be used to construct DSM. Figure 5-4 shows the use of each document. The requirements documents should have been used to construct DM like in the CVC case study. However, The dashed line linking requirements documents and DM indicates that the requirement documents were actually used to directly construct DSM in this case. The implication of doing so on the matrix transformation method will be explained in detail in the subsections here.

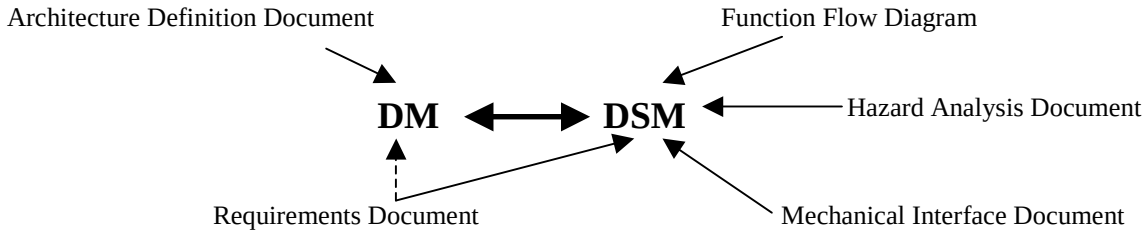


Figure 5-4: The Use of Each OCD System Engineering Design Documents

The reason to use not only requirement documents but also other system engineering documents was to compare whether the matrix transformation method can capture all of the system interactions documented in the existing JNJ OCD system engineering design documents. During the data gathering process, the system interactions captured by each of the documents in Figure 5-4 are marked out so that we can later on analyze the contribution of each type of document on identifying system interactions.

5.2.1 Building a Design Matrix Based on the Architecture Definition Document

The Architecture Definition Document was written by the system engineer Mike Dee [Dee (2001a)] in May 2001 before the detailed design phase started in the OASIS program. This document records the final decision of the product architecture (Figure 5-2) and the functionality of each subsystem in the analyzer.

According Axiomatic Design [Suh (2000)], the process of building a Design Matrix (DM) is the process of design concept selection. However, as stated earlier, this research case study missed the opportunity to construct a DM to influence the concept design of the OASIS analyzer. The DM built in this step is a mapping between the major functional requirements and the chosen design concept and architecture based on the OASIS Architecture Definition Document. Yet, all rules in the Axiomatic Design about building a DM were followed. After the initial DM was built, based on the author's Matrix conversion method, a DSM containing the interactions among the subsystems was generated from the DM.

Although the Architecture Definition Document enabled us to obtain a DM and a DSM, these two matrices were fairly sparse with few off-diagonal marks. The reason was that the Architecture Definition Document defined the major functions of each subsystem, but contained little information about the interfaces between subsystems. The resulting DSM from this document will be shown later in the results and discussion section (Figure 5-24).

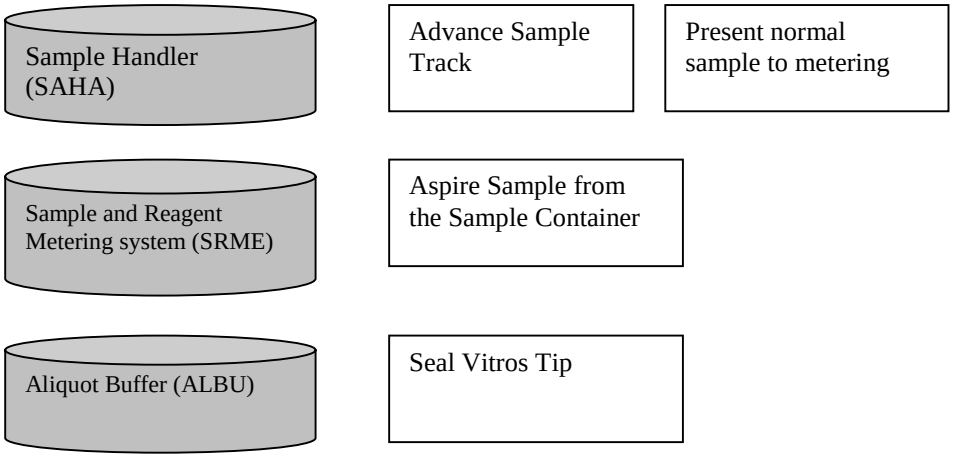
5.2.2 Identify System Interactions Using Function Flow Diagram

During the concept development phase, the system engineers at OCD constructed function flow diagrams to understand how functions are decomposed into and fulfilled by the subsystems in the analyzers. The function flow diagrams that OCD engineers built are also called Function Analysis System Technique (FAST) diagrams [VAI (1993), Wood and Otto (2001)]. The function flow diagrams were constructed before the product requirements of the OASIS analyzer were finalized, and were still continuously being build for subsystems during the period of this case study (summer 2001).

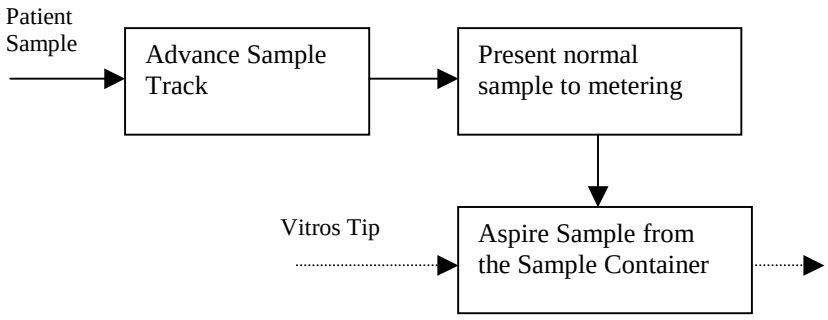
The first part of the OCD function flow diagram was a list of subsystems and the major functions they performed. The second part of the OCD function flow diagram is a flow diagram linking various functions in the list. Figure 5-5 shows a simplified example of how a DSM was built from the OASIS function flow diagram. The actual diagram was much larger.

Once the additional off-diagonal marks are discovered in this step, these marks can be easily reflected back to the DM at the same location in the matrix, based on the discussion on output variable selections in Chapter 4 (4.3.6.4 Implication on the Conversion between DM and DSM). The significance of constructing a DM before the DSM is to use the major functional requirements to identify the major design parameters that should be in the DSM. In the traditional DSM building process, the design experts come up with the system elements (the titles of rows and columns), which is more subjective than using the DM construction process to identify the system elements that are truly critical to the system interaction and fulfilling the requirements. However, as shown in this step, sometimes the DSM view is better at capturing off-diagonal marks in the matrices. Fortunately, the discovery of the output variable selection rules allows us to easily switch between the two views.

Part 1: list all subsystem functions



Part 2: draw function flow diagram



Part 3: Identify DSM Relationship

	SAHA	SRME	ALBU
SAHA	X	X	
SRME	X	X	X
ALBU		X	X

Figure 5-5: Building a DSM from Function Flow Diagram

5.2.3 Identify System Interactions Using Mechanical Interface Document

The third document that helped to identify more off-diagonal marks in the DM and DSM was the Mechanical Interfaces Document [Dee (2001b)]. This document records all of the spatial relationship among subsystems. The system engineers in the OASIS program constructed this document at the beginning of the detailed design phase. It is continuously revised as the engineers learn more about the spatial configuration between subsystems. The interfaces discovered in this documentation was recorded in the DSM and reflected back in the DM.

5.2.4 Identify Subsystem Interactions Driven by Product Requirements

Two types of requirement documents existed in the OASIS program. The OASIS Product Requirement Document (PRD) [Dee (2001c)] provides the complete list of the product-level requirements. PRD was written during the concept design phase by the system engineers (see Appendix A for examples). At the next level of the system, subsystem engineers responsible for each subsystem generated subsystem requirement documents (SSRD) at the beginning of the detailed design phase of each subsystem. Early on in the OASIS program, the PRD's and SSRD's were written separately by different groups of people without communicating to each other. With the effort of the OASIS system engineering team, engineers were educated about the importance of the requirement trace-ability. During the period of May to June 2001, the engineers at OCD started to record the links between the SSRD's and PRD. These links were recorded in the OCD requirement management software tool—RequisitPro.

The trace-ability links between the PRD and SSRD were important to this research, because these links helped to identify which subsystem(s) would be responsible to fulfill a product level requirement. If a product level requirement flows into more than one subsystem, then it is possible for these subsystems to interact with each other. The reason is that these subsystems may have to work together to fulfill the same requirement.

In this case study, a database was built to record which subsystem(s) each product level requirement flows into, instead of building a Design Matrix. The reason for doing this is

explained in Section 5.2.4.1. Then the author took the product requirements that flowed into multiple subsystems, and examined how the subsystems contributed to the product requirement. When subsystems each contributed partially to the fulfillment of the product, they were identified as having system level interactions. The interaction marks can then be entered in the DSM and subsequently in DM. Figure 5-6 depicts this process using a simple example.

1. Build the Database to Record PRD Decomposition

PRD #	Subsystems flow into
3	REFL
3	SAIN
3	MACO
3	APPS
3	SRME

PRD3: The OASIS analyzer shall be able to perform XXX sample integrity indices on spinal fluid, serum, and plasma samples within the performance levels stated in XXX document (XXX here replaces proprietary information).

2. Build the Resulting DSM:

The subsystems listed in the above database have to work together to deliver the sample integrity function and performance.

	REFL	SAIN	MACO	APPS	SRME
REFL	X	X	X	X	
SAIN	X	X	X	X	X
MACO	X	X	X	X	X
APPS	X	X	X	X	X
SRME		X	X	X	X

Note not all subsystems interact with each other although all of them contribute to PRD3. The knowledge of which subsystems should interact with each other cannot be found in the requirements flow down record in Step 1 (RequisitPro in the JNJ OCD case). The marks in the DSM are filled in by asking whether a DP interact with another DP to fulfill this particular requirement. For instance, Sample Integrity (SAIN) subsystem interacts with Reflectometer (REFL) to fulfill PRD3. Therefore a mark exists between the two in the DSM. Sample and Reagent Metering System (SRME) does not interact with Refelctometer (REFL) to fulfill PRD 3, therefore no mark is put in the DSM between the two DP.

Figure 5-6: Requirements Decomposition Process

The resulting DSM in Figure 5-6 can be easily converted back to the DM with all the marks remaining at the same place. The reason again was explained by the implications of the choice of output variables (see 4.3.6.4 Implication on the Conversion between DM and DSM).

However, by the end of June, not all SSRD's were finished and not all subsystem engineers were able to solidify the links between their SSRD and PRD. Due to the time constraint of this case study, the cut-off time for collecting the information from the RequisitPro was set to July 6th, 2001. Then the author consulted responsible system engineers for each subsystem to identify the PRD-SSRD links not recorded in RequisitPro by July 6th. The RequisitPro database may contain more documented links between requirements after July 6th.

5.2.4.1 Was the Matrix Transformation Method Used?

The procedure in Figure 5-6 does not follow the three steps of the matrix transformation presented in Chapter 3. Did you skip the design matrix? The answer is that we could have taken the three steps of matrix transformation, but we used a shortcut instead to directly obtain a DSM from requirements. Detailed explanations are as follows.

First, we built a partial DM from the architecture definition already. Therefore, we started with a DM. When the requirements document was used, each requirement could have two possible effects on the design matrix. The first effect is that the requirement became an additional FR in the DM. The second effect is that the requirement became a sub-FR of an existing FR.

Take the PRD3 in Figure 5-6 as an example. Assume we started with a DM that looked like:

	DP1	DP2
FR1	X	
FR2		X

(1)

The first possible scenario was that PRD3 became an additional FR.

	DP1	DP2	DP3
FR1	X		
FR2		X	
PRD3		X	X

The matrix transformation method thus would give:

	DP1	DP2	DP3
DP1	X		
DP2		X	
DP3		X	X

However, turning PRD3 into an additional FR requires an additional DP. In this case study, it was assumed that all the system level DP's (the major subsystems) were already fixed by the JNJ system engineers. The architecture definition document produced the DM that included all of the DP's. Therefore, the PRD's in this case study would not become FR's in the DM. This first possible scenario did not exist in this case study.

Then we are left with only one other possibility for the PRD's—turning into sub-FR's. Assume the PRD3 can be broken down into sub-FR's—FR1.1 and FR2.1. Since FR1.1 and FR2.1 together fulfilled PRD3, DP1.1 and DP2.1 would interact with each other because they had to fulfill the same requirement at system level—PRD3. Then:

	DP1	DP1.1	DP2	DP2.1
FR1	X		X	
FR1.1		X		X
FR2	X		X	
FR2.1		X		X

The resulting DSM is:

	DP1	DP1.1	DP2	DP2.1
DP1	X		X	
DP1.1		X		X
DP2	X		X	
DP2.1		X		X

Since we are only interested in the interactions among the subsystems in this case study, we can collapse the DSM into:

	DP1	DP2
DP1	X	X
DP2	X	X

(2)

Compare matrix (1) and (2), we can conclude that if we know a PRD affects multiple DP's, those DP's are most likely to have interactions with one another because they had to serve the same high-level requirement. Since we are only interested in the interactions among the subsystems not the sub-DP's within each subsystem, we can directly fill in the interactions among the subsystems in the DSM rather than going through the DM decomposition.

Therefore, the procedure in Figure 5-6 although did not follow the three steps of matrix transformation, does not contradict with the research method. It is merely a short cut of the matrix transformation method.

5.2.4.2 When is DM Necessary?

From Figure 5-6, it seems that we can build a DSM directly based on requirements without building a DM as the first step. Can DSM replace DM? The answer is yes for this case study, but not necessarily true for general cases. The reasons are discussed below.

The Design Matrix has two contributions when used in the matrix conversion method to predict system interactions. First, DM can help to identify DP's based on the functional requirements. These DP's then turn into the system elements (titles of rows and columns) in the DSM. Second, the zigzagging method can be used in the construction of a DM to identify lower system level DP's and FR's. Therefore, DSM's for various levels of the system hierarchy can be easily derived.

However, this study had its own special conditions. First, the system level DP's—the subsystems in the OASIS analyzer—were already identified and fixed. Hence the first use of

a DM was not necessary any more. Second, the goal of this case study was to identify the interactions among the subsystems only. The interactions among the components of the subsystems were not needed. Therefore, the DM's system decomposition capability was not used here. Consequently, after using the architecture definition to build an initial DM and converting it into a DSM, it was no longer necessary to work with the DM any more.

In summary, DM must be used when either of the following conditions exists:

1. We need to identify the DP's in the system.
2. We need to predict the system interfaces at multiple levels of the system hierarchy.

5.2.5 Identify Subsystem Interactions from Hazard Analysis and Mitigation Document

The last piece of information to help identify subsystem interactions was the Hazard Analysis and Mitigation document (HAMG). The HAMG document is created by expert engineers based on their knowledge of the existing analyzers in the field. This document is constructed before the OASIS concept development phase so that past lessons learned from the field can be applied to the new analyzer design. Below is an example of the document:

HAMG Tag	Text	Traced to Subsystem Requirement
HAMG 1	Design: Control sealer temperature below flash point of tip material	ALBU 8, 12

The problems learned from past analyzers are flown into the relevant subsystems in the current analyzer design, and take the form of subsystem requirements.

When a HAMG item flew into more than one subsystem (the above example only flew into one subsystem--ALBU), the relevant subsystems have potential to interaction. The author then judges how interactions between subsystems exist due to the particular HAMG item, and record the interactions in the DSM.

5.2.6 When to Use DM and When to Use DSM

The process of data gathering in this case study revealed that some design documents are suitable to be used to gather the interactions in the DSM, while some others can be used to construct the DM (Figure 5-4). Here is a summary of when to use DM and when to use DSM based on the observations made from this case study.

Use DM, and construct the DSM following the matrix conversion method when any of the following situations is true:

1. We need to identify the DP's in the system.
2. We need to predict the system interfaces at multiple levels of the system hierarchy.

Use DSM when any of the following situations is true:

1. We know the key design parameters in the system already. This situation could very much be true for a design that reuses existing components or subsystems. This situation also applies when the design makes only incremental changes to the existing product. Therefore, as shown in Figure 2-7, DSM is suitable for incremental design changes.
2. We are only interested in the interactions at one level of the system hierarchy. The knowledge about system decomposition is not necessary.
3. The available information concerns the interactions within a system rather than how they relate to requirements.
4. The system interaction concerns the emergent properties learned later in the design process.

Fortunately, from the discussion in chapter 4 regarding the selection of output variables (4.3.6.4 Implication on the Conversion between DM and DSM), we know the DM and DSM can be converted to and from each other. Therefore, whether we get the system interaction information from a DM or a DSM does not really matter. We can always switch to the other representation depending on the interest of discussion.

5.2.7 The DSM built by JNJ OCD Engineering Experts Using the Traditional DSM Construction Method

Under the lead of the system integration manager Mark Raymond, the expert engineers in the OASIS program built two DSM's to capture the interactions among major subsystems in the analyzer. These two DSM's were built using the traditional DSM construction method. Experts from various areas of the product development team sat together in meetings and built the DSM's through discussions. The first expert DSM was built in February 2001 and the second one was built at the beginning of August 2001. The expert DSM's contain three major sources of interactions:

- Functional interactions
- Spatial interactions
- Reliability interactions

When the first DSM was built in February, the subsystem architecture (Figure 5-2) was not finalized. Therefore the subsystem names were different in the February DSM. When the August DSM was constructed, for the convenience of the engineering experts, the same subsystem names as those in the February DSM were used.

Fortunately, the differences of the system elements between the experts' DSM and the official subsystems (Figure 5-2) are very small. The author converted the expert DSM's subsystems into ones using the current subsystem names. Table 5-1 lists the conversion process.

In Experts' DSM	Convert to Current Subsystem Names
Aliquot Buffer	Aliquot Buffer (ALBU)
Cardcage	Structure (STRU)
Cabinetry	Structure (STRU)
Circuit Boards	Power (POWR)
Cuvette	Cuvette Delivery (CUDL)
Cuvette Loader	Cuvette Loader (CULD)
Electrometer	Electrometer (ELME)
ERF Container	Miscellaneous Fluids Delivery (MFDL)
ERF Metering	ERF Metering (ERME)
Frame	Structure (STRU)
Harnessing	Power (POWR)
IWF Container	Miscellaneous Fluids Delivery (MFDL)
IWF Metering	Immuno-rate wash fluids metering (IRME)
Master Computer	Machine Controller (MACO)
Microtips	Microtip Delivery (MTDL)
Microtip Loader	Microtip Loader (MTLD)
Photometer	Photometer (PHMT)
Power Supply	Power (POWR)
Primary Metering	Sample and Reagent Metering (SRME)
Reagent Container	Reagent Delivery (RGDL)
Reagent Supply	Reagent Supply (RGSU)
Reflectometer	Reflectometer (REFL)
Slides	Slide Delivery (SLDL)
ERF Container	Miscellaneous Fluids Delivery (MFDL)
Sample Handler	Sample Handler (SAHA)
Sample Integrity	Sample Integrity (SAIN)
Secondary Metering	Sample and Reagent Metering (SRME)
Slide Incubator	Slide Incubator (SLIN)
Slide Supply	Slide Supply (SLSU)
Slide Transport	Slide Incubator (SLIN)
Tip Sealer	Aliquot Buffer (ALBU)
Vitros Tips	Vitros Tip Delivery (VTDL)
Vitors Tip Loader	Vitros Tip Loader (VTLD)
Measurement Channel	Cuvette Incubator (CUIN)

Table 5-1: Converting the Experts' System Elements to the Official Subsystem Names

The February DSM represents the experts' prediction at early stage of the detailed design process. August DSM represents the change in experts' knowledge about the system.

5.3 Results and Discussion

This section presents the results of this case study in three parts. First, the DSM built from the requirements is compared with the two experts' DSM's made in February and August

2001. How realistic it is to predict system interactions from requirements is discussed. The second part of the discussion focuses on how requirement decomposition is carried out for various types of requirements, and the strength and limitation of the author's matrix conversion method. Third, the DSM's obtained from various design documents (Figure 5-4) are presented and compared. The author will discuss the various sources of information for identifying system interactions.

5.3.1 How Realistic is the Prediction DSM from Requirements

5.3.1.1 Compare the DSM Constructed Using All Design Documents and the DSM Constructed from Only Requirements

A DSM can be constructed combining all the information regarding system interactions in the documents listed in Figure 5-4. The resulting DSM is shown in Figure 5-7.

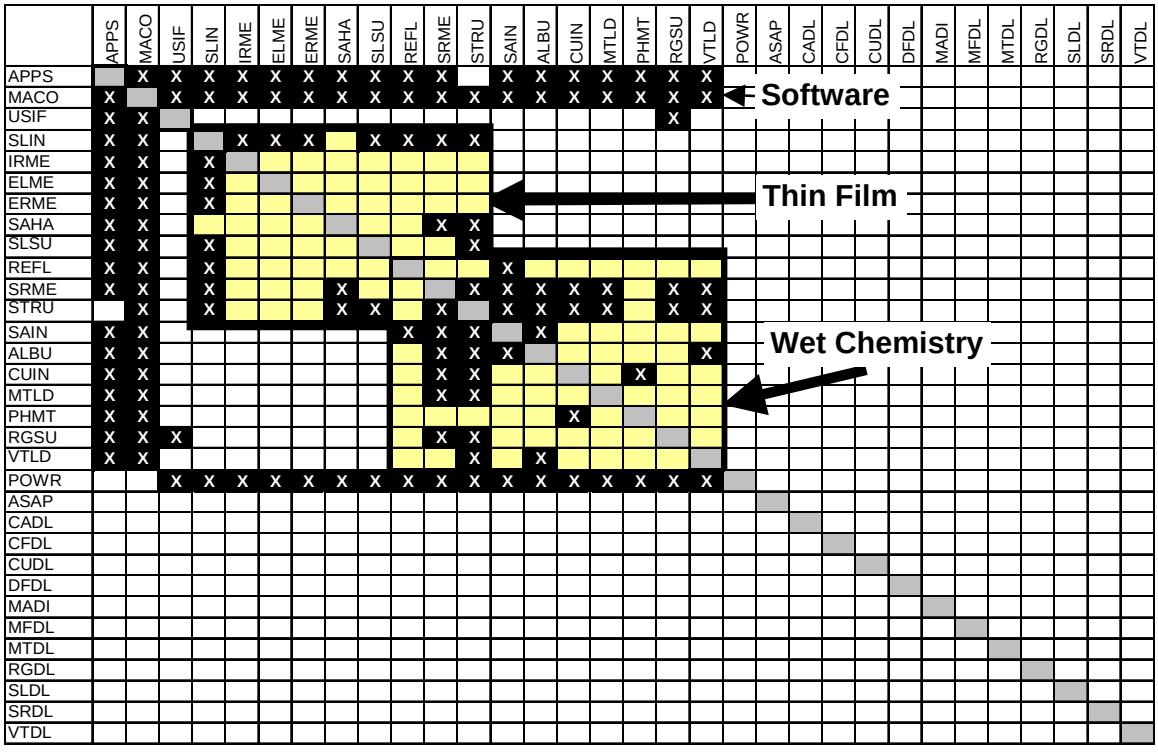


Figure 5-7: The DSM Constructed Using all Document Sources in Figure 5-4

The DSM partition algorithm identifies the system in Figure 5-7 to be coupled from "APPS" to "POWR". The author manually arranged the system elements in the iteration block based on the three major functions the OASIS analyzer had—the software, the thin film (dry) chemistry, and the wet chemistry. As shown in Figure 5-7, the Software (APPS, MACO, and USIF), the Structure (STRU), the Metering subsystem (SRME), the Power subsystem (POWR), and the interaction between the Reflectometer (REFL) and Sample Integrity (SAIN) tie the wet and the dry chemistry functions together.

The DSM in Figure 5-7 is constructed using information from all OASIS systems engineering documents. However, this case study is interested in the effectiveness of obtaining a DSM from requirements. Therefore, another DSM was built to include only the interactions captured by the requirements document (PRD and SSRD) and the architecture definition document. This DSM is shown in Figure 5-8.

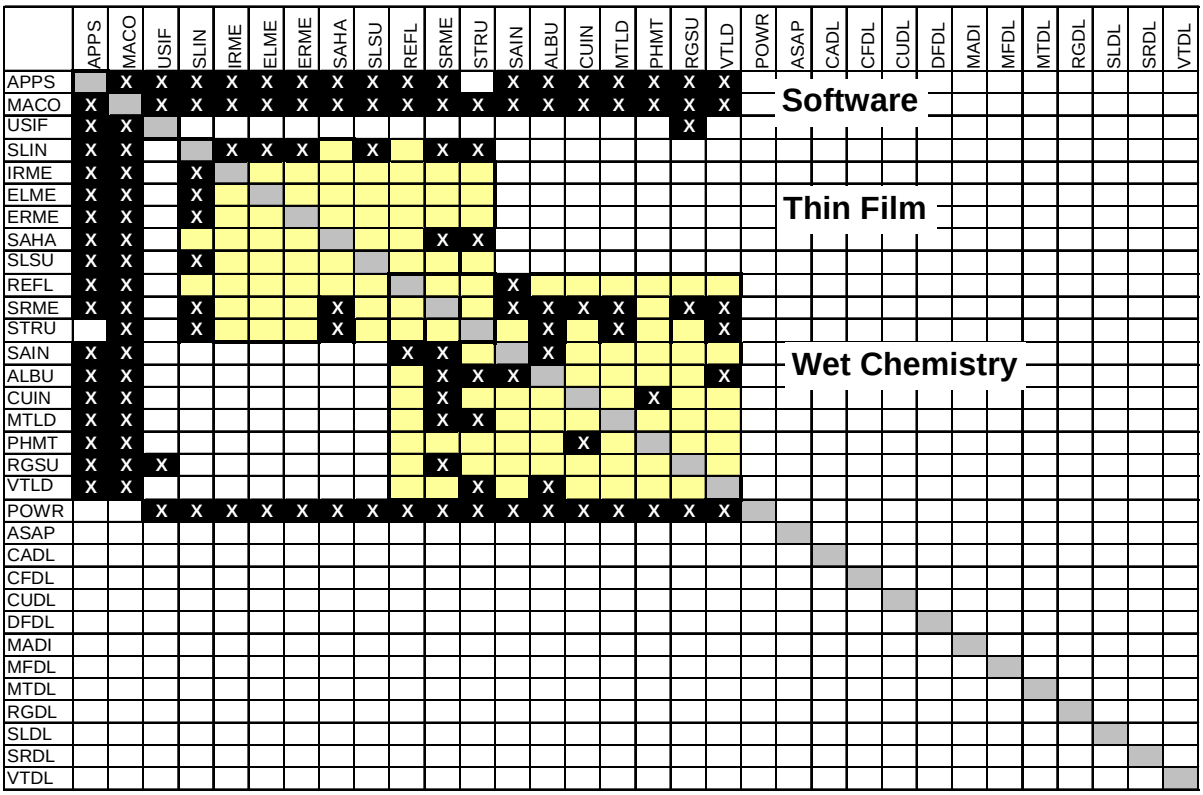


Figure 5-8: DSM Constructed Based on Requirements and Architecture Definition

Compare the DSM in Figure 5-7 to that in Figure 5-8, 14 marks (7 pairs of interactions) are missing in Figure 5-8: DSM Constructed Based on Requirements and Architecture Definition. They are:

REFL—SLIN	SLIN—REFL
SRME—MTLD	MTLD—SRME
STRU—CUIN	CUIN—STRU
RGSU—STRU	STRU—RGSU
SRME—STRU	STRU—SRME
STRU—SAIN	SAIN—STRU
SLSU—STRU	STRU—SLSU

All of the above interactions are due to spatial relationship. Therefore, building a Design Matrix from requirements does not easily capture the packaging and assembly issues in the system interface. Fortunately, this problem can be resolved by employing another existing technique--the Datum Flow Chain method—so that assembly interfaces can be predicted at the early stage of the design process [Mantripragada and Whitney (1998)].

In Ulrich and Eppinger (2000 p. 195), spatial relationship is identified as one of the *incidental interactions* in the system. Incidental interaction is the interaction that arises because of the particular physical implementation of functional elements or because of the geometric arrangement of the chunks in the system. Therefore, using requirements to build DSM misses the incidental interactions in the system. Other system engineering documents and techniques are needed in addition to requirements document.

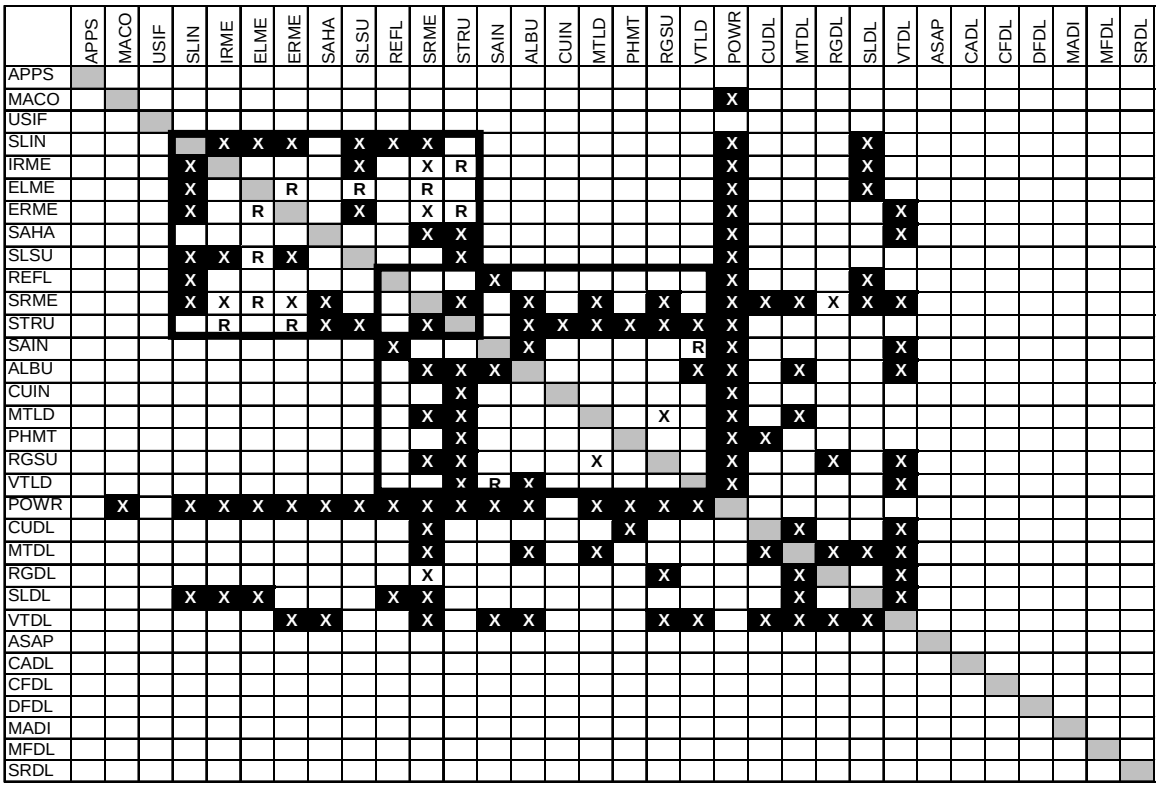
5.3.1.2 The DSM Constructed by the Experts

5.3.1.2.1 The Experts' DSM's

The system engineers and design experts at JNJ OCD have built two DSM's using the traditional DSM construction method in the time frame of February and August 2001. The

resulting DSM's are presented in Figure 5-9 and Figure 5-10. The marks captured by the experts in both February and August are highlighted.

Compare the matrices in Figure 5-9 and Figure 5-10, 130 marks in either matrix match. The DSM in February (Figure 5-9) has 20 additional marks unmatched by the August DSM (Figure 5-10). Twelve out of the 20 unmatched marks are due to reliability, because when the experts constructed the DSM in August, they did not include the interactions that may have contributions to reliability, while in February they did. The other 8 marks missed by the experts in August were because either the experts thought less of the importance of those interactions in August, or they simply forgot to put marks in.



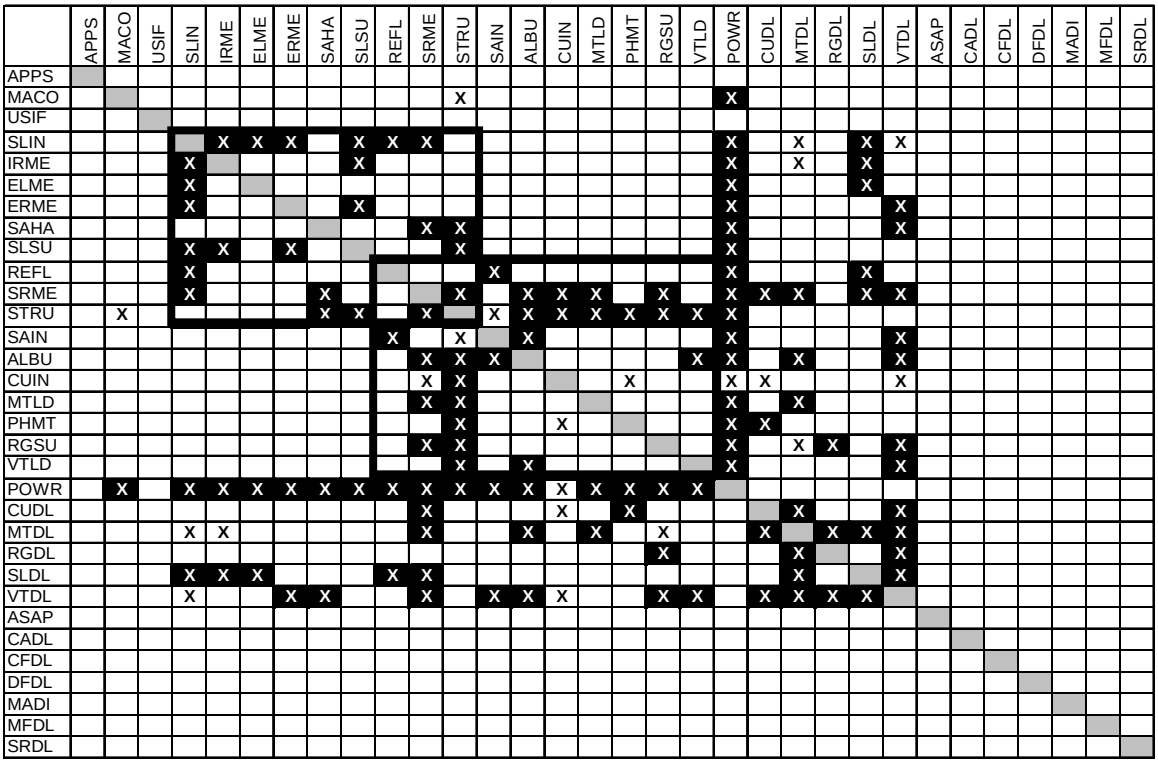
R: means the interaction is due to reliability issue, and was not predicted by August Expert DSM.

X This interaction is predicted by both the February and August Expert DSM.

X: means the interaction is due to other issues, and was not predicted by August Expert DSM.

Figure 5-9: February Expert DSM

The DSM in August (Figure 5-10) has 22 extra marks unmatched by the February DSM (Figure 5-9). These marks were added mainly because the experts' knowledge about the design and the system has increased over the past six months. More interactions were discovered as design moved on. In general cases, as more interactions in the system may be discovered as time goes by, some of the previously identified interactions may no longer be valid. However, in this case study, when the JNJ system engineers reviewed February DSM, they still thought all of the interactions identified in February were valid in August, even if they missed some of the interactions in the August DSM.

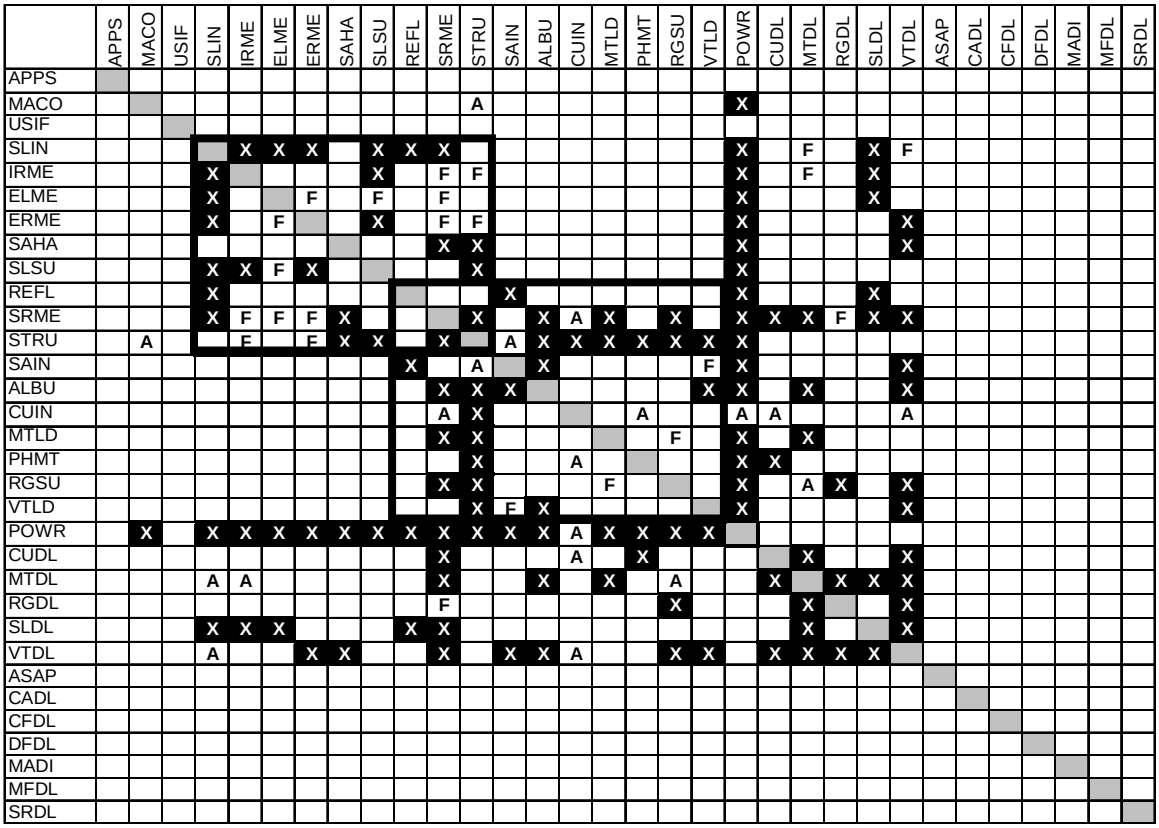


X: this interaction was only captured by August expert DSM.

X This interaction is predicted by both the February and August Expert DSM.

Figure 5-10: August Expert DSM

Since both February and August DSM's missed some of the system interactions, combining both DSM's can give us a quite complete view of the system interactions perceived by the experts. The combined matrix is shown in Figure 5-11.



F--this mark appeared in February DSM

A—this mark appeared in August DSM

X This interaction is predicted by both the February and August Expert DSM.

Figure 5-11: Combined Experts Prediction DSM from February and August

5.3.1.3 Compare the Requirement-driven DSM to the Experts' DSM

To verify the effectiveness of the matrix conversion method, the DSM from the requirements is compared with the experts' DSM's. One of the main objectives of this case study was to investigate the effectiveness of building a DSM from requirements only. Therefore, the DSM made from requirement documents only (Figure 5-8) is used to compare with the experts' DSM made from traditional DSM construction method (Figure 5-11). The result of

comparison is in Figure 5-12. The highlighted cells are the marks that were identified by both expert DSM's and the author's prediction based on requirements.

	APPS	MACO	USIF	SLIN	IRME	ELME	ERME	SAHA	SLSU	REFL	SRME	STRU	SAIN	ALBU	CUIN	MTLD	PHMT	RGSU	VTLD	POWR	CUDL	MTDL	RGDL	SLDL	VTDL	ASAP	CADL	CFDL
APPS		q	q	q	q	q	q	q	q	q	q		q	q	q	q	q	q	q									
MACO	q		q	q	q	q	q	q	q	q	q	A,q	q	q	q	q	q	q	q	F,A								
USIF	q	q																q										
SLIN	q	q			F, A, q	F, A, q	F, A, q		F, A, q	F,A	F, A, q	q								F,A		A		F,A	A			
IRME	q	q		F, A, q					F,A		F	F								F,A		A		F,A				
ELME	q	q		F, A, q			F		F		F									F,A				F,A				
ERME	q	q		F, A, q		F			F,A		F	F								F,A					F,A			
SAHA	q	q									F, A, q	F, A, q								F,A					F,A			
SLSU	q	q		F, A, q	F,A	F	F,A					F,A								F,A								
REFL	q	q		F,A									F, A, q							F,A				F,A				
SRME	q	q		F, A, q	F	F	F	F, A, q				F,A	q	F, A, q	8,q	F, A, q		F, A, q	q	F,A	F,A	F,A	F	F,A	F,A			
STRU		A,q		q	F		F	F, A, q	F,A		F,A		A	F, A, q	F,A	F, A, q	F,A	F,A	F, A, q	F,A								
SAIN	q	q								F, A, q	q	A		F, A, q					F	F,A					F,A			
ALBU	q	q									F, A, q	F, A, q	F, A, q						F, A, q	F,A		F,A			F,A			
CUIN	q	q									A,q	F,A					A,q			A	A					A			
MTLD	q	q									F, A, q	F, A, q						F		F,A		F,A						
PHMT	q	q										F,A			A,q					F,A	F,A								
RGSU	q	q	q								F, A, q	F,A				F				F,A		A	F,A		F,A			
VTLD	q	q									q	F, A, q	F	F, A, q						F,A					F,A			
POWR		F,A	q	F, A, q	F, A, q	F, A, q	F, A, q	F, A, q	F, A, q	F, A, q	F, A, q	F, A, q	F, A, q	F, A, q	8,q	F, A, q	F, A, q	F, A, q	F, A, q									
CUDL											F,A				A		F,A					F,A			F,A			
MTDL				A	A						F,A			F,A		F,A		A			F,A		F,A	F,A	F,A			
RGDL											F,A							F,A					F,A		F,A			
SLDL				F,A	F,A	F,A				F,A	F,A											F,A			F,A			
VTDL				A			F,A	F,A			F,A		F,A	F,A	A			F,A	F,A		F,A	F,A	F,A	F,A			
ASAP																												
CADL																												
CFDL																												
DFDL																												
MADI																												
MFDL																												
SRDL																												

F—marks in the February expert DSM

A—marks in the August expert DSM

q—marks in the author's prediction DSM made from requirements document only

Figure 5-12: Compare the Experts' DSM with The DSM from Requirements

There are total 247 interaction marks in the combined DSM in Figure 5-12. Only 53 of the marks match (Figure 5-13). Why are there so many marks unmatched? Does this mean the method of building a DSM from requirements is ineffective? The discussion is carried out below.

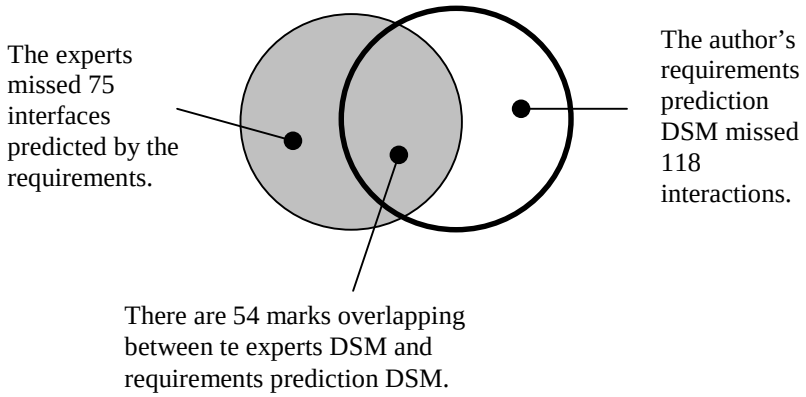
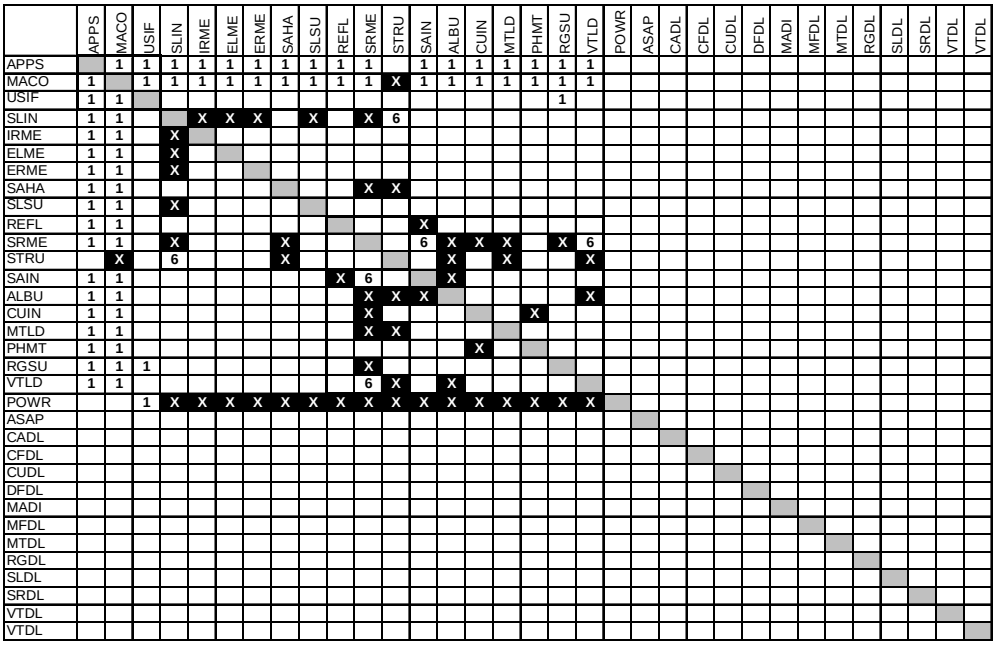


Figure 5-13: Matching and Unmatched Marks in the Requirements DSM and the Expert DSM

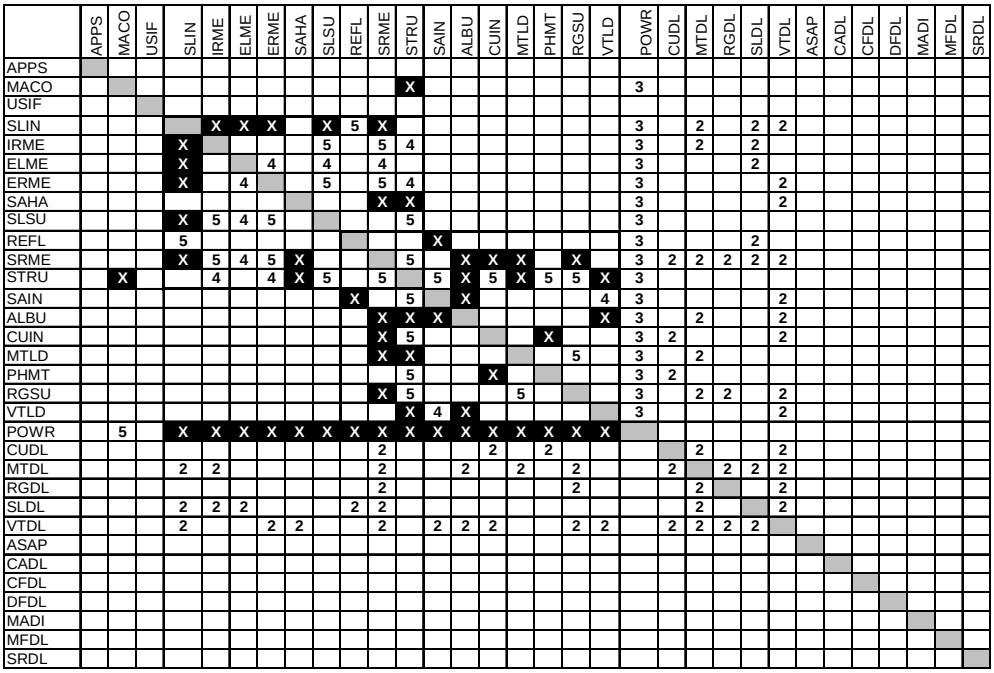
5.3.1.3.1 Analysis of Unmatched Marks

The differences between the author's DSM and the experts' DSM are categorized into six types as marked out in Figure 5-14 and Figure 5-15. The highlighted cells indicate the interactions captured by both matrices. The next portion of this paper contains discussion of each one of the six types of unmatched marks in Figure 12 and 13.



1: Type 1 unmatched marks—interaction between hardware and software.
6: Type 6 unmatched marks—miscellaneous interactions missed by the experts.

Figure 5-14: The Requirement Prediction DSM



2: Type 2 unmatched marks—interaction between assay chemistry and hardware.
3: Type 3 unmatched marks—feedback marks from power subsystem.
4: Type 4 unmatched marks—interactions due to reliability.
5: Type 5 unmatched marks—interactions due to function or spatial relationships.

Figure 5-15: The Expert DSM Combining February and August Results

Type 1 Unmatched Marks—Interaction Between Hardware and Software

This category of unmatched interactions in the DSM's are marked by "1" in Figure 5-14. There are total 69 marks of this type. The requirement prediction DSM has these marks, but they are missing from the experts' prediction matrix. Mark Raymond, the JNJ OCD systems integration manager, pointed out that when the experts built the DSM, they did not consider the interactions between the software and hardware. In fact the software engineers were not invited to either DSM building exercises.

Mark Raymond's comment reflects an organization gap at JNJ between software and hardware. Historically, the two groups were separate organizations and had different work cultures. The experts involved in building the DSM were from hardware and chemistry groups. Therefore the DSM's they built did not capture the interactions with software group. In addition, the author observed that the software subsystems not only did not have requirement trace-ability documented in the OASIS requirement management software—RequisitPro, but also did not write subsystem requirements document in the way that is consistent with the hardware subsystems. The author therefore had to interview the software system experts to capture the decomposition of the product requirements related to software subsystems.

The organization gap between software and hardware is a warning sign for managing systems in the OASIS project. The DSM in Figure 5-14 shows that very complicated and tight coupling exists between the hardware and software design. Yet, the software and hardware teams were not using the same tools and terms in their design. The OASIS systems engineering group should be cautious about the organizational gap in order to avoid delay and rework in the integration and verification phase.

Type 2 Unmatched Marks—Interaction Between Assay Chemistry and Hardware

The second category of the unmatched interactions are marked by "2" in Figure 5-15. There are total 64 marks of this type. The author's prediction DSM (Figure 5-14) does not have these marks, while the experts' prediction DSM (Figure 5-14) has them. These marks are

about the system interactions related to the design and delivery of Assay, thin films, cuvettes, tips, etc. The author's DSM prediction missed these interactions because the Delivery and Assay subsystems did not write Subsystem design requirement document. Consequently, the author was unable to trace the PRD to SSRD and derive the resulting system interactions.

There are three major reasons for which the delivery subsystems requirements document were not written. First, historically, the hardware and Assay/Chemistry groups were two separate organizations. Although the hardware group were educated about requirements management and put in the effort to develop requirements document, the Assay/Chemistry group was not as involved in the documentation process. Second, most of the delivery subsystems reuse existing components (e.g. reagents, recipes, etc.). The existing components have already been sold and used in the JNJ OCD analyzers for many years. The Assay /Chemistry group did not want to spend the time to write up a requirements document for the existing parts any more. Third, the new components in the delivery subsystems, such as Cuvettes, were co-developed with outside vendors. The systems engineering team in OASIS has not pushed the requirements management system to the vendors yet. Therefore, the author's DSM built from the requirement document missed these marks.

Yet, the DSM in Figure 5-15 shows tight coupling between the delivery subsystems and the rest of the analyzer. Therefore, the system management and integration team should pay close attention to this organization gap and avoid delays later in the program.

Type 3 Unmatched Marks—The Feedback from the Power Distribution Subsystem

The author's prediction DSM does not have the feed back marks from the power distribution subsystems (POWR), while the expert prediction DSM does (See Figure 5-15 mark "3"). There are total 17 marks of this type. There are two reasons for which the author's prediction DSM missed these feedback marks.

The first reason is that the POWR subsystem engineer said there would be very little feedback because the requirements on the power subsystem was easily met. Therefore, the author's matrix ignored the weak feedback interactions. The author's DSM prediction

reflects a more recent and realistic situation of the system, while the expert prediction reflects the theoretical and conservative situation. Neither view is better than the other one. When used for planning, the author's DSM provides an aggressive view of the system while the experts' view is more conservative. It is the manager's judgment to decide which one he/she prefers to work with.

The second reason is that according to the current architecture definition, the harnessing and electric boards are considered to be low-level components and are not discussed in the PRD. The author therefore did not capture the marks due to packaging of the electrical system, while the experts considered harnessing and boards as a part of the POWR system, and hence identified feedback marks to the hardware subsystems. The lesson learned here is that the subsystem architecture should take into account of the harnessing and boards although they may appear very low in the system hierarchy. Harnessing and boards can introduce important system interactions that may relate to wiring errors, heat dissipation, assembly problems, etc.

The conclusion hence is that it is better to include the feedback marks between the power subsystems and the rest of the hardware subsystems. Yet the missing marks in Figure 5-15 are not due to the fault of the matrix conversion method, but rather the incomplete input information.

Type 4 Unmatched Marks—The Interaction Introduced by Reliability

The DSM built from requirements (Figure 5-14) missed the interactions due to reliability that were captured by the DSM built by the experts (Figure 5-15). This type of unmatched marks are marked out by "4" in Figure 5-15, and they are:

ERME—>ELME	ELME—>ERME
SLSU—>ELME	ELME—>SLSU
SRME—>ELME	ELME—>SRME
STRU—>IRME	IRME—>STRU
STRU—>ERME	ERME—>STRU

VTLD—>SAIN SAIN—>VTLD

There are total 12 reliability types of interactions the requirements failed to predict. These marks are missing from the author's prediction matrix because the reliability product requirement is an emergent property of the system, which is difficult to predict before the system is put together. Based on the conversation with the reliability engineer for the OASIS program, the reliability requirement is usually allocated into each subsystem as a target. Whether the target will be met, or how each subsystem trade off with one another cannot be predicted until the prototype is put together and some tests are run. Some of the system interaction may not be discovered until the product is in the field. Very little system interaction related to reliability can be predicted at early stage of the design. Therefore, starting from the requirements fails to capture the reliability type of interactions that experienced experts know in their minds.

A remedy to the situation is to collect past war stories and try to avoid them in the next design. The OASIS Hazard Analysis and Mitigation (HAMG) document was produced exactly for this purpose. The HAMG document for the OASIS program contained 961 mitigation cases. However, only 161 were traced to subsystems in the current analyzer design (as of mid July 2001). The cause of the rest was not traced yet. Among the 161, only 7 mitigations were traced into more than one subsystem. Among the 7, only 4 revealed system interactions.

4 / 961 = 0.4%

Therefore, the HAMG contributed very little to the system interactions in the DSM. Bartkowski (2000) suggests that the reliability problems arise from chains effects that links several subsystems to each other are the most difficult to discover. To improve the effect of the HAMG document, two possible remedies exist:

1. The JNJ OCD engineers could improve in flowing down the HAMG into the subsystems so that past lessons learned could be reflected on the current design to avoid potential

pitfalls. Currently, only 161 out of the 961 mitigations were related to subsystem designs.

2. As indicated by Ted Farrell, director of the OASIS systems engineering group, most of the mitigation probably related to only one subsystem. This could be seen in the existing data too. Among the 161 mitigations that were related to subsystems, only 7 related to more than one subsystem. Therefore, the HAMG might not be a good source to discover lessons-learned about the system interactions related to reliability after all. Then another better way to record the product history is needed.

In conclusion, the interactions caused by reliability are hard to predict from the requirement documents. Predicting reliability types of interactions is a weakness of the matrix conversion method. The reason is that meeting the reliability requirements depends on emergent system properties.

Type 5 Unmatched Interactions—Functional and Spatial Relationships

The rest of the unmatched marks in the expert prediction DSM (Figure 5-15) are due to one of the two reasons—interaction due to functional reason (total 11 marks) or interaction due to spatial relationship (total 14 marks). These marks are represented by "5" in Figure 5-15. The causes for which these marks were not captured by the requirements-driven DSM (Figure 5-14) are explained below.

- REFL-SLIN

This relationship exists because the reflectometer needs to align with the slide position to read the light reflection. The slide incubator determines the position of the slides. Therefore, reflectometer interacts with the slide incubator.

This interface is a function type of relationship. The OASIS Function Flow Diagram captured this relationship. It is reflected on the DSM built from combined document sources (Figure 5-7). However, this relationship was not reflected in the requirement decomposition document. Hence this interaction was missing from Figure 5-14.

Since the interface is relevant to the success and quality of a function, this interface should have been recorded in the requirement documents.

- SLSU-IRME

This is a functional relationship. When the immuno-rate wash fluids are metered to the slides, the slide supply subsystem determines the location of the slides relative to the immuno-rate metering system. This interface is critical to performing the function of dispensing the immuno-rate washing fluid. However, the requirement documents do not record this interface. Even the Mechanical Interface Document does not record this interaction. This is an interaction that should have been documented.

- SLSU-ERME

This interaction is similar to that between the SLSU and IRME (above).

- STRU-SRME

This is a spatial relationship. The frame determines the location of the metering system. As a spatial relationship, it is form-dependent, and hence is usually not incorporated into requirement documents. Therefore, the prediction DSM derived from requirements does not contain this interface. In this case study, however, the mechanical interface document captured this interaction.

For all of the form-dependent spatial relationship, another method is needed in order to predict them early in the design process. The technique of building a DSM based on requirements cannot capture this type of relationship. An excellent substitute is the Datum Flow Chain method, which is capable of predicting the relationships in an assembly [Mantripragada and Whitney (1998)].

- SAIN-STRU

This is a spatial relationship with the same situation as the “STRU-SRME” interface.

- CUIN-STRU

This is a spatial relationship with the same situation as the “STRU-SRME” interface.

- RGSU-STRU

This is a spatial relationship with the same situation as the “STRU-SRME” interface.

- PHMT-STRU

This is a spatial relationship with the same situation as the “STRU-SRME” interface.

- SLSU-STRU

This is a spatial relationship with the same situation as the “STRU-SRME” interface.

- MTLD-RGSU

This interaction is due to spatial arrangement. Therefore, the same discussion as in the “STRU-SRME” relationship applies. Furthermore, even the Mechanical Interface Document did not capture this interface.

- SRME-IRME and SRME-ERME

No PRD has been decomposed into these subsystems together, but the SSRD has record of one subsystem interacts with another. Therefore, the author’s matrix conversion method, which traced the PRD decomposition, did not find these relationships. This reveals that there is a problem about how the SSRD and PRD are written. When an interface is discovered at subsystem level, the engineers should go back to the system level requirements to make sure this interface is captured at system level. Otherwise, these interfaces can easily be missed because the system engineers only pay attention to the system level interactions. Again, missing information rather than the incompetence of the matrix conversion method caused the failure.

From the above discussion, we can conclude:

1. The functional types of missing marks reveal warnings of the completeness of the requirement documents. These marks are missing not because of the capability of the matrix conversion method, but because of incomplete requirements document. Among the above interactions, 5 pairs (total 10) belong to this type. They are:
    - REFL-SLIN
    - SLSU-IRME
    - SLSU-ERME
    - SRME-IRME
    - SRME-ERME

    The JNJ engineers should revise the requirement documents to include these missing interfaces.

2. The packaging types of missing marks cannot be captured in requirements flow down structure, and hence cannot be easily predicted by the matrix conversion method. Therefore, besides the requirements document, we shall also use methods such as Datum Flow Chain or documents such as the OASIS Mechanical Interface Document to predict these spatial relationships. Among the above missing marks, the following 7 pairs of interfaces (total 14 interactions) are of this type:
    - STRU-SRME
    - STRU-SAIN
    - STRU-CUIN
    - STRU-RGSU
    - STRU-PHMT
    - STRU-SLSU
    - RGSU-MTLD

    All of these interfaces could have been captured if we were to use the OASIS Mechanical Interface Document (Figure 5-7).

Type 6 Unmatched Marks—Miscellaneous Interactions Missed by the Experts

Figure 5-14 shows that the experts were unable to predict all of the interactions predicted by the matrix conversion method either. These unmatched interactions are marked "6" in Figure 5-14. Therefore, the author's matrix conversion method enabled us to capture more complete set of interactions in a systematic manner.

5.3.1.3.2 Summary of the Data

The above discussion has shown:

Total amount of system interfaces	247
Matching marks between the experts DSM and the author's prediction	54
Unmatched marks between the experts DSM and the author's prediction	193
Type 1 software/hardware interactions	69
Type 2 hardware / assay delivery subsystem interactions	64
Type 3 power distribution subsystem feedback	17
Type 4 reliablity induced interactions	12
Type 5a function types of interactions	11
Type 5b spatial types of interactions	14
Type 6 experts missed	6

Figure 5-16 further summarizes the data.

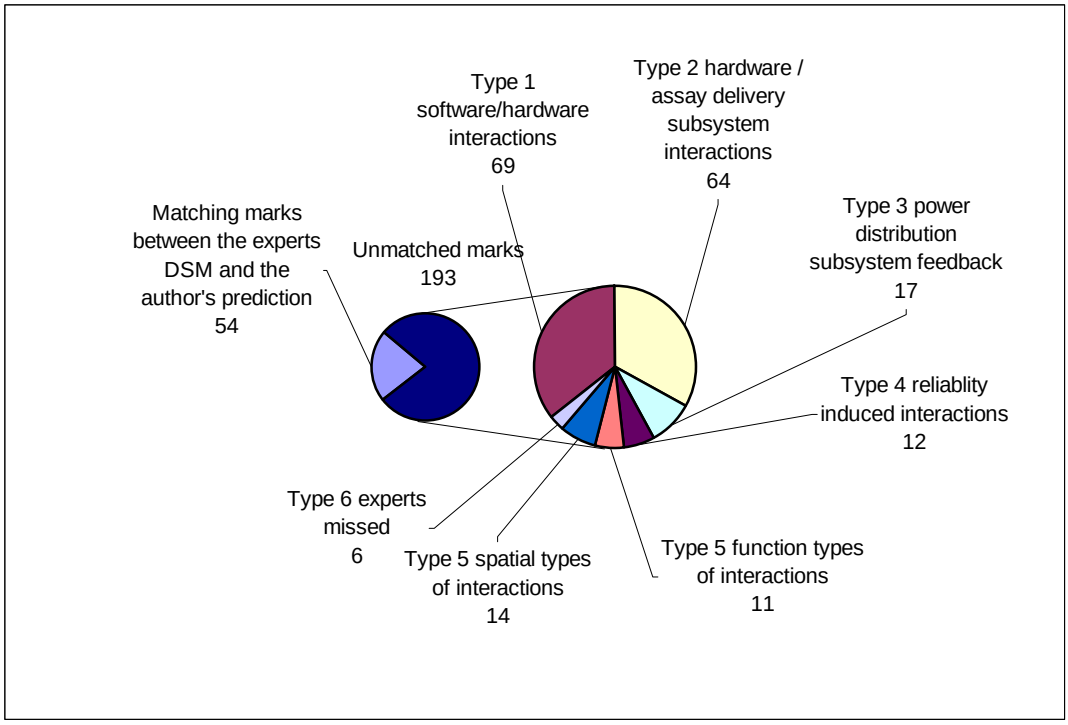


Figure 5-16: Percentage of Various Unmatched System Interaction Marks

The reasons for each type of missing marks are summarized below:

- Type 1 and 2 unmatched marks regarding software and assay delivery subsystems (total 69% of all unmatched marks) are due to the inconsistency of requirement documentation in different parts of the organization, not the matrix conversion method. The method is capable of capturing these two types of marks if the documentation is complete.
- Type 3 unmatched marks concerning the power subsystem (total 8.8% of all unmatched marks) are really a subjective choice of the system engineers. It does not reflect of the capability of the matrix conversion method.
- Type 4 unmatched marks concerning reliability (total 6.2% of all missing marks) reflect the limitation of the matrix conversion method. The reliability interactions cannot be predicted from requirements. We need other types of methods or documents to predict interactions due to reliability.

- Type 5 unmatched marks concerning function and spatial relationship (12.9% of all unmatched marks) can be avoided if the engineers update their requirement documents, and if other methods for predicting spatial relationships, such as Datum Flow Chain, are employed.
- Type 6 unmatched marks--miscellaneous missing marks by the experts (total 3.1% of all unmatched marks) show that the matrix conversion method provides a systematic way to capture the system interfaces that are sometimes even overlooked by design experts.

Type of Unmatched Mark	Number of unmatched marks	Reason for Missing	Who Missed them	Remedy	Problem of the matrix conversion method
(1) Hardware-software interaction	69	The experts did not involve software people in the DSM exercises.	JNJ engineers	involve software people in the next DSM building exercise	no
(2) Assay-hardware interaction	64	No Assay design requirement has been documented.	requirements DSM	JNJ chemists produce assay design requirements documents.	no
(3) Power subsystem interaction	17	The power subsystem engineer says there will be no need for information feedback to other subsystems.	requirements DSM	Does not count as a mistake.	no
(4) Reliability induced interaction	**12**	**Reliablity requirement decomposition is difficult to use to predict system level tradeoffs.**	**requirements DSM**	**Use past design history on relialbity issues (e.g. the hazard analysis document at JNJ)**	**yes**
(5-1) Function types of interaction	11	Not reflected in reuqirements decomposition structure.	requirements DSM	better requirements writing and management by JNJ engineers	no
(5-2) Spatial types of interaction	**14**	**Spatial relationship is not detailed by requirements.**	**requirements DSM**	**Use Datum Flow Chain**	**yes**
(6) experts missed interaction	6	experts did not bring them up during DSM building exercises.	JNJ engineers	JNJ engineers can learn from the requirements driven DSM.	no

Table 5-2: Summary of Unmatched Marks

Table 5-2 shows many of the unmatched marks could have been prevented if we had known better about the rules of constructing DSM from requirements. The rules we learned here are:

1. JNJ engineers must involve the software engineers in the DSM building exercise. In general, the DSM building exercise must involve all relevant teams.

2. If requirements are used to predict system interactions, it is better to keep a complete set of requirements document. Thus, JNJ chemists should write the requirements for the assay, slides, and cuvetts.
3. The requirement traceability relationship should be updated constantly to reflect the learning about system interactions.

If all of the above rules had been followed, only 26 marks would have been missing in the DSM built from the requirements. The potential overlap between the DSM predicted from requirements and the DSM built by the experts could be those shown in Figure 5-17. If Datum Flow Chain method were employed, the overlapping would have been even greater.

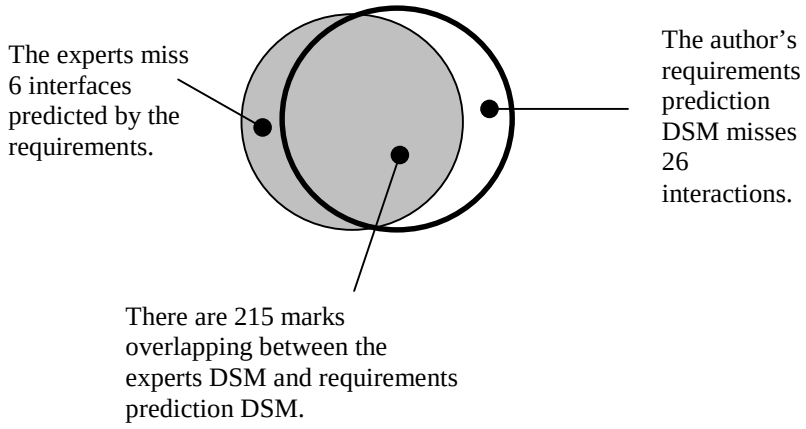


Figure 5-17: Potential Overlapping between the Requirements and Expert DSM

From the above discussion, we can claim the matrix conversion method is capable of predict most of the system interactions. Another note is that the above observation shows the weakness of the Axiomatic Design's Design Matrix. An uncoupled or decoupled design may indeed be coupled due to reliability or spatial relationship, but appear uncoupled or decoupled because the coupling due to reliability or packaging cannot be captured through requirements.

5.3.1.4 System Interaction Density

The DSM system interaction density concept is introduced in Chapter 4 (see 4.3.5.2). To compare the marks per row ratio in both DSM's, we shall compare only the parts of the matrices where actual thoughts were put in. From the discussions in 5.3.1.3.1, the interactions with the delivery subsystems were not considered in the author's matrix conversion method (Type 2 in Figure 5-15). The software and hardware interactions were not considered in the experts' prediction DSM (Type 1 in Figure 5-14). Therefore, in the requirements prediction DSM (Figure 5-14), only the portion from "APPS" to "POWR" was considered to calculate the interaction density ratio. In the experts' prediction DSM (Figure 5-15), only the portion from "SLIN" to "SRDL" was taken into account. Therefore, we have:

The author's prediction DSM Interaction Density = 129 / 20 = 6.45
The experts' prediction DSM Interaction Density = 172 / 29 = 5.93

Both ratios are very close to 6. Therefore, probably enough thoughts were put into building the DSM from requirements and the DSM built from experts.

However, we still need to note that the number of system elements (the subsystems in this case) was kept constant in this case study. This constraint ruled out the flexibility of adjusting the selection of system elements some other case studies had. Since the marks-per-row density may be a reflection of the human cognitive limitation when dealing with complex systems, constraining the amount of system elements in the DSM limits the flexibility of human thinking. Therefore, the above observation about the interaction density ratio provides only a reference for whether the DSM made from requirements was good enough.

5.3.1.5 The Topology of System Interfaces

Only counting the number of overlapping marks in both DSM's are not enough to compare their similarity. Figure 5-18 demonstrates this point. Both DSM's in Figure 5-18 have the same amount of interactions, but the system iterations loops are different. In the system

represented by the DSM on the left side, *A, B, C,* and *D* for one iteration loop. In the system represented by the DSM on the right side, only *B, C,* and *D* are involved in iteration. *A* is not.

Figure 5-18: Two DSM's with the Same Number of Elements but Different Topology

Therefore, system topology in our sense include the following three aspects:

1. Which element in the system is not involved in iteration?
2. Which elements are involved in iteration, and which iteration are they in?
3. What are the sequences of elements that are not involved in iterations?

The best way to identify the system topology is to partition the DSM.

The DSM from requirements document (Figure 5-14) and the experts' prediction DSM (Figure 5-15) are compared for their similarities in topology. The results of the partitioning are listed below. Note the subsystems in the parenthesis are involved in an iteration loop. Each level includes subsystems either involved in the same iteration loop or can be designed in parallel. Subsystems in later levels require inputs from subsystems in earlier levels.

- The requirements driven DSM (Figure 5-14):

Level 1--(APPS, MACO, USIF, SLIN, IRME, ELME, ERME, SAHA, SLSU, REFL, SRME, STRU, SAIN, ALBU, CUIN, MTLD, PHMT, RGSU, VTLD), CUDL, MTDL, RGDL, SLDL, VTDL, ASAP, CADL, CFDL, DFDL, MADI, MFDL, SRDL.

Level 2—POWR

- The experts' combined prediction DSM (Figure 5-15):

Level 1—APPS

Level 2—(MACO, SLIN, IRME, ELME, ERME, SAHA, SLSU, REFL, SRME, STRU, SAIN, ALBU, CUIN, MTLD, PHMT, RGSU, VTLD, POWR, CUDL, MTDL, RGDL, SLDL, VTDL), USIF, ASAP, CADL, CFDL, DFDL, MADI, MFDL, SRDL

The partitioning results of the above two DSM's are very similar except for the following elements:

1. The experts' DSM did not consider the interactions between software and hardware. Therefore, the APPS and USIF subsystems are grouped differently in the two DSM's.
2. The author's DSM did not consider the interactions introduced by the delivery subsystems. Therefore, the delivery types of subsystem elements—CUDL, MTDL, RGDL, SLDL, VTDL—were excluded from the major iteration loop in the requirements driven DSM.
3. The author's DSM omitted the feedback marks of the POWR subsystem to the rest of the hardware. Therefore, the POWR subsystem is not included in the major iteration loop in the requirements driven DSM.

All above three cases can be eliminated as explained in *5.3.1.3.1 Analysis of Unmatched Marks*. Therefore, the matrix conversion method matched the expert prediction DSM well topologically.

5.3.1.6 The System Element Priority List

The DSM's are built for helping systems engineering planning. The ultimate test for the usefulness of the DSM built from requirements is to test whether it provides the same advice for the planning of the system integration activities as the DSM built by the JNJ experts.

When the experts at JNJ OCD constructed the DSM, they assigned weighing factors to the interactions. Therefore, JNJ engineers identified a priority list of the system elements ranking from the most important to system interfaces to the least. Here are their top seven subsystems:

1. POWR
2. SRME
3. SLIN
4. STRU
5. IRME
6. ALBU
7. SAIN

The author's DSM from requirements using the matrix conversion method is binary. The following rules were used to pick the elements in the system that deserve the most attention:

- The elements that cause the largest iteration loop are the most important. These are the elements that interact with a lot of other elements both in the horizontal direction and vertical direction in the DSM. For instance, the APPS subsystem is a large-iteration-causing element (Figure 5-14).
- The elements that interface different iteration blocks are important.

Therefore, based on the DSM in Figure 5-14 (ignore the software part as the experts did), the priority list is:

1. SRME
2. STRU
3. SLIN
4. ALBU
5. SAIN
6. REFL

7. POWR

The above two lists include mostly the same elements, with three exceptions explained as follows:

- The author failed to predict the importance of the IRME subsystem because the reliability types of interactions were not captured in the requirement flow down process. The observation again demonstrates the limitation of relying on requirements to predict system interfaces.
- The experts failed to predict the importance of the REFL subsystem. The cause might be that the weighing factor value given to that subsystem is low. However, in the author's opinion, since REFL and SAIN belong to two separate subsystem design groups, their interaction is worth close attention.
- The POWR subsystem is in the first place in the experts' list but the last in the author's list. The reason is that the experts' DSM captured the feeding back actions of the POWR subsystem to the rest of the hardware, while the author took the POWR subsystem engineer's view not to include the weak feedback interactions.

Despite the differences, both priority lists warn the system engineers about the same things regarding system integration issues. Therefore, the DSM from requirements is capable of predicting the priority of the system elements.

5.3.1.7 What Makes Two DSM's Similar?

All of the discussions in this section so far concern the similarity between two DSM's. In this case study, it was the similarity between the DSM built from the requirements and the DSM built by the experts. We may summarize the measures used to compare the similarities between two DSM's. These measures may be useful for other DSM studies.

1. Having marks at the same location in the DSM

This is a very obvious comparison criterion. A large portion of this section was devoted to comparison the overlapping and unmatched marks between the requirements DSM and the

expert DSM. Having the same marks at the same location in the DSM definitely make two DSM's similar or the same. However, not having the same marks at the same location does not necessarily mean a DSM is incorrect or useless. The next few measures looks at the comparison from other perspectives of the problem.

2. The System Topology

When the results of DSM partitioning reveal the same sequential, parallel, and iterative relationships among elements in the DSM, the two DSM's are similar. A DSM may miss a few details, but if it reveals all of the important system level issues such as iterations, it is still a valid and useful DSM.

3. The System Element Priority List

When two DSM's warn about the same elements in the system, the two DSM's are similar from project management point of view.

4. The Convergence Rate of System Iterations

This measure was not used in this case study. According to Smith and Eppinger (1997), the Eigen values of iterations can tell the convergence speed of iterations. Therefore, we may assign sensitivity values to the interactions in an iteration block, and compare the convergence values.

For the future research, it is worth investigating whether there are techniques in linear algebra regarding the similarities of two matrices besides the Eigen value measure. Since a DSM is equivalent to a digraph, the literatures on comparing the similarity of graphs will also help on this topic.

5.3.1.8 Summary of the Effectiveness of Building a DSM from Requirements

This first objective for this thesis research was to find out if we could use the matrix transformation method to obtain a DSM from requirements again for a different product from CVC's. The above discussion showed that a DSM was built using OASIS requirements.

The second objective was to see how realistic the requirements DSM's predictions are. Comparing the requirements DSM to the DSM JNJ engineers built based on their experience in the detailed design phase, we made the following observations:

- Most of the unmatched marks between the requirements DSM and the expert DSM are due to missing information, rather than the method.
- The resulting DSM from requirements has an interaction density ratio around 6.
- The topology of the resulting DSM is very similar to that of the expert DSM. They both identified the same elements involved in the iterations.
- Both requirements DSM and expert DSM predicted very similar priority list.

Therefore, the DSM built from the requirements is close to what expert engineers learned from experience, and is useful for the purpose of planning out system integration work.

In the process of comparing the requirements DSM with the expert DSM, we discovered the weaknesses of the idea of building a DSM from requirements.

- We cannot predict spatial relationship from requirements. Other techniques such as Datum Flow Chain must be used.
- We cannot predict emergent properties of the system, such as the system tradeoff's for meeting the reliability requirement.

However, as a planning tool, the DSM constructed at early stage of the design process does not have to be complete or perfect. It was seen that the DSM constructed from the requirements gave the same advice about system topology and element priority list as the DSM constructed based on expert knowledge. In addition, the DSM from requirements revealed organization communication gaps experts did not pay attention to. Therefore, despite the limitations, the technique of building a DSM from requirements is still useful.

Furthermore, we learned from this case that to get a DSM from requirements, we might take a shortcut without going through DM, if we are only interested in interactions at one level of the system hierarchy. The DSM can then be transferred back to a DM, revealing the requirements trace-ability.

5.3.2 The Requirements Decomposition Process vs. Various Types of Requirements

The matrix conversion method developed by the author was built upon the basis of Axiomatic Design Matrix [Suh (2000)], and hence is inevitably influenced by the capability of Axiomatic Design method. The most critical influence is the requirements decomposition. The Axiomatic Design theory classifies all requirements into two types. The first type is the Functional Requirement, which can be decomposed into lower level requirements using the zigzagging method. If a requirement cannot be decomposed using the zigzagging method, then the requirement is called a constraint in Axiomatic Design. Axiomatic Design does not provide a systematic approach in dealing with constraints in the design process. Chapter 2 contains more detailed discussions on the limitations of Axiomatic Design.

Since one of the goals of this research is to understand how much of the system interaction we can predict at early stage of the design process from design requirements, whether all requirements can be addressed using the matrix conversion method is of high importance. This section of the discussion is interested in understanding the following issues:

1. What types of product level requirement (PRD) were decomposed into subsystems? How were they decomposed? Note when a requirement is decomposable, we mean that requirement can generate subsystem level requirements using the Axiomatic Design zigzagging method.
2. What types of PRD have not been decomposed? Why was it so difficult to decompose them?
3. How do decomposed requirements influence the system interfaces? How well can we predict the system interfaces from the decomposable requirements?
4. Compare the types of requirements from the OASIS Product Requirements Document (PRD) to the existing requirements classification list (2.1.2.9.2 Based on the Subject of the Requirements) and see what types of requirements are missing in PRD and why.

All above four questions are intended to help answer the research question Q2-c.

5.3.2.1 Requirements Decomposition

Decomposition of a requirement in this report means using the zigzagging method in Axiomatic Design to flow a higher level requirement into lower level subsystems, and consequently generate lower level subsystem requirements. Most product level requirements in this case study were decomposed into subsystems. However, some requirements were not.

Rather than doing what Axiomatic Design did to classify the requirements into two classes—the functional requirements and the constraints—based on whether they can be decomposed using the zigzagging technique, this thesis intends to create a mapping between the conventional way of classifying requirements and the decomposability of a requirement. Summary of the requirement categories based on subjects from 2.1.2.9.2:

- Functional Requirements
- Performance Requirements
- Reliability
- Maintainability/serviceability
- Operational Environment
- Operability
- Safety
- Appearance
- Packaging
- Weight and Size
- Installation
- Upgrading, expandability/configurability
- Transportation (including storage, loading, logistics)
- Manufacturing and assembly
- Training
- Retirement, disposal
- Distribution

- Funding, resource, timing
- Cost
- Patents
- Policy and Procedure, Regulatory requirements, standards
- Reuse of components
- Design Constraints

The next part of the report discusses how various conventional types of requirements perform during the decomposition process.

5.3.2.1.1 The Decomposable Requirements

The following types of product level requirements (PRD) of the OASIS program were decomposed into subsystems:

1. Functional requirements—what the product shall do
2. Performance requirements—how well the product shall perform certain functions
3. Maintenance requirements—how the product should be designed for ease of maintenance
4. Packaging requirements—how parts of the products shall locate relative to one another spatially
5. Design constraints—introduce a pre-selected design implementation to narrow the choice of design concepts during synthesis. Reusing existing parts or subsystems is an example of introducing a design constraint.
6. Disposal / environment—how the product shall be designed to be environmental friendly after the life of the product is finished
7. Operational environment—the environment that the product has to be in during normal operation
8. Expandability—the capability of the product to take additional optional function if the customer choose so
9. Installation—the ease of installation
10. Reliability—the reliability of the product over its service life time

11. Size—the dimension of the product

A database was constructed to record how each product level requirement was decomposed into existing subsystems. The following fields were also filled out in the database for each product level requirement (PRD):

- During the decomposition, did this PRD introduce one or more DP's in the Design Matrix? Note that the scope of this research limits the DP in the Design Matrix generated in this case study to be the subsystems of the OASIS product.
- How was this PRD decomposed into subsystems? In other words, what was the process of taking this particular PRD and using the zigzagging process to generate subsystem requirements?
- Did this PRD induce interaction between the DP's (subsystems)?
- Did this PRD introduce one or more DP's in the subsystems it was decomposed into? What type of DP did the requirement introduce in the relevant subsystems? Note according to Axiomatic Design, a DP could be physical components or design features such as dimension or material.

Table 5-3 records the answer to each of these questions for each of the above types of requirements. The purpose of this exercise is to identify how different types of requirements might act differently in the decomposition process. Note that the answers in Table 5-3 are only based on this particular case study. They may not be universal to all cases.

Requirements Types	Does the requirement introduce one or more new subsystems?	How is this requirement decomposed into subsystems?	Does this requirement induce interactions between the subsystems?	What kind of DP does this requirement introduce in subsystems?
Functional	Sometimes	Allocate to each related subsystems	Sometimes yes sometimes no	Physical component
Performance	No	• Direct allocation • Models and simulation to allocate • Budgeting	Sometimes yes sometimes no	Physical component and design feature
Maintenance	No	Allocate to each related subsystems	Yes	Physical component
Packaging	No	Allocate to each related subsystems	Yes	Design feature
Constraints	No	Allocate to each related subsystems	No	Physical Component, Design feature, or nothing
Disposal	No	Allocate to each related subsystems	No	None
Operational Environment	No	Allocate to each related subsystems	Yes	Physical Component
Expandability	No	Allocate to each related subsystems	No	Design feature
Installation	No	Allocate to each related subsystems	Unclear	Unclear
Reliability	No	Budget to each subsystem	Hard to predict	Hard to predict
Size and Weight	No	Budget to each subsystem	Hard to predict	Hard to predict

Allocation—decompose the requirements to relevant subsystems. The decomposition can be qualitative, such as assigning high-level functions to lower level subsystems. The decomposition can also be quantitative, such as using models to identify the performance requirement values for the subsystems. This definition includes both the Allocation and Flowdown definition in Grady (1993 p. 104).

Budgeting—decomposing the requirements by assigning target values to subsystems. The assignment of the values does not have to be based on strong scientific reasoning. These target values can be changed and traded later on if they are not achieved. One example is the reliability requirement budgeting process.

Note the answers in this table are only based on this particular case study at JNJ.

Table 5-3: How Various Types of Requirements were Decomposed

The rest of this section details the reasons behind the answer in each of the cells in Table 5-3. Note that many OASIS product level requirements are referenced below. The reader can find the original values in the requirements in Appendix A have been changed to protect the proprietary data.

Functional Requirements

- **Does the requirement introduce one or more subsystems?**

The functional requirements may or may not introduce new subsystems in the system level DM. The subsystems in the DM are defined according to the architecture definition. Some of the functional requirements are very close to the architecture definitions, and therefore can be seen as introducing that particular subsystem in the DM (e.g. PRD 139). Some of the functional requirements mention particular subsystems, and hence can be seen only as more detailed requirements for that particular subsystem (e.g. PRD 6). Some of the functional requirements need several subsystems to accomplish. In this case, the functional requirement is allocated into related subsystems and become subsystem requirements rather than introducing new subsystems at the system level (e.g. PRD 6).

In the OASIS program, the functional requirements in PRD were generated after the architecture of the product was already chosen. Therefore, the functional requirements in the PRD are less likely to introduce new subsystems, but more likely to flow into existing subsystems. This may not be the case for other products.

- **How is this requirement decomposed into subsystems?**

Functional requirements are allocated into related subsystems. The allocation process is usually to qualitatively assign certain requirements to relevant subsystems. Functional requirements may be allocated to a single subsystem or multiple subsystems.

- **Does this requirement induce interactions between the subsystems?**

If the functional requirement flows into only one subsystem, of course no system level interaction is introduced (e.g. PRD 77). If the functional requirement flows into multiple subsystems, it is still possible that no system level interaction is introduced because the system level completion of the function may be the mere sum of the completion of its sub-functions in related subsystems (e.g. PRD 120). The last case is that the functional requirements flow into multiple subsystems and these subsystems have to interact with each other in order to fulfill the requirement (e.g. PRD 6). The last case is the most common, because this is the reason for systems to exist—the whole is more than the mere sum due to the interactions among the parts of the system.

- **What kind of DP does this requirement introduce in the relevant subsystems?**

The functional requirements are always decomposed into lower level functional requirements in the subsystems. Therefore, the new DP's introduced in the relevant subsystems are physical components that can fulfill certain functions.

Performance Requirements

- **Does the requirement introduce one or more subsystems?**

The performance requirements do not introduce any new subsystems. The reason is that the performance requirements are about how well certain functions are performed by the product. Therefore, the performance requirements are always associated with existing subsystems instead of creating new subsystems to fulfill the requirement.

- **How is this requirement decomposed into subsystems?**

The performance requirement decomposition takes three possible routes:

1. The requirement can be directly flown into subsystems (e.g. PRD 21). Then it becomes the performance requirement only those particular subsystems have to meet.
2. The requirement is allocated to subsystems using model simulation and maybe robust design studies to identify the subsystems critical to this particular performance (e.g. PRD 16). Then the requirement is flown into these relevant subsystems.

3. The requirement is budgeted among relevant subsystems. The budgeting process sets design goals for the subsystems on the particular performance, but whether the goal is realistic is not guaranteed (e.g. PRD 51).

- **Does this requirement induce interactions between the subsystems?**

The performance requirement may or may not add interactions among the subsystems. When the requirement is only decomposed to one subsystem, obviously no interaction is added (e.g. PRD 134). When the requirement is flown into multiple subsystems in parallel, i.e. the requirement is not decomposed, so no system interactions are introduced (e.g. PRD 21). Nevertheless, many performance requirements are decomposed into multiple subsystems that introduce system level interactions (e.g. PRD 16).

- **What kind of DP does this requirement introduce in the relevant subsystems?**

According to Axiomatic Design, DP can be physical components in the system or design features such as dimension and material selection. The performance requirements are capable of generating both types of DP's.

Sometimes, the system level performance requirement introduces functional requirement at the subsystem level. An example is PRD 95. In this case, the performance requirement introduces physical component in the subsystem level. In other cases, the performance requirement only decomposes into performance requirements in the subsystems. Then the DP's generated from this particular system level performance requirements are design features. An example is PRD4.

Maintenance Requirements

- **Does this PRD introduce one or more new subsystems?**

Maintenance requirements for this product do not introduce new subsystems because they are about how to maintain the existing subsystems (PRD 256). However, it is easy to imagine cases where new subsystems are introduced just for the purpose of maintenance, such as a access door or a diagnostic sensor.

- **How is this PRD decomposed into subsystems?**

The decomposition of the maintenance requirement is very similar to that of the functional requirements. The requirement is allocated into the subsystems that are being maintained, and the subsystems that support the maintenance activities.

- **Does this PRD induce interaction between the subsystems?**

In this case study, the maintenance requirements always added system level interactions among the subsystems. Since the decomposition of the maintenance requirements are very similar to that of the functional requirements, it is possible in general that there will be cases where the maintenance requirements do not introduce system level interactions.

- **What kind of DP does this requirement introduce in the relevant subsystems?**

In this case study, the maintenance requirements always introduced physical components in the subsystems just as the functional requirements do.

Packaging Requirements

- **Does this PRD introduce one or more new subsystems?**

The packaging requirements do not introduce new subsystems, because only when subsystems exist, could one talk about their spatial relationship (PRD 187).

- **How is this PRD decomposed into subsystems?**

The packaging requirements are allocated into the existing relevant subsystems. When there is detailed tolerance information about an assembly, the packaging information may be budgeted to relevant subsystems. However, in this case study, no such example was found in the product level requirements document.

- **Does this PRD induce interaction between the subsystems?**

If the packaging requirements are at the product level, it always causes interactions among the subsystems.

- **What kind of DP does this requirement introduce in the relevant subsystems?**

In this case study, the packaging requirements do not add physical components in the subsystems. However, the packaging requirements may cause one or more design features in the subsystems to be critical. Potentially, the packaging requirements could cause special physical components to be designed in the subsystems.

Constraints

The constraints are defined here as the product requirements that restrict the choices of design implementations.

- **Does this PRD introduce one or more new subsystems?**

The design constraints do not introduce new subsystems because these PRD's restrict the choice design concept during synthesis. In this case study, the design choices can be restricted in three ways:

1. Fixed external interfaces (e.g. PRD 43, 44)
2. Reuse existing/known/tested parts or technology (e.g. PRD 34)
3. Specify what to do in a design (e.g. PRD 54)

- **How is this PRD decomposed into subsystems?**

The constraints are allocated directly to relevant subsystems. They will not show up as FR's in the DM, but rather become the rationale behind the choices of a design concept and the corresponding DP's.

- **Does this PRD induce interaction between the subsystems?**

The constraints in this case study do not directly add system level interactions because they specify the implementation choices on particular components or subsystems. However, a certain choice on the DP set by the constraint may cause the interactions between other DP's in the system. In Table 5-3, only the direct effect of the constraints was listed.

- **What kind of DP does this requirement introduce in the relevant subsystems?**

The constraints do not necessarily introduce DP's in the subsystems. For example, PRD 43 adds design features in the subsystem. PRD 44 adds physical components in the subsystem. On the other hand, PRD123 specifies the values for DP's without adding anything new.

Disposal Requirements

In the OASIS PRD, the disposal requirements are about the packaging materials rather than the analyzer. Therefore, these requirements are only flown down into the directly related subsystems without introducing subsystem interactions. The author thinks that in general, this type of requirements can be similar to the case of maintenance requirements.

Operational Environment

- **Does this PRD introduce one or more new subsystems?**

In this case study, the operational environment requirements do not introduce new subsystems. The reason is that the environment requirements are something that every designed subsystem has to be able to withstand. In the OASIS analyzer, no subsystems are designed solely for the operating environment. In general cases, it is possible that special subsystems are designed just for the environment such as that in the spacecraft.

- **How is this PRD decomposed into subsystems?**

The operational environment requirements are flown into any subsystems that are relevant.

- **Does this PRD induce interaction between the subsystems?**

Most of the time, this type of requirements does not introduce system level interaction. The decomposition is equal and parallel in each subsystem (e.g. PRD 46). However, when an operational environment has to be taken by several subsystems with tradeoffs, system level interactions exist too (e.g. PRD 49).

- **What kind of DP does this requirement introduce in the relevant subsystems?**

This type of requirements may introduce new physical components and DP's in the subsystem for the purpose of ensuring the product to work in the specified environment (PRD49).

Expandability

The expandability requirement is like the functional requirement for the future of the product.

- **Does this PRD introduce one or more new subsystems?**

The expandability requirement does not introduce new subsystems in this case study. It is about the design of the existing subsystems. However, like the functional requirements, it could potentially introduce new subsystems.

- **How is this PRD decomposed into subsystems?**

The expandability requirement is directly allocated into relevant subsystems.

- **Does this PRD induce interaction between the subsystems?**

The expandability requirement in this case study does not introduce interactions among subsystems. However, like the functional requirements, it could potentially introduce system interactions if more than one subsystem is involved in achieving an expandability requirement.

- **What kind of DP does this requirement introduce in the relevant subsystems?**

In this case study, the expandability requirements cause certain design features to be important, but do not introduce new physical elements (PRD 146).

Installation

- **Does this PRD introduce one or more new subsystems?**

In this case study, there is only one installation requirement (PRD 287). It did not introduce any new subsystem. However, the ease of installation could potentially require special subsystems and therefore introduce new subsystems in the product.

- **How is this PRD decomposed into subsystems?**

It is allocated to relevant subsystems.

- **Does this PRD induce interaction between the subsystems?**

The installation requirement in this PRD is very high level, and it is unclear how this requirement may affect the subsystem interfaces. The system interactions will be discovered by the service experts through examining the prototype. An expert engineer may be able to suggest some of the possible system interactions up front before detailed design, but the suggestion may not be completely correct and accurate, and highly depend on whether there is a knowledgeable expert on the team. Therefore, the answer to this question in this case study is unclear.

- **What kind of DP does this requirement introduce in the relevant subsystems?**

This particular installation requirement does not add any new DP in the subsystem. But it could in general if we start to consider alignment pin, etc.

Reliability

The reliability requirements (PRD 58) are decomposed into each subsystem as a target requirement. The decomposition process is called budgeting where targets for each relevant subsystem are set and traded off. Whether the target can be achieved and whether there are design tradeoffs among the subsystems are not known until the prototype and testing starts. The decomposition of this type of requirements is more like setting goals rather than requirements.

Size and Weight

The weight and size requirements (e.g. PRD48) for this product are of minor importance, unlike that for aircrafts. The contribution of each subsystem is monitored, but not really traded off. How well these requirements are met is unknown until the details of the design are determined.

5.3.2.1.2 Requirements that are Difficult to Decompose

As shown in Figure 5-30, 8% of the product level requirements are not decomposed. These requirements are examined and the reasons for which they are not decomposed are discussed below.

Standards

The standards in the OASIS PRD are very general (e.g. PRD 280). To understand the impact of the standards on the design, once must first obtain a good understanding of the standards, and then examine whether the form that the design takes meets the standards. The common practice at JNJ OCD is to have experts on the standards to come to design review and check whether the design meets standards. Therefore, decomposing the standards before a detailed design is a difficult task.

Performance

Certain performance requirements are written at very high level, and hence are difficult to relate to specific subsystems (e.g. PRD 1). These performance requirements can only be validated after the entire system is put together.

Shipping

There is only one shipping requirement (e.g. PRD57) and it is at a very high level. It is very difficult to see the immediate impact of this requirement on the design until the design is detailed into specific forms.

Safety

Safety requirements seem to touch everything, and cannot be decomposed until the design comes into form. It is another requirement like the standards that are checked by the experts in the design reviews (e.g. PRD 269).

Summary on the Reasons for Requirements to be Hard to Decompose

Requirements can be difficult to decompose for the following two reasons:

- The requirement is written at very high level, and cannot easily be related to subsystems, such as some of the performance requirements and the standards.
- The requirements cannot be decomposed until the details of the form that a design is taking are determined. Such requirements include the shipping and the safety requirements.

5.3.2.1.3 Missing Classes of Requirements

Compare the requirements types in the OASIS Product Requirements Document to the standard list of the requirements types (see 2.1.2.9.2), the following types of requirements are missing from the OASIS Product Requirements Document:

- Cost
- Appearance
- Distribution
- Design for Manufacturing, Assembly, and Serviceability (DFMAS)
- Operability
- Training
- Budget and Timing
- Patents
- Component Reuse

The cost, appearance, and distribution requirements are in the marketing report. They are not in the Product Requirements Document because the Product Requirements Document is considered to be an engineering requirements document.

The DFMAS, the operability, and training requirements are in the design guideline document, which is a separate document from the requirements document. Since DFMAS

and operability are difficult to measure with hard scales, they are listed as design guidelines rather than requirements, because requirements in JNJ are to be inspected by the FDA.

The budget and timing requirements are management issues and hence are not included in the design requirements document. The managers are expected to plan and keep track of the team's performance in the design process.

The requirement of not violating patent law and to reuse component is implicit. They are effective during design concept selection.

Hence, the inputs to a design are more than just the verifiable engineering requirements. When we use the matrix conversion method to predict system interactions, we must be sure not to overlook any other document sources.

5.3.2.1.4 Summary on Requirements Decomposition

The goals of studying the various types of requirements and how they flow down into subsystems are:

1. Requirement decomposition is important because it provides trace-ability of the design goal through the system hierarchy. This study is interested in understanding how each type of requirements shall be dealt with during decomposition. Can all requirements be decomposed? Whether the decomposition of requirements guarantee the success of the system integration.

2. Axiomatic Design claims that if the Functional Requirements can be decomposed into the system in a de-coupled or uncoupled manner, then the design is at the optimal. It further claims that all systems can find this optimal design. This research intends to find out whether all requirements for a system can be decomposed in the Axiomatic Design's manner. Whether the decomposition can correctly predict the system interactions, and

hence claim an ideal design based on the axiomatic Design standards is really an ideal design.

From the analysis so far, we can make the comparison in Table 5-4. The following observations can be easily made:

- If a requirement can be decomposed like the functional requirement, it can be used to predict system interactions without knowing very much design details.
- If a requirement cannot be easily decomposed into the hierarchy of the system, then it cannot be used to predict system interactions using the matrix conversion method.
- Many requirements cannot be decomposed as the functional requirements in the Axiomatic Design method, but they can be decomposed in other ways and help to predict system interactions.
- On the other hand, although some requirements such as reliability can be decomposed using budgeting method, they cannot predict system interactions very well.

There exist many types of requirements that cannot be easily decomposed, and cannot predict system interactions at early stage of the design.

	Can predict system interactions	**Cannot predict system interactions**
Can be decomposed in the same way as the FR's in the Axiomatic Design	Functional Maintainability Operational Environment Expandability Appearance	None
Can be decomposed but not in the same way as decomposing the FR's in the Axiomatic Design	Performance (Modeling) Packaging (DFC, DSM) Design Constraints (DSM)	Reliability (budgeting) Size (budgeting) Weight (budgeting) Cost (budgeting)
Difficult to decompose	None	Installation Standards Safety DFMAS Component Reuse Operability Shipping
No strong evidence in this case study	Disposal Distribution Training Budget and Timing Patents	

Table 5-4: Requirements Decomposition Summary

Therefore, the answers to the above two questions are:

1. Although we hope to be able to decompose every requirement to ensure the design intent is met, some requirements cannot be easily decomposed. Therefore, unanticipated problems will always exist during prototyping, integration, and testing. This is a proof for the emergent properties of the system in product design. Understanding which requirements are decomposed and which ones are not is the key to the success of system integration.

2. The requirement decomposition method in axiomatic design only covers part of the requirements for products. If we were to use Axiomatic Design's Design Matrix (DM) to predict system interactions, based on Table 5-4, only interactions caused by the "Functional", "Maintainability", "Operational Environment", "Expandability", and "Appearance" requirements can be captured. That's total 108 interactions out of possible 247 interactions (see Figure 5-17). Therefore, we cannot judge whether a system is uncoupled or decoupled only based on the decomposable requirements and the DM. There are more causes of system interactions.

From the above discussion, it is clear that the matrix conversion method the author has cannot predict all of the system interactions that will happen. Some interactions will always be missing. However, this limitation in the method may not be so bad if the prediction can give the correct overall picture. Earlier in this report, it has been shown that the matrix conversion method gave a close approximation of what the experts can predict. After all, we are looking for a prediction to aid planning, and not all details of a system are necessary.

5.3.2.2 Which Types of Requirement Drive System Interfaces

In this section, we are interested in seeing which type of requirement provides the best approximation to the prediction of system interactions. The motivation for this objective is to find out how far from the truth Axiomatic Design theory is by using only Functional Requirements to judge the coupling in the system.

The interactions captured by the requirements-only DSM (Figure 5-8) and the rationales behind each interaction are examined. The types of requirements that cause each system interactions are recorded. A partial DSM is built for each requirement types and compared with the comprehensive DSM.

In Table 5-4, we can see that not all of the requirements decomposed introduce interactions among the subsystems. Only 5 types of requirements introduced system level interactions in the OASIS analyzer. Figure 5-19 shows the percentage of system interactions they

introduced in the final DSM. Table 5-5 shows the amount of interactions these requirements predicted among all interactions derived from requirements.

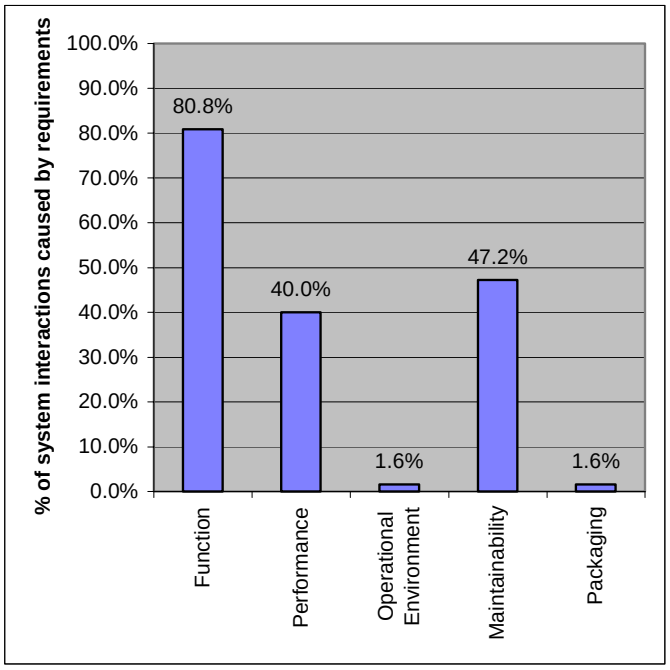


Figure 5-19: Contribution to System Interactions from Various Types of Requirements

	Amount of Interactions	Percent of Total Interactions
Functional Requirements	101	80.8%
Performance Requirements	50	40%
Operational Environment	2	1.6%
Maintainability	59	47.2%
Packaging	2	1.6%
Total Interactions from Requirements	125	------

Note the values and percentage in this table do not add up to 100% because multiple types of requirements could identify the same interaction mark in the DSM.

Table 5-5: Contribution of Each Type of Requirements to the System Interactions

In this case study, functional, performance, and maintenance requirements contribute the most to the system interactions predicted. This list also matches with the result in Table 5-4, where the above types of requirements are listed as decomposable and capable of predicting system interactions. At this point, we still cannot say any of the four above types of information do a good job at predicting the system interactions. How well the system interactions are predicted is measured not only by the quantity of the DSM marks covered, but also by how well the system topology (see Section 5.3.1.5) is predicted. The following 5 figures are intended to compare the system topology predicted by the top four drivers of the system interactions.

Figure 5-20 shows the final DSM using all types of requirements. This figure is the same as Figure 5-8. Figure 5-21, Figure 5-22, and Figure 5-23 show the partial DSM if we only took inputs from functional, maintainability, and performance requirements.

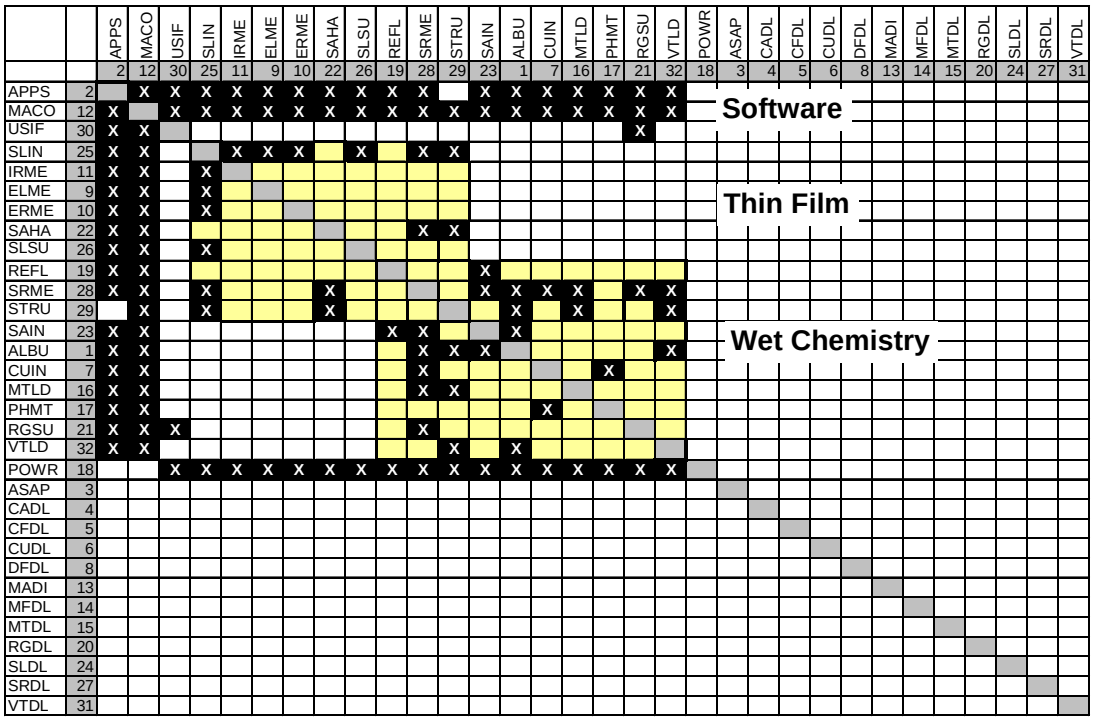


Figure 5-20: The Final DSM Derived from All Types of Requirements

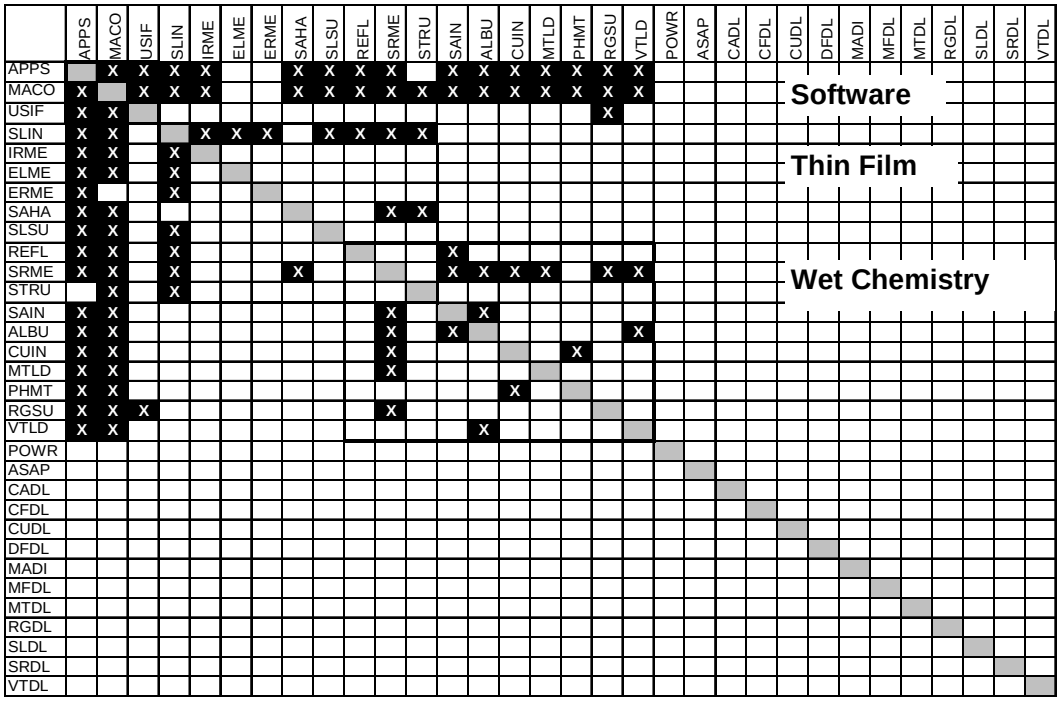


Figure 5-21: The DSM Due to Functional Requirements

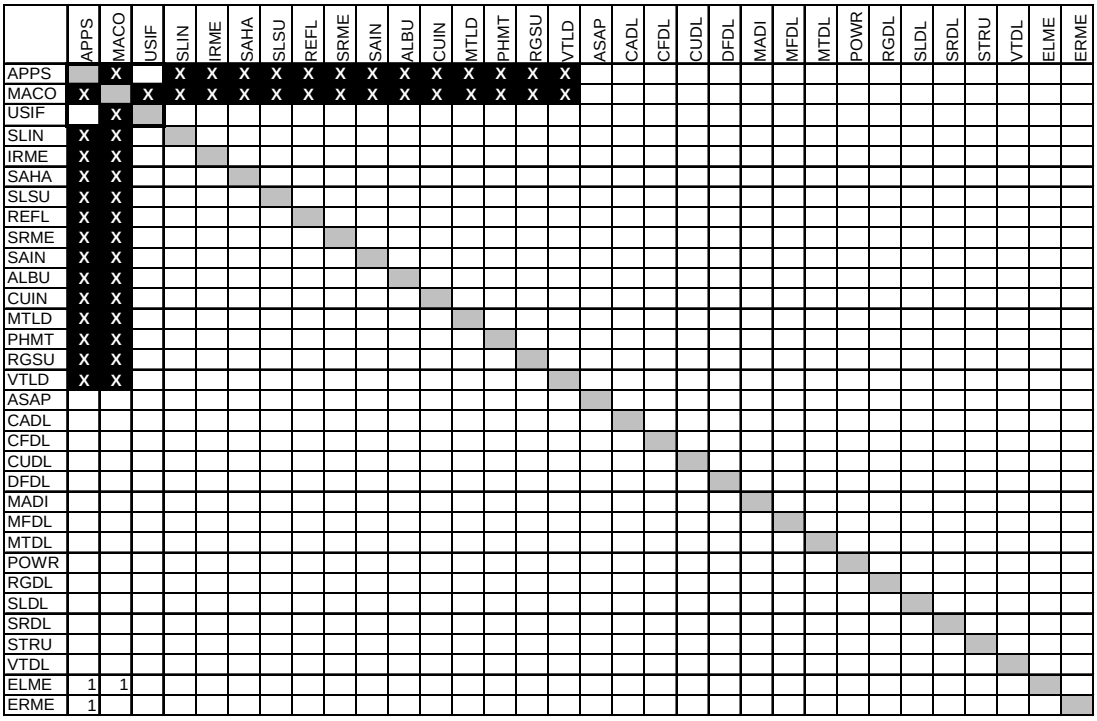


Figure 5-22: DSM Due to Maintainability Requirements

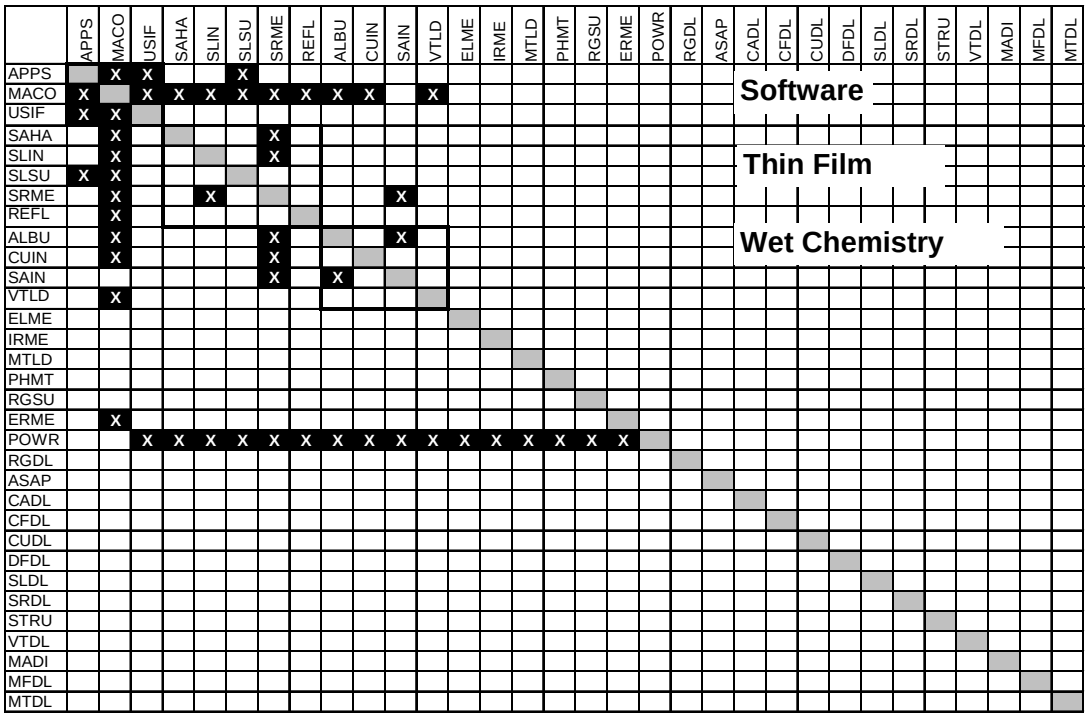


Figure 5-23: DSM Due to Performance Requirements

Functional Requirements

Compare Figure 5-20 and Figure 5-21, the two DSM has the same topology after partitioning, although there are fewer marks in the DSM due to functional inputs. Therefore, we can conclude the functional requirements provide good prediction to the system interactions.

Maintainability Requirements

Compare Figure 5-20 and Figure 5-22, we can see that the maintainability requirements, although provide a good percentage of marks in the DSM, only provide a small part of the total picture of the system. Only the prediction between the hardware subsystems, and the computer software and controls are predicted. There should also be a lot of interactions among the hardware subsystems where maintenance and service issues are important. Yet these interactions were not reflected in the DSM in Figure 5-22 at all. The reasons are as follows.

First, maintainability issues among hardware subsystems cannot easily be predicted until the design of the subsystems is complete and the system is integrated. This type of maintainability interaction is the same as the serviceability requirements in its capability of predicting system interactions (see Table 5-4). Second, the maintainability requirements between the software and the hardware are similar to additional functions the two subsystems have to provide together. Therefore, the interactions between the hardware and software are easily captured early on.

Therefore, the maintainability requirements can be decomposed when the requirement can be translated into certain functions the system has to provide. Consequently, the decomposition process can provide prediction of the system interactions. Yet, a lot of the interaction due to maintainability cannot be seen at early stage of the design as additional functions. These interactions cannot be predicted easily.

Just as a note, placing the maintainability requirements in the same location as the functional requirements in Table 5-4 is okay because even for the functional requirements, we cannot

predict everything. The unpredictable part about the maintainability can be seen as in the Design for Serviceability (DFS) in Table 5-4.

Performance Requirements

Compare Figure 5-20 and Figure 5-23, the DSM in Figure 5-23 captures only a small part of the entire picture. Therefore, performance requirements, although introduced a lot of the interactions in the DSM, do not provide good prediction of the system.

Summary of Prediction Capability

From the above analysis, it is clear that in this case study, the functional requirements provide most of the interactions in the final requirements DSM. The topology of the DSM built from functional requirements also match closely with the requirements DSM. From Table 5-4, we can see functional requirements and maintenance requirements are among those that can be decomposed using Axiomatic Design's DM. Therefore, Axiomatic Design's Design Matrix, although cannot capture all of the system interactions, still provides a fairly good method to capture most of the system interactions that can be predicted.

5.3.2.3 Summary on Requirements Decomposition

The third objective of this case study was to find out whether all requirements could be used to predict system interactions or not. The first part of this section showed that not all requirements could be decomposed. Among the requirements that could be decomposed, not all requirements could be decomposed in the way Axiomatic Design's Design Matrix was constructed. In addition, not all requirements that can be decomposed could be used to predict system interactions. Therefore, the Axiomatic Design's view on determining system interactions based on Design Matrix' coupling is naïve.

Yet, the second part of this section showed among all the requirements that was decomposed in this case study, the two requirements that could be decomposed using the Axiomatic Design Matrix contributed the most to the prediction of system interactions, which was a fairly good match with the expert's DSM (see 5.3.1). Therefore, the Design Matrix, although

does not provide a complete view of the system, is still very useful in helping planning for system integration at early phase of the design process.

In addition, Table 5-4 further showed the matrix transformation method explained in Chapter 3 was not the only way to get system interactions from requirements. Some requirements cannot be decomposed using the Axiomatic Design's DM, but their influence on system interactions can still be reflected in a DSM. Since the DSM and the DM have the same look except for the row headings (proof see 3.2.4 The Choice of Output Variables), the influence of these requirements could be reflected back in the DM without having to decompose them using the zigzagging method.

5.3.3 The Sources of System Level Knowledge

In the above discussion, the comparison between the DSM derived from the requirements and the DSM made by JNJ OCD experts was made. Nonetheless, this case study used more than just the requirements document to construct the prediction DSM. Total five types of system engineering documents were used:

1. Architecture Definition document
2. Function Flow Diagram
3. Mechanical Interface Diagram
4. Requirements document
5. Hazard Analysis

This section is interested in understanding how each types of information source contributes to our understanding of the system interfaces.

5.3.3.1 Which Document Tells the Most about System Interactions

Figure 5-24 through Figure 5-28 show the system interfaces in the DSM identified by each individual sources of information.

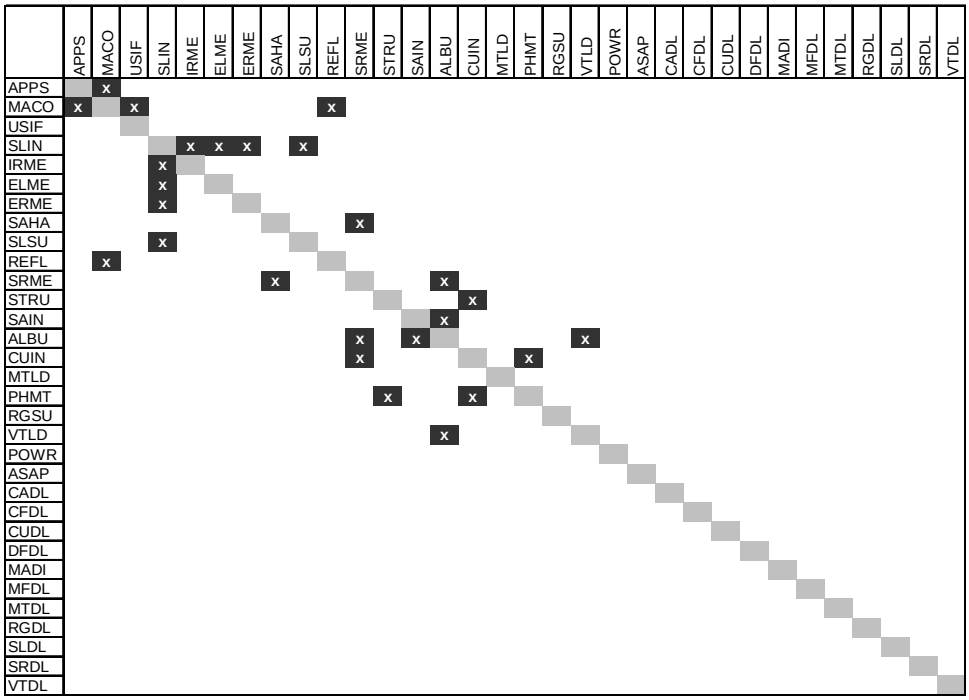


Figure 5-24: System Interactions Obtained from Architecture Definition

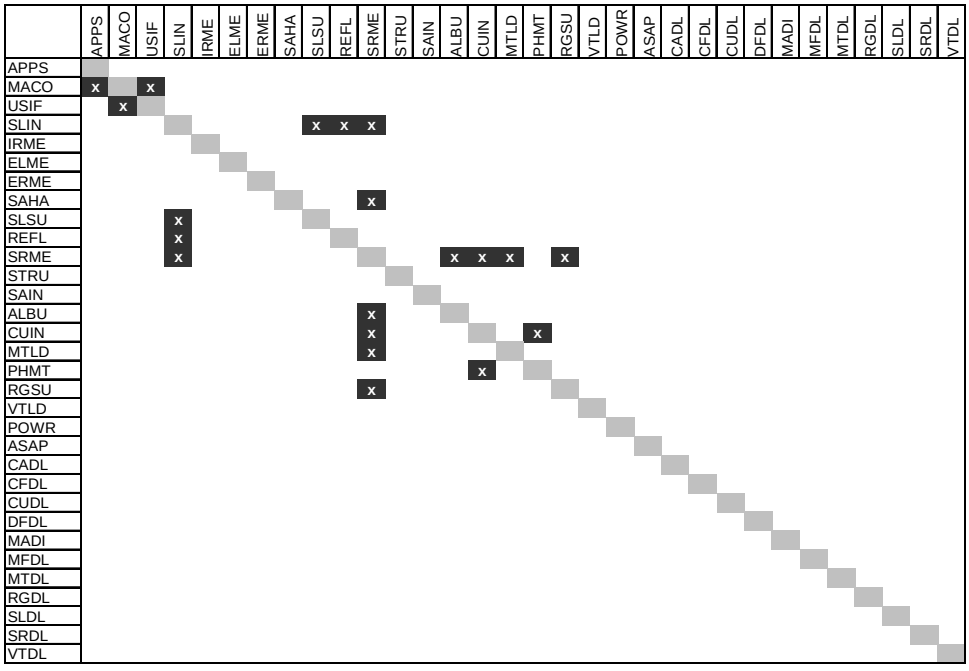


Figure 5-25: System Interactions Obtained from Function Flow Diagram

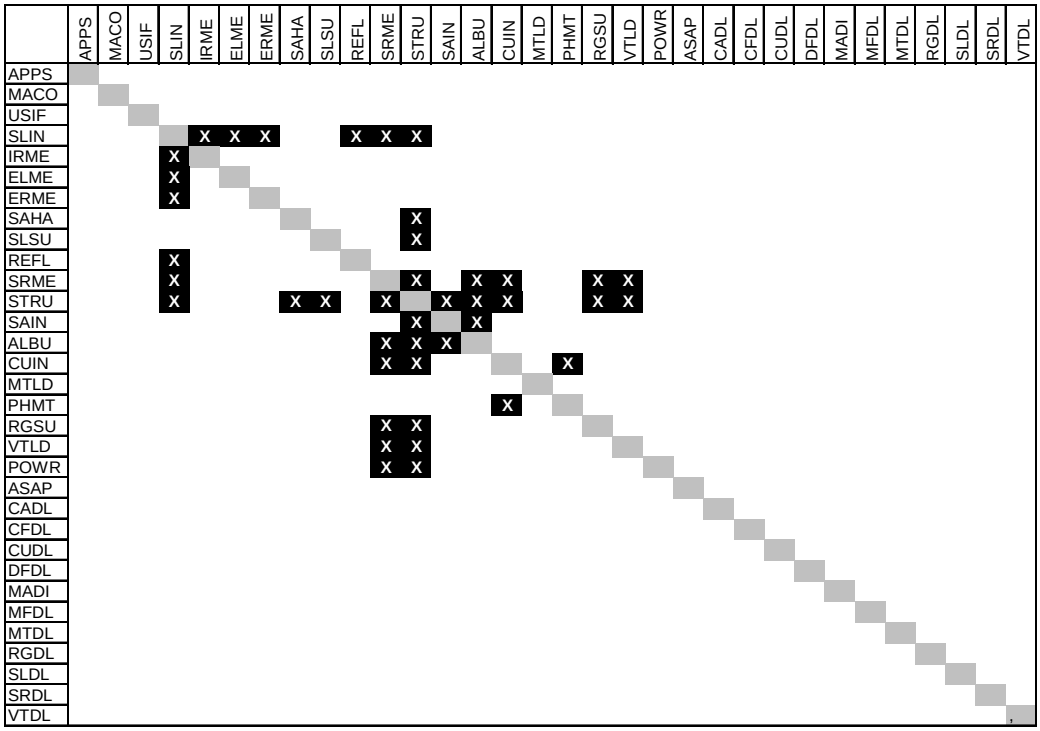


Figure 5-26: System Interaction Obtained from Mechanical Interface Document

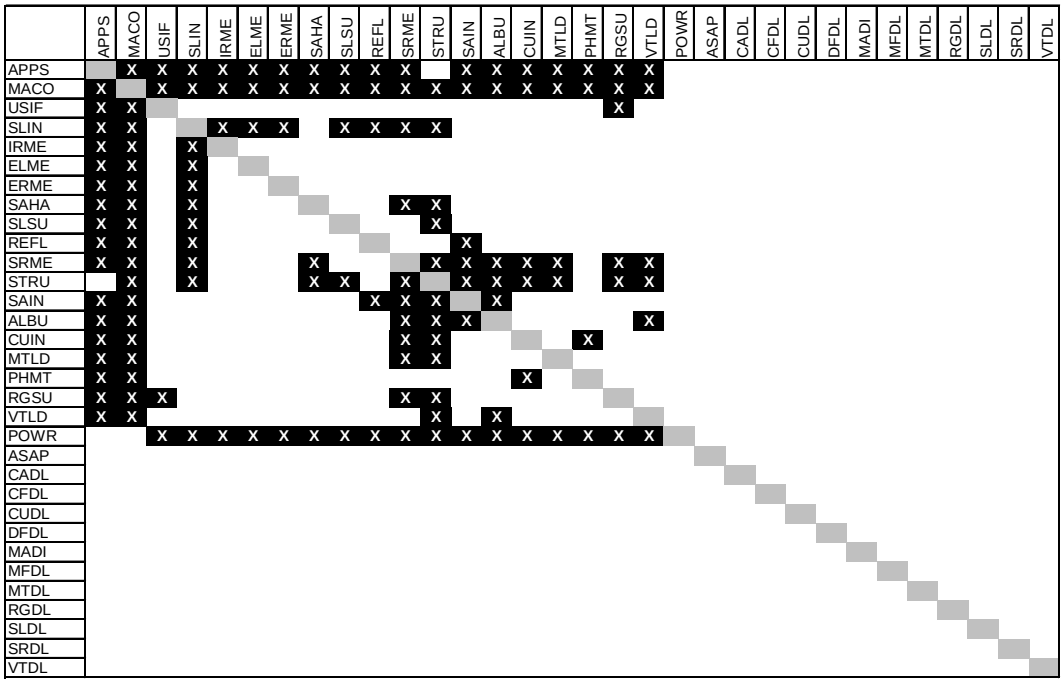


Figure 5-27: System Interactions Obtained from Requirements Document (Same as Figure 5-14)

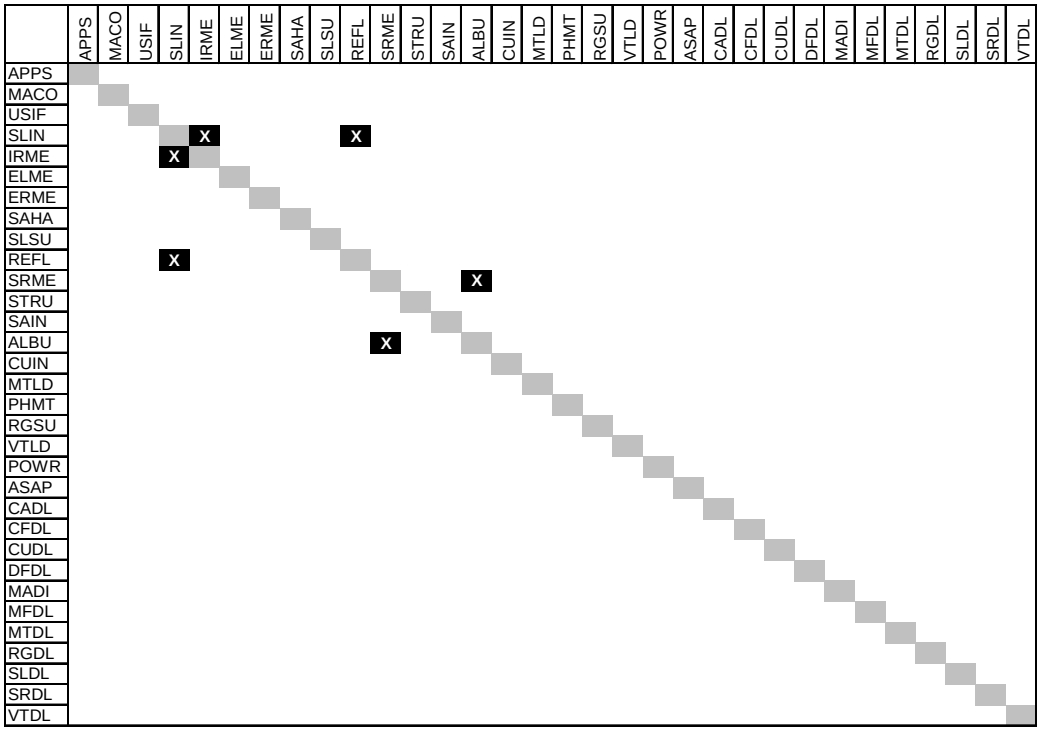


Figure 5-28: System Interactions Obtained from Hazard Analysis Document

To measure the capability of capturing system interactions each of the above 5 information sources has, the amount of interactions each information sources captures is measured. In addition, the interaction density is measured for each information source. Again as a reminder:

Interaction Density Ratio = Number of Non-diagonal Marks in the DSM / Number of Rows

In order to compute the *Interaction Density* to reflect the reality, the following subsystems are not considered: CUDL, MTDL, RGDL, SLDL, VTDL, ASAP, CADL, CFDL, DFDL, MADI, MFDL, SRDL. The reasons are:

1. Most of these subsystems (ending with DL and ASAP) are not part of the analyzer. This DSM study wants to concentrate on the subsystem interactions within the analyzer.

2. These subsystems do not have SSRD written for them yet. Therefore, tracing the PRD decomposition not done. Many of these subsystems also do not have a person fully responsible for them. When building the DM and DSM in step 1-5, the author ignored these subsystems.
3. These subsystems do not have any non-diagonal interaction marks in the DSM's in Figure 5-24 to Figure 5-29. Therefore, counting them in the *Interaction Density* is not fair.

Therefore, the denominator used to compute the ratio is 20 rather than 32. Table 5-6 contains the comparison of the five information sources for identifying system interactions. Note the sum of the number of interactions introduced by each document source is greater than the total number of interactions in the final DSM, because some of the interactions can be identified by multiple information sources.

DSM	**Source of Information**	**Number of Interactions**	**Average Marks per Row**
Figure 5-24	Architecture Definition	25	1.25
Figure 5-25	Functional Flow Diagram	20	1
Figure 5-26	Mechanical Interface Document	41	2.05
Figure 5-27	Product Requirements and Subsystem Requirements	127	6.35
Figure 5-28	Hazard Analysis	6	0.3
	Total	141	7.05

Table 5-6: Contribution of Each Information Source in Identifying Subsystem Interactions

The percentages of contribution of each information source in identifying the system interfaces are shown in Figure 5-29. The percentage values are calculated using the following formula:

Percent Contribution = Number of Interactions Introduced / Total Interactions Identified by all Documents

For example:

Percent Contribution of Architecture Definition document = 25 / 141 = 17.7%

Again, note the same system interaction can be identified by multiple information sources. Therefore, the percentage values in do not add up to 100%.

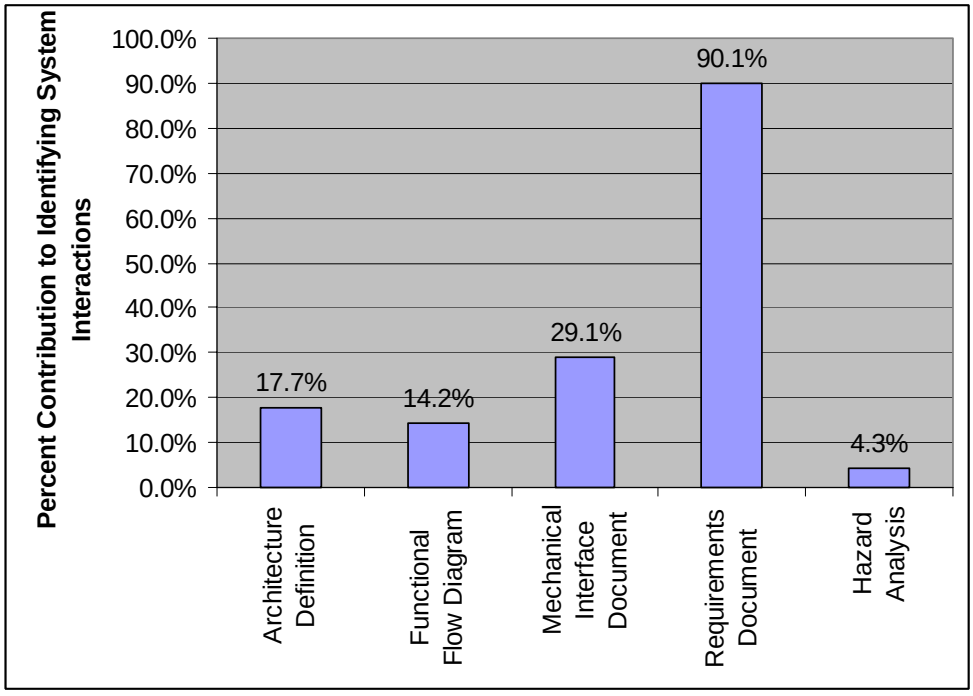


Figure 5-29: Percent Contribution to Identifying Subsystem Interactions from Each Document Source

From Figure 5-29, we can see the requirement documents (PRD and SSRD) predict the majority of the system interactions, while other traditional types of system engineering documents capture less system interface knowledge.

Since some of the interactions are identified by multiple documents, Table 5-7 shows the intersections of the contribution of each document. Note type 4 documents in Table 5-7—the requirements document—identified 85 interactions that other documents were unable to capture. In other words, the requirements document contributed additional 60% more interactions that other traditional types of system engineering documents were unable to

capture. Meanwhile, the requirements document (type 4 information source in Table 5-7) also captured 42 system interactions, which at least one other document also predicted. Only 14 out of the total 141 interactions in Table 5-7 cannot be predicted by requirements document, all of which are related to spatial relationships and can be predicted by information source type 3—the Mechanical Interface Document. Therefore, we can conclude that with good and complete requirements document enables us to predict and capture most of the majority of the system level knowledge using the matrix conversion method. In addition, the above comparison also suggests that the benefit of documenting and managing requirements can be much greater if the matrix conversion method is employed to predict system interactions.

Interaction Identified by Information Source(s)	Number of Interactions
3	10
4	85
1,3	2
1,4	2
2,4	4
3,4	10
1,2,4	4
1,2,3,4	8
1,3,4	4
2,3,4	6
1,3,4,5	2
1,2,3,4,5	2
2,3,5	2
Total	141

1—Architecture Definition Document 2—Function Flow Diagram
3—Mechanical Interface Document 4—Requirements Document
5—Hazard Analysis

Table 5-7: Number of Interactions Identified by Single or Multiple Information Sources

5.3.3.2 How well the System Level Knowledge is Documented

The second important question to ask is that whether the system interface knowledge resides in experts' heads or in documents. In the previous research work at Ford and CVC, most of the system interface knowledge resides in people's heads (see Figure 1-3 and Figure 4-16).

This case study is interested in discovering whether the same situation about system level knowledge exists in JNJ OCD the OASIS product program.

There are two types of records need to be counted:

1. When product level requirements were decomposed into subsystem requirements, how much of the decomposition knowledge is recorded in the document (ReqisitPro software in JNJ OCD) and how much still resides in human's mind?

2. Combining all the information sources for identifying interactions, how many interactions can be identified from reading documents and how many rely on experts' knowledge?

5.3.3.2.1 PRD Decomposition Knowledge Documentation

Ideally for OASIS program, all of the subsystem requirements should be traced back to the product level requirements, and the requirements trace-ability should be documented in the RequisitPro software. However, by the end of July, not all of the product level requirements decomposition was recorded in the RequisitPro. Some subsystems were still working on their requirements and understanding the relationship between the product level requirements and the subsystem requirements. Therefore, from time to time, the author had to consult the the expert system engineers to capture how certain product level requirements were to be decomposed into which subsystem(s).

In this case study, an EXCEL database was built to record the decomposition of each product level requirement and the information source of the decomposition. This database was used to count how many of the information source was from human and how many was from documents. Some of the PRD decompositions were combined knowledge of both human and documents, with each telling a part of the story. In such cases, the decomposition was counted still as relying on human knowledge, because without human knowledge, the

decomposition would not have been complete. However, the estimate is conservative because the contribution of the knowledge in the document is overlooked.

Total Number of PRD: 290

PRD not decomposed: 23 (reasons are explained later in this report)

PRD that I was told to be obsolete: 10

PRD actually decomposed: 257

PRD decomposed based solely on documents: 26

PRD decomposed based on people's knowledge: 231

Percent of decomposition relying on documents: 26/280 = 9%

Percent of decomposition relying on human knowledge: 231/280 = 83%

Percent of requirements not decomposed: 23/280 = 8%

Figure 5-30 summarizes this situation.

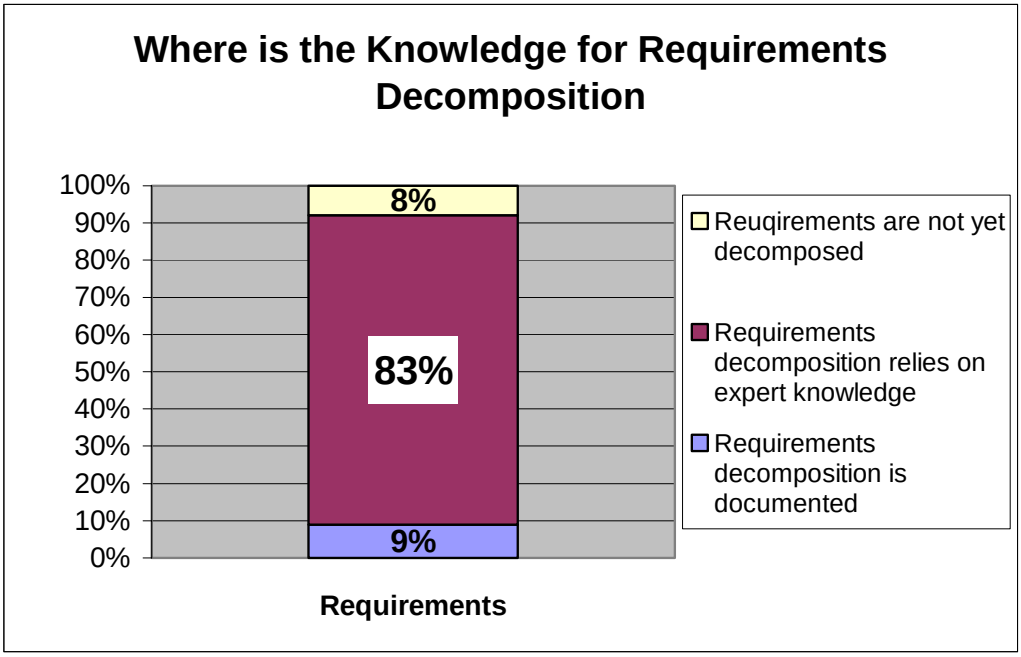


Figure 5-30: Information Sources for PRD Decomposition

Figure 5-30 shows that by the end of July, only a small percent of the product requirements decomposition was recorded in the requirements documentation tool used at OCD—the RequisitPro. Therefore, the OASIS program engineers need to do a better job in documenting the trace-ability of requirements. The documentation tool (RequisitPro) is available for use. It is just a matter of taking the time to do it. Once the trace-ability of the requirements is recorded in RequisitPro, JNJ OCD can have a much higher percentage in documented system interaction knowledge (Figure 5-31).

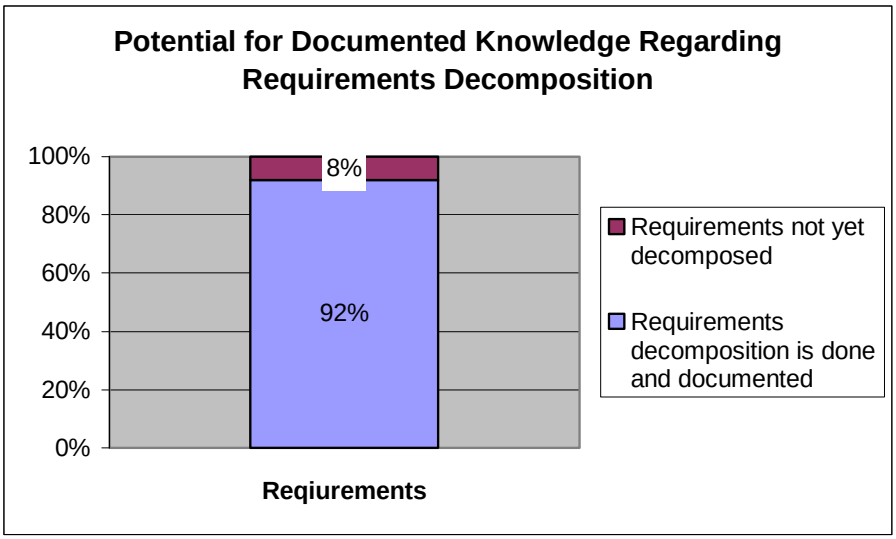


Figure 5-31: Potential Improvement on Documented Requirements Decomposition

5.3.3.2.2 Where is the System Interaction Knowledge

In order to find out how much knowledge regarding system interactions is documented, we may take into account all of the available system engineering design documents. Therefore, besides the requirements documents, the architecture definition document, the function flow diagram, the mechanical interface document, and the hazard and mitigation document are all considered. The DSM built from all systems engineering documents is shown in Figure 5-7. This DSM has 12 additional marks compared with the DSM built from requirements. However, the expert DSM (Figure 5-11) captures all of the 12 additional marks in Figure 5-7. Therefore, although here we consider the DSM built using all documents rather than

requirements only, the total number of system interactions identified by both the experts and the documents are still 247.

When the DSM was built using systems engineering documents, an EXCEL database was constructed to record the rationale behind each interactions captured in the DSM. The sources of the interactions were also recorded. An interaction could have multiple information sources, all of which are recorded. To determine which interaction was found in documents and which were from human knowledge, the rules of counting are as follows:

1. If there is a non-requirements document (such as the architecture definition, the Mechanical Interface document, etc) used as the source of the interaction, the interaction is counted as residing in documents.

2. If there is a product level requirement that did not depend on human inputs to decompose into subsystems, the DSM interaction using this PRD as an input is counted as depending only on documents.

The results of counting are:

Total number of interactions in the final DSM: 247
Interactions captured by documents only: 39
Interactions relied on human knowledge: 208

Percent of Interaction from Human: 208/247 =84 %
Percent of Interaction from Documents: 39/141 = 16%

If the OASIS engineers were to document all of the knowledge about requirements decomposition, then:

Total number of interactions captured = 247

Number of interactions captured by documentation if the requirements decomposition were properly documented = 141 (all marks in Figure 6 will be from documents)
Percentage of system interaction knowledge documented as a result = 141/247 = 57%

The number 141 is calculated as follows based on the values in Figure 5-16:

- Type 1 (hardware software interactions) and Type 6 (miscellaneous interactions experts missed) will be documented in the requirements. That's total 69+6 = 75 additional documented interactions.
- There are 54 matching marks between the requirements DSM and the expert DSM. Since all interactions in the requirements DSM are assumed to be documented, there are additional 54 marks being documented.
- The requirements DSM missed 12 interaction marks comparing to the DSM built using all systems engineering documents (Figure 5-7). These twelve marks will be accounted for as well.
- Therefore, 75+54+12 = 141.

The comparison between documenting requirements decomposition and not documenting is shown in Figure 5-32. We can gain 41% in documented system level knowledge by documenting requirements decomposition knowledge properly.

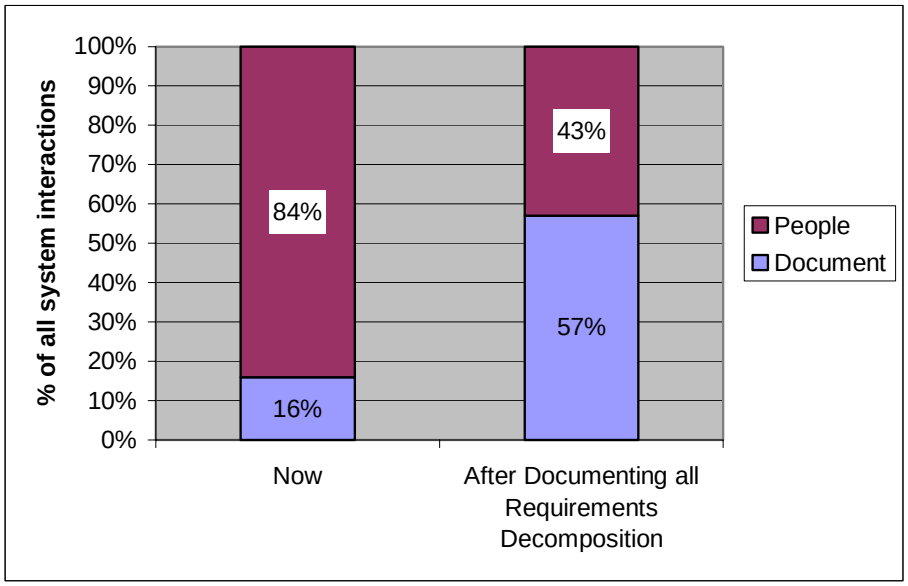


Figure 5-32: The Improvement on Documenting System Level Knowledge after Documenting Requirements Decomposition

Note even after proper documentation of the requirements decomposition, still 43% of the system level interaction depends on the experts. These 43% (247-141=106) of the undocumented system interactions includes:

1. Interaction between Assay Chemistry and the hardware design—64. The reason is the assay group did not write requirements document.
2. Feedback interactions from the power subsystem—17. This was a subjective decision by the power subsystem engineer.
3. Reliability types of interactions—12.
4. Function and spatial types of interactions undocumented—13.

Some of these interactions, such as *1, 2*, and *4*, can be documented, so that the percentage of documented interactions can rise higher in another case. Yet the documented system interaction will only approach 100% at the end of the design process because the interactions introduced by system emergent properties cannot be captured early on.

The conclusions to draw from the above discussions are:

1. The JNJ OCD OASIS program engineer should improve their documentation on the requirements decomposition, because it contains most of the system interaction knowledge.

2. It is possible to use requirements to capture and document the majority of the system interaction information.

5.3.3.3 Summary on the Documentation of System Level Knowledge

Objective 4 of this case study asks whether there is a document source that's better for capturing system level knowledge. From the above observations, we can conclude that the requirement documents and the requirement decomposition documents are excellent sources of system interface information. The rest of the existing system engineering documents at JNJ OCD did not capture as much information about system interfaces as the requirements documents.

Objective 4 also asks how well JNJ OCD does in documenting system level knowledge. The finding is that OCD engineers need to put in additional work for documenting the requirements decomposition in order to take the advantage of using requirements documents to predict system interactions. Currently, only 9% of the knowledge about requirements decomposition is documented. In addition, taking into account all of the system engineering design documents, only 16% of the system level knowledge is documented. That is a lower percentage than Ford throttle body case study even with the help of using requirements to predict system interactions (Ford case study did not have this method). After documenting the requirements decomposition knowledge, the documented system level knowledge can rise to 57%, which is a significant improvement. Therefore, again, the observation shows JNJ engineers should improve their work in documenting the requirements decomposition relationships.

5.4 Progress Made Regarding the Research Questions

The research questions this case study tries to answer include Q1-b and c, and Q2-c and d. The objectives of this case study were set to answer these research questions. The findings are summarized here.

Q1-b. How to predict system interactions early? How to predict system interactions for new technology?

This case study again tested the idea of building a DSM from requirements. A DSM was built from the OASIS requirement documents. It was compared to the DSM's built by the engineering experts following the traditional DSM building process. Total 247 marks did not match between the two matrices. Among the 247 unmatched marks, 167 unmatched marks could have been avoided.

- 75 marks could have been captured by the requirements DSM if
  - The JNJ OCD engineers produce complete set of requirements document, including the Assay Chemistry design requirements.
    - Requirements flow-down information is correctly documented and updated.
- 75 marks could have been captured by the expert DSM if
    - The JNJ OCD engineers invited all of the necessary people to the DSM building exercise, including the software engineers.
    - JNJ OCD engineers had learned some of the insights about the system interactions contained in the requirements document.
- 17 marks could have been avoided if the experts and the power subsystem engineers were equally conservative or aggressive about the feedback from the power subsystem to the rest of the hardware design.

Yet, among the 247 unmatched marks, 26 unmatched marks could not have been avoided, which reveal the limitation of the idea of constructing a DSM from requirements only. These 26 marks missed by the requirements DSM are:

- 14 marks were due to spatial types of interactions.
- 12 marks were due to reliability types of interactions.

The spatial types of interactions cannot be predicted based on requirements. Method such as Datum Flow Chain can help to capture these interactions from early on. System interactions concerning reliability requirements are just very difficult to predict early on, like any requirements concerning emergent properties of the system. Therefore, using the requirements to predict system interaction will always leave out some of the interactions concerning system emergent properties, which is the limitation of the matrix conversion method.

In addition, the comparison between the requirements DSM and the expert DSM also showed the following similarities:

- Both DSM's identified very similar iteration loops in the system.
- Both DSM's identified very similar elements in the system that may be the most critical to successful system integration.
- Both DSM's had system interaction ratio very close to 6, indicating they probably contained enough information about the system.

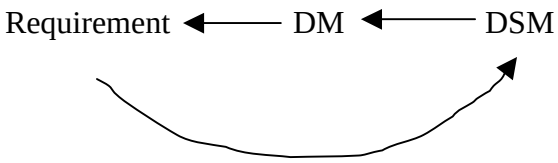
Therefore, we can conclude that the JNJ OCD case study again showed that we could obtain a valid and useful DSM from requirements without expert knowledge regarding the detailed design. The matrix transformation method is transferable to a product different from the CVC case study.

Furthermore, a shortcut was discovered to bypass the construction of a DM and obtain a DSM directly from requirements. It only works when dealing with system elements in the same hierarchy. Yet, combining this shortcut with the findings regarding how requirements are decomposed (Table 5-4), we obtained a more flexible method than the matrix transformation method. There are three ways to predict and capture system interactions.

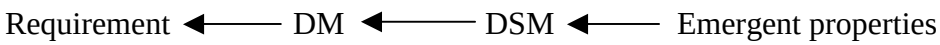
1. When the requirement is decomposable using the Axiomatic Design matrix, we may construct a DM and use the matrix transformation method to get a DSM, so that we could predict the system interactions.

   Requirement ⟶ DM ⟶ DSM

2. When the requirement is decomposable but not using the Axiomatic Design matrix method, we may use the shortcut method used in this case study to directly predict the effect of the requirement on the system interactions. Once the interaction is recorded in the DSM, it can then be reflected back to DM.



3. When the requirement concerns system emergent properties, we may have to wait until we discover the system interactions that affect the requirement. Then we could put the interaction in the DSM, and reflect it back to DM to show the effects of emergent properties on the decomposable requirements.



The next phase of the research is probably to test out whether these three paths are enough to cover all of the situations in predicting and documenting system interactions.

Q1-c. If we can predict system interactions, how complete is the prediction?

The discussion in Q1-b regarding the comparison between the requirement DSM and the expert DSM already revealed that the prediction of system interactions from requirements is not complete. The spatial types of interactions cannot be derived from requirements. Datum Flow Chain method is needed to capture them at early phase of the design. The reliability types of requirements represent the requirements concerning the emergent properties of the system, whose influence on the system interactions can never be completely predicted at early phase of the design process.

Table 5-4 further summarized the contribution to predicting system interactions from various types of requirements in the OASIS product design. Only part of the requirements could be decomposed and used to predict system interactions. The influences of the indecomposable requirements on the elements of the systems are not known until detailed design, prototyping, or even later in the product lifecycle. Therefore, it is important to document system level knowledge throughout the product lifecycle so that what learned in the past regarding certain system emergent properties may be used early on in the next design.

Table 5-4 also tells that only some of the requirements can be decomposed using the Axiomatic Design's design matrix. Many other requirements must be decomposed differently or even cannot be decomposed. Therefore, the Design Matrix in Axiomatic Design does not give a complete view of the system interactions. Using the Design Matrix to determine whether a system is designed to ideal (Axiom 1) is not appropriate.

Nevertheless, if we look at the other side of the problem, although the prediction of system interactions from requirements is not complete, it is still useful for planning purpose. In this case study, the two requirements that could be decomposed using the Axiomatic Design Matrix contributed the most to the prediction of system interactions, which was a fairly good match with the expert's DSM (see 5.3.1). Therefore, the Design Matrix, although does not provide a complete view of the system, is still very useful in helping planning for system integration at early phase of the design process.

Q2-c. What are the best sources of information for predicting system interactions?

Table 5-7 and Figure 5-29 show that the requirements documents in this case study are the best sources for predicting system interactions. Other traditional system engineering documents may not be constructed as early as requirements documents, and did not contribute as highly to the capturing of system interactions. Therefore, the method of capturing system interactions from requirements enabled us to better predict, capture, and document system interactions.

Q2-d. How companies are doing with managing system level knowledge?

JNJ OCD is not doing such a great job in documenting system level knowledge. The knowledge regarding requirements decomposition, being so important to capturing system level interactions (Figure 5-29), is poorly documented (Figure 5-30). Considering all of the available system engineering documentation for OASIS program, only 16% of the knowledge regarding system interactions could be found in documents. If all requirements decomposition knowledge were to be documented, we can get 41% increase in documented system level knowledge. Further increase could be achieved by completing the requirements for assay chemistry. Therefore, JNJ OCD can improve its situation very quickly.

JNJ OCD is the third example to show that many industries do not have good practice or method to document the very important system level knowledge. A framework such as the one proposed in Chapter 3 is very much needed. Using the method to derive system interactions from requirements, other companies may also be able to quickly improve the amount of documented knowledge regarding system interactions.

5.5 Summary

This case study was carried out at Johnson and Johnson Otho-clinical Diagnostics, in the OASIS program. The research objectives of this case study included:

1. Find out if the matrix conversion method is transferable to a different case study to build a DSM from requirements.
2. Investigate how closely the DSM derived from requirements predicts the reality by comparing it with the DSM produced by expert engineers and scientists using the traditional DSM building method.
3. Understand whether the matrix conversion method can be used for all types of requirements or just Functional Requirements. Observe what types of requirements drive the system interactions.
4. Observe the sources of information in identifying system interactions. How much system interaction knowledge is captured in documents and how much is in people's heads?

JNJ engineers' objectives were:

- Predict system interface problems that may happen during integration and verification.
- Aid the system integration manager's work on planning and managing OASIS subsystem interfaces.

This case study took 5 OASIS systems engineering documents to build DSM's—the architecture definition document, the requirements document, the function flow diagram, the mechanical interface document, and the hazard mitigation document. The DSM built from the requirements are missing some of the spatial relationships captured by some of the other 4 documents. The observations made regarding each objective are summarized as follows:

1. Regarding objective 1, a DSM was made from requirements. In the answer to the second objective, we learned that the DSM derived from the requirements matched well with the DSM constructed by the expert engineers in the OASIS program. The unmatched marks were not due to the method, but rather the incomplete documentation of the requirements.

   In addition, a short cut to construct a DSM from requirements without going through DM was discovered. This discovery enabled us to do the following:
   - Save time when we only need to know the interaction at one level of the system hierarchy.
   - Allow requirements that cannot be decomposed using the DM to show their effects on system interactions directly in a DSM.

2. The DSM built from requirements matched many marks in the DSM predicted by the experts.
   - Most of the unmatched marks between the requirements DSM and the expert DSM are due to missing information, rather than the method.
   - The resulting DSM from requirements has an interaction density ratio around 6.
   - The topology of the resulting DSM is very similar to that of the expert DSM. They both identified the same elements involved in the iterations.

- Both requirements DSM and expert DSM predicted very similar priority list.

Therefore, the method of obtaining a DSM from requirements is valid.

3. Not all requirements can be decomposed using the Axiomatic Design method. Therefore, the Axiomatic Design's Design Matrix cannot predict all system interactions. The Axiomatic Design "ideal" engineering design may not be so ideal when all factors that contribute to system interactions are truly taken into account.

   However, predicting system interaction is still an attractive idea because good prediction, even if incomplete, can aid us in decision making at early stage of the design process. The functional requirements used in this case study, including the functional requirements, the function flow diagrams, and the architecture diagram can very well predict most of the system interactions and the system topology. Therefore, the Axiomatic Design's Design Matrix is not so bad after all. A DM may not be able to determine whether a design is ideal or not, but it can aid to predict system interactions for the purpose of managing iterations and interfaces.

4. JNJ OCD did not well document their system level knowledge. The requirements decomposition knowledge was poorly documented despite the availability of requirements management software. Among all existing systems engineering documents, only 16 percent of the system interaction knowledge was documented.

   Compare to the traditional types of system engineering documents, the requirements document can provide the most information about system interactions if the matrix conversion method is used. Using the requirements document, JNJ OCD can get a 41% boost in the documented system interaction knowledge.

The objectives of the JNJ engineers were achieved by this case study too. The DSM built from the requirements provided very similar system element priority list as the experts' DSM for focusing the system engineering effort. Therefore, this research work confirmed that JNJ system engineers were working on the correct direction.

However, the DSM built from the requirements predicted 75 additional interactions that were not captured by the experts. The comparison between the requirements DSM and the experts' DSM revealed the danger of the organizational gap between the hardware and the software groups. The requirements DSM helped the JNJ engineers to do a better job in dealing with system interactions for system integration purpose.

In addition, this case study showed the JNJ OCD engineers needed to improve their requirements documentation and system level knowledge documentation. These are all valuable lessons for JNJ OCD to achieve their corporate goals.

6 Status of Research Questions

6.1 The Initial Research Questions

The initial research questions were given in Chapter 1. The motivations and details of each question are reviewed and summarized here.

6.1.1 Obtaining System Interactions at Early Stage of the Design Process

The first set of research questions concerns predicting system interactions in the product at early stage of the design process. System interactions were defined as the interactions among the variables in the product, not the design tasks for project management, and not the interactions among the people. The high level assumption made here for studying the product system interactions rather than the task or human interactions is that the product system interactions should drive the interactions among the design tasks and the human communication.

Obtaining system interactions at early stage of the design process is important because of the following reasons:

1. To avoid costly late design changes in the product.
2. To aid the project management work and facilitate the communication among the people involved in the design of the system.
3. To aid the concept and architecture selection based on the complexity of system interactions.

It was also recognized that the prediction of system interactions at early stage of the design process might never be complete due to the existence of the emergent properties of the systems. Therefore, the detailed research questions under this topic are:

Q1-a: What methods have been used in the past to capture system level interactions? What are the strengths and weaknesses of existing methods? Is DSM a good way to predict system level interactions?

Q1-b: How to predict system interactions early? How to predict system interactions for new technology?

Q1-c: If we can predict system interactions, how complete is the prediction?

6.1.2 Managing System Level Knowledge in the Organization

The second set of the research questions deal with managing the system level knowledge in an organization. The system level knowledge exists in an organization where each individual is only responsible for a piece of the system. The system level knowledge is the knowledge about the rest of system outside of the component that an engineer is responsible for. System level knowledge includes the knowledge about what other elements exist in the system, what the interactions are among the elements, and who or where to find information about other parts of the system.

System level knowledge management is important because of the following reasons:

1. Past research experiences had shown the system level knowledge is poorly documented in companies.
2. Relying on expert's tacit knowledge to deal with system level knowledge limits companies' ability of compete in the market.
3. Good system level knowledge management can aid the knowledge-based engineering effort.
4. A knowledge management framework is needed to identify what system level knowledge should be documented and in what format, so that the browsing and reuse of documentation can be much more efficient.

Therefore, the research questions under this topic include:

Q2-a: What has been done in managing system level knowledge?

Q2-b: Is there a better way to capture, store, and represent system level knowledge?

Q2-c: What are the best sources of information for predicting system interactions?

Q2-d: How companies are doing with managing system level knowledge?

Q2-e: How to encourage engineers to document system level knowledge? Make recommendation to the management.

6.2 Progress Made and Future Research Questions

In order to investigate the above research questions, literature searches were conducted and research methods were proposed. Two case studies were carried out at CVC and Johnson and Johnson Ortho-clinical Diagnostics (JNJ OCD) in order to apply the research methods to find out more about the research questions. The details of the research investigations are in Chapter 2 to 5. Table 6-1 shows which section(s) in each previous chapter has discussion on which research questions. This section summarizes the findings so far on each research questions.

Research Question	Chapter 2	Chapter 3	Chapter 4	Chapter 5
Q1-a	2.1 and 2.2			
Q1-b		3.2	4.2.1, 4.2.2, 4.3.1, 4.3.2, 4.3.3, 4.3.4, 4.3.5, 4.3.6	5.2, 5.3.1
Q1-c			4.3.7	5.3.1, 5.3.2
Q2-a	2.3			
Q2-b		0	4.2.3, 4.3.9	
Q2-c		0	4.4	5.3.3.1
Q2-d		0	4.3.8	5.3.3.2
Q2-e			4.3.3	5.2.6

Table 6-1: Research Questions Discussed in Each Previous Chapter

6.2.1 Obtaining System Interactions at Early Stage of the Design Process

The findings regarding research questions Q1-a through Q1-c are discussed here.

6.2.1.1 Existing Methods Dealing with System Interactions (Q1-a)

Q1-a: What methods have been used in the past to capture system level interactions? What are the strengths and weaknesses of existing methods? Is DSM a good way to predict system level interactions?

Many existing systems engineering methods that deals with system interactions are reviewed in Chapter 2. Table 2-3 summarized the strengths and limitations of each method. Since the only place in this thesis that addresses this question is Chapter 2, the answer to this research question at the end of Chapter 2 (see section 2.4) does not need to be repeated again. In short, the review of literature suggests find a way to combine the DSM, DM, and requirements classifications so that we could predict and analyze system interactions at early stage of the design process.

6.2.1.2 A New Method to Predict System Interactions (Q1-b)

Q1-b: How to predict system interactions early? How to predict system interactions for new technology?

The answer to this research question contains four parts. First, a matrix transformation method was proposed to predict system interactions for product design facing the architecture or radical innovation. This thesis research work revised and matured the matrix transformation method through trials on actual engineering projects. Second, the usefulness of this method in industry settings is examined, which includes whether the method is needed by industry projects, whether it can be executed in real industry settings, and whether its results are beneficial to the companies.. Third, how valid the results of the matrix transformation method were reviewed. Fourth, from the case study experiences, a larger and more flexible framework is discovered to predict and capture system level knowledge throughout the product lifecycle.

6.2.1.2.1 The Matrix Transformation Method

The literature search revealed that a method relating DM and DSM would enable us to overcome the DSM's inability to be constructed at early phase of the design process, or be applied on a new product. Therefore, a method to transform an Axiomatic Design's Design Matrix (DM) into a Design Structure Matrix (DSM) was introduced in Chapter 3. The method includes three steps:

1. Construct a Design Matrix (DM) following the same rules in Axiomatic Design.
2. Choose the output variables in each row. The selection of output variables is unique for each row and column. In other words, there could only be one output variable selected in each row and column of the DM.
3. Permute the DM by row so that all the output variables are moved to the diagonal position. The resulting matrix is a DSM for the Design Parameters. The column headings remain the same as the DM, the row headings are the same as the column headings.

To refresh the readers' memory, the same example in Chapter 3 is copied below to demonstrate the three steps of matrix transformation method.

Step 1: Construct a DM:

	DP1	**DP2**	**DP3**
FR1	X	O	X
FR2	X	X	O
FR3	O	X	X

Step 2: Choose output variables:

	DP1	**DP2**	**DP3**
FR1	X	O	(X)
FR2	(X)	X	O
FR3	O	(X)	X

Step 3: Permute the DM by row to move the output variables to the diagonal location. Rewrite the headings of the rows to follow those of the columns. Then we have a DSM for the DP's.

	DP1	**DP2**	**DP3**
DP1	X	X	O
DP2	O	X	X
DP3	X	O	X

The selection of the output variable in the above Step 2 matrix is not unique. Figure 3-3 demonstrates that different choices of output variables may give different resulting DSM's. During the CVC case study, it was discovered that when DM and DSM are used in engineering design situations, the only valid output variable selection is the diagonal elements in the DM. Only by selecting diagonal elements, could we have a DSM that represents logical design process, and has converging iterations. The proof is in section 4.3.6. The valid choice of output variables originates from the Axiomatic Design's rules on how to construct a DM. The most influential Design Parameter to the requirement is placed on the diagonal position. Therefore, the diagonal elements have the most influence to the realization of the requirements. More details are in section 4.3.6.4.

The implication of the choice of the output variables is that DSM and DM can be transformed back and forth easily. This implication contributes largely to the more complete framework for predicting and capturing system interactions, which will be discussed later in the third part of this section.

The CVC case study also experimented with the question of when to stop the decomposition in the system. The general guidelines learned from the CVC case study are:

- When the Design Parameters in the DM are assigned to a single person or very small groups of people whose interactions can be easily managed.

- When further decomposition of the system does not provide any more insights about the elements involved in system iterations any more.

These observations are made based on CVC case study. They may or may not be true for other cases, but could be used as general guidelines.

The JNJ OCD case study discovered that it is possible to make a DSM directly from requirements if we are only interested in one level of the system hierarchy (see 5.2.4). This shortcut did not invalidate the matrix transformation method. It is just a special case of the matrix transformation method when system decomposition is not of interest.

Several general rules were learned from the JNJ OCD case study to obtain complete and meaning for results from the matrix transformation method. They are as follows:

- A complete set of requirements needs to be produced. In the JNJ OCD case, when the assay chemistry group did not produce the requirements for their subsystems, the system interactions between assay and hardware were not captured by the matrix transformation method.
- Use Datum Flow Chain method [Mantripragada and Whitney (1998)] to capture the system interaction related to the packaging of the assembly.
- Update the requirements decomposition structure to reflect the latest learning regarding system interactions.

In short, the matrix transformation method became more mature in these two case studies. The correct choice of the output variables was discovered to be the diagonal elements. The matrix transformation method was discovered not only to be able to convert a DM into a DSM, but also the other way around. Thus we can take different views of the product easily. A short cut of obtaining a DSM for only one level of the system hierarchy directly from requirements was discovered. Several general guidelines for when to stop the decomposition in the system and for constructing a good DM and DSM were established, which will benefit future applications of this method.

6.2.1.2.2 Usefulness in Industry

The matrix transformation method fits the practical needs of engineering companies. In the CVC case study, the project was to integrate a new chuck into the existing cluster machines. If Axiomatic Design were applied, probably the entire cluster machine has to be redesigned so that the system interactions could become decoupled or uncoupled. However, pressed by project deadline, CVC engineers had no time to redesign the whole machine. They need a method to help them to integrate the chuck into the existing machines quickly. Therefore, compared to Axiomatic Design, the matrix transformation method provides more practical help. There exist countless cases in many companies where the change in design is only partial and the project time pressure is high. The matrix transformation method provides realistic help to these engineering projects.

Being the first of its kind, the matrix transformation method was shown to be executable in real engineering projects. In the CVC project, the three steps of matrix transformation were followed. A Design Matrix (DM) was built with several levels of system decomposition. A Design Structure Matrix (DSM) was obtained from the DM. In the JNJ OCD case study, the three steps of matrix transformation were not strictly followed because only one level of the system hierarchy was of interest. A shortcut of building DSM from requirements was discovered, which does not invalidate the matrix transformation method. Following the shortcut, a DSM was easily constructed from requirements again. Therefore, both case studies have shown that the matrix transformation method to obtain system interaction information from requirements without very much experience with the product was feasible in real engineering projects.

Both case studies showed that the resulting DSM from the matrix transformation method provided valuable learning for the engineers on the projects. In the CVC case study, the DSM built from the DM provided the following benefits:

- It predicted the iteration loops that would happen during the integration process before the actual integration started. The engineers and managers thus were able to prepare for the iteration loops early on.
- It identified necessary system interactions based on the nature of the design. These iterations would have been overlooked by the engineers on the team if they were to follow the same way that things had always been done.
- It provided a basis for the information flow between design organizations that were located geographically far apart.

The benefits from the JNJ case study included:

- The DSM built from the requirements pointed out that the engineers had overlooked the complex system interactions between the hardware and software due to the existence of the organization gap. The DSM provided a warning on the situation, and a way to facilitate the communication.
- The requirements DSM also captured system interactions that expert engineers did not realize even when they are already in the detailed design phase.
- The requirements DSM produced a priority list for the elements that were important to system integration. This priority list matched closely with the one that the system engineers produced. Therefore, the matrix method gave JNJ engineers confirmation on the direction of their work.

In summary, both CVC and JNJ case studies have shown that the matrix transformation method is needed in the industry. The method is executable in real engineering projects, and the results of the method provided large benefits to both projects. The engineers in both companies took the resulting DSM's from these case studies seriously as references to aid their systems engineering efforts. Therefore, the matrix transformation method is very useful in industry settings.

6.2.1.2.3 Verifying the Effectiveness of the Method

In a strict sense, the matrix transformation method should be validated with the following steps:

1. During the concept development phase of a design project, apply the matrix transformation method and produce a prediction on the system interactions.
2. Keep tracking of the actual system interactions in a DSM format as the design progresses into detailed design and through the entire lifecycle of the product.
3. Compare the DSM produced at various stage of the design process with the DSM produced at the beginning, and see how complete the prediction is.

However, many factors in the two case studies prevented the above steps to be carried out. In the CVC case study, although the DM and DSM were built at early phase of the design project, the sudden restructuring of the company after merger stalled the case study project. The change in the project personnel and the author's school semester timing made it difficult to follow up with the actual development of the project. In the JNJ OCD case study, the school semester timing also made timing a bit off. The author had to join the project during the detailed design phase. Fortunately, with the requirements documents, it was still possible for the author to apply the method without being influenced by the system engineers' experience. Therefore, although the DSM from the requirements was built during the detailed design, it was still a valid prediction matrix. Being in the detailed design phase, it was possible to ask the JNJ engineers to construct their own DSM and compare with the prediction DSM. This comparison provided one point in time about how well the requirements DSM can capture the system interactions. Yet, following up with what actually happened later on in the integration phase of the project was impossible due to the distance and school semester.

Therefore, the verification of the DSM prediction method in both cases did not follow the strictest procedure. Yet, this research did the best it could to find out whether the DSM from the matrix transformation method was valid. In the CVC case study, the system engineers from various departments who were assigned to the project reviewed the DSM produced from requirements. They agreed the DSM captured what they expected to be future system

interactions. Therefore, the engineers accepted the DSM for future project planning. In the JNJ OCD case, the DSM made from requirements was compared with the DSM expert engineers built based on their knowledge. Although the requirements DSM missed a lot of the interactions in the expert DSM, the expert DSM equally missed a large amount of interactions in the requirements DSM. Detailed analysis found that the unmatched marks between the two DSM's were mainly due to missing information or improper requirements documentation. Only a small amount of unmatched marks were due to the method (see Table 5-2). Therefore, if the experts constructed the DSM's correctly, and if the requirements documents were complete, the two DSM's would have been very similar (Figure 5-17). In addition, the requirements DSM and experts DSM identified the same system iterations and the same system interaction priority list. Therefore, the effects of the DSM build from the requirements were not so different from the expert DSM.

Despite the fact that the method was not validated strictly, both case studies had shown very positive signs about the usefulness and validity of the prediction DSM's built based on requirements. Being the first of its kind, the positive results from the case studies are enough to encourage us to continuously improve the method and use it to improve real engineering projects. We should also design future research projects focusing on discovering how much of the system interactions could be predicted at early phase of the design process. More discussions will be in the later section on future research questions.

6.2.1.2.4 A More Complete Framework for Predicting and Capturing System Interactions

In light of the choice of the output variables, Design Matrix (DM) and Design Structure Matrix (DSM) can be transformed from each other in both ways. Therefore, a more complete framework for not only predicting but also continuously capturing system interactions can be proposed. The need for such a framework can be found in the next section regarding the completeness of the prediction DSM. The framework is first stated in Chapter 5 (5.4) and is summarized in Figure 6-1:

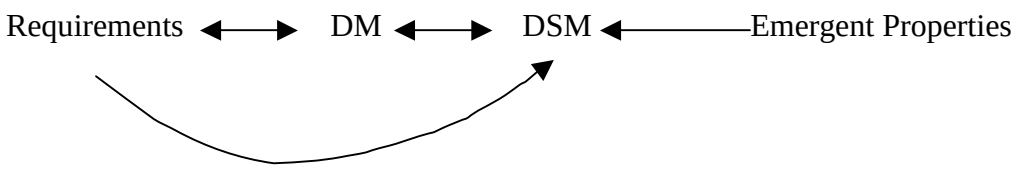


Figure 6-1: A More Complete Framework for Predicting and Capturing System Interactions

1. When the requirement is decomposable using the Axiomatic Design matrix, we may construct a DM and use the matrix transformation method to get a DSM, so that we could predict the system interactions.

   Requirement ⟶ DM ⟶ DSM

2. When the requirement is decomposable but not using the Axiomatic Design matrix method, we may use the shortcut method used in this case study to directly predict the effect of the requirement on the system interactions. Once the interaction is recorded in the DSM, it can then be reflected back to DM.

   Requirement ⟵ DM ⟵ DSM

3. When the requirement concerns system emergent properties, we may have to wait until we discover the system interactions that affect the requirement. Then we could put the interaction in the DSM, and reflect it back to DM to show the effect of the emergent properties on the decomposible requirements.

   Requirement ⟵ DM ⟵ DSM ⟵ Emergent properties

In short, this new framework of predicting and capturing system interactions contains both the reductionism and holistic views of system interactions. It enables us to predict what we can and still capture the learning throughout the product lifecycle.

6.2.1.3 The Limitations of the Method (Q1-c)

Q1-c: If we can predict system interactions, how complete is the prediction?

A complete prediction DSM would be one that captures all of the system interactions that happen throughout the product lifecycle. As explained in the above section regarding the validity of the method (section 6.2.1.2.3), neither case study was able to verify the prediction DSM by following up with what happened later on the development process. However, two observations in the case studies indicate that the prediction DSM was not complete.

The first observation is on the system interaction density (for the definition, see 4.3.5.2). The JNJ OCD case study had a DSM interaction density at around 6, which falls into the normal range based on past case studies. The CVC case study, however, has a system interaction density of only 2.8, which is much lower than the normal value 6. Although it is still a hypothesis that a well-constructed DSM should have system interaction density at around 6, having so few marks in the CVC DSM may be an indication that many of the system interactions were not captured or known. Unfortunately, it was difficult to follow up with the CVC project to verify this point.

The second observation of incomplete prediction of system interactions was made in the JNJ OCD case study. From Table 5-2, the DSM made from requirements was unable to capture the reliability and spatial arrangement related system interactions that were in the DSM built from experts. In addition, the DSM built from all design documents (Figure 5-7) also captured interactions related to spatial relationships that the requirements DSM (Figure 5-8) did not have. Table 5-4 further demonstrates that not all requirements can be decomposed and used to predict system interactions. Therefore, many of the system interactions that contribute to the requirements cannot be found until later in the design process.

The observations made from the case studies regarding the incompleteness of the predictions should not have come as a surprise. Systems have emergent properties. The system interactions associated with the emergent properties may not be predicted early in the design process. Therefore, the prediction on system interactions is always incomplete.

However, even when the DSM's built from the requirements were incomplete, they were still very useful in providing insights to aid the system engineering efforts in both case study companies (see summary in 6.2.1.2.2). Later on in the future research direction section, a framework is proposed to aid the product development practitioners to judge which matrix method should be used and how much trust one can have in the prediction DSM.

6.2.1.4 Future Research Directions for Predicting System Interactions

6.2.1.4.1 Which Matrix Methods to Use for Which Type of Product Design

The first new hypothesis is that we can determine which matrix method is the most suitable to use based on the type of product innovation we are facing. Figure 6-2 shows the hypothesis based on Henderson and Clark (1990)'s framework of product innovations. Each quadrant in the graph is explained as follows:

- When the design project is an incremental innovation, both the core technology and the system interfaces do not change. Therefore, the existing expert knowledge on the system and the components can be reused and a DSM can be built at early phase of the project, without having to construct a DM first. If the engineers are interested in how requirements relate to the system interactions, they may transfer the DSM into a DM for that purpose. The example in CVC case study had shown that the DSM transformed from a DM reveals system interactions that experts overlooked (see section 4.3.3.2). Therefore, the construction of the DM may also help to find out whether all the system interactions identified by experts were correct from product design point of view.

Since the past expert knowledge on the system and component design is equally applicable to the incremental innovation design project, the DSM constructed at early phase of the design process can already be quite complete. Therefore, the DSM can be fully trusted to make decisions on product architecture, design process improvement, and project planning and management.

An example of such innovation is the Ford mechanical throttle body described in the author's master degree thesis [Dong (1999)]. The mechanical throttle body has been designed for many years and the design is mature. New projects may change the cam curve or spring design parameters, but does not make any significant changes. Therefore, a DSM can be made for the design of this mature product. The DSM can be used to improve the efficiency of the design team and design process.

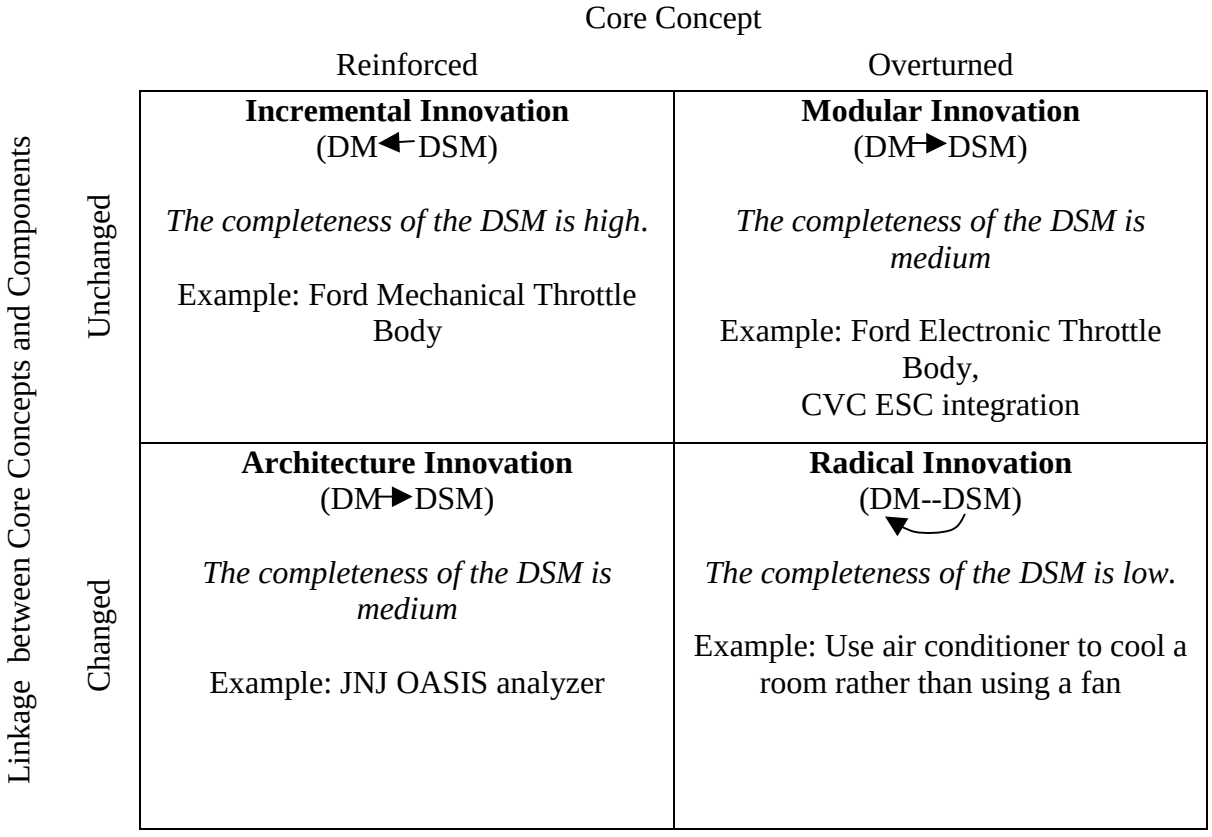


Figure 6-2: Which Matrix to Use for Which Type of Product Innovation

- In modular innovation, the core concept of the module has been changed, but the system interfaces between the module and the rest of the system are not changed. In this case, although the interfaces between the module and the bigger system remain the same, the design of the module changed. Therefore, when designing the new module, we do not have previous knowledge of the interactions inside the module. We need to build a DM in order to get a DSM that predicts the interactions within the module, and shows how system interfaces of affects the module design.

  The CVC Electro-static chuck system integration project studied in this thesis is an example of modular innovation. Ford's new electric throttle body, which will replace the mechanical throttle body, is also an example of modular integration.

  In modular innovation, the DSM cannot be built from experts' past experiences, but must be built from DM. Therefore, the limitation of the matrix transformation affects the completeness of the DSM that predicts the system interactions. In the CVC case study, despite the incompleteness of the DSM, the engineers still very much appreciated the new insights the DSM provided at early phase of the project. Future research should follow up with one of these modular design projects from beginning to end in order to track how complete the prediction DSM is.

- In architecture innovation, the core concepts of the modules in the system remain the same, but the interface between the modules changed. In this case, the past knowledge regarding the system interfaces is no longer valid. Therefore, we cannot rely on the experts to build a DSM. Instead, we must go through the DM and the matrix transformation method in order to obtain a DSM that predicts system interactions.

  The JNJ OCD OASIS product studied in this thesis is an example of architecture innovation. The thin film and the wet chemistry modules are already built in existing

products. Yet to integrate them into one analyzer will have to introduce new system interfaces.

Again, due to the limitation of the matrix transformation method, the DSM built from the DM cannot predict all of the system interactions. However, because the experts already understand the technology, there might be a better chance to capture many of the interactions between the modules by working through DM. In the JNJ case study in this thesis, the system interaction density ratio was at the normal level (around 6), which may be an indication that the prediction DSM from the requirements has captured a lot of the important interactions. However, in order to verify this point, we should again follow through a design project that involves architecture innovation to see the change in the amount of the system interactions learned from the early phase of the design until the end of the product lifecycle.

- The last type of innovation is the radical innovation. Both the core concept and the relationship among the components in the system change. Therefore, the only way to predict system interaction is to use the matrix transformation method. There has not been a case of radical innovation among the author's past case studies. But we can certainly borrow the example in Henderson and Clark's paper (1990). If a ceiling fan company decided to start designing and manufacturing air conditioners, the new product design is a radical innovation for the ceiling fan company.

  In the radical innovation, because very little is known about the technology and the system, the DSM constructed from the DM is expected to have low system interaction density ratio and not to capture a lot of the system interactions. The usefulness of the DSM for system planning and management is questionable, but the managers probably don't have anything better than the prediction DSM to help their thinking either. In this type of innovation, when the DSM is built at early phase of the design process, the engineers and managers must keep in mind that more and more system interactions will be learned as the design carries on. They should update the DSM frequently to reflect the new learning, so that the system interfaces can be analyzed

based on the new situation, and the project management plan can be revised to better direct the design work. Of course, we again need a case study to follow through this type of product development process and record the maturity of the DSM.

On the other hand, when a product design is a radical innovation, it is the best time for the company to take advantage of the Axiomatic Design's philosophy on selecting the concept with the least system interactions. Radical innovation provides the best opportunity for product improvements, while process improvements is done the best for incremental innovations.

We may also associate this hypothesis between the relationship of the product innovation and the applicability of the matrix design methods with the maturity of the technology. At the left side of the S-curve, the technology is new, and the competitive advantage is mainly on the product design. At the right side of the curve, the competitive advantage is in the efficiency of the product development process. The ability of predicting the system interactions increases as we go from left to right, and the importance of having a DSM early on in the design process also increases, because DSM can provide system analysis to improve process efficiency.

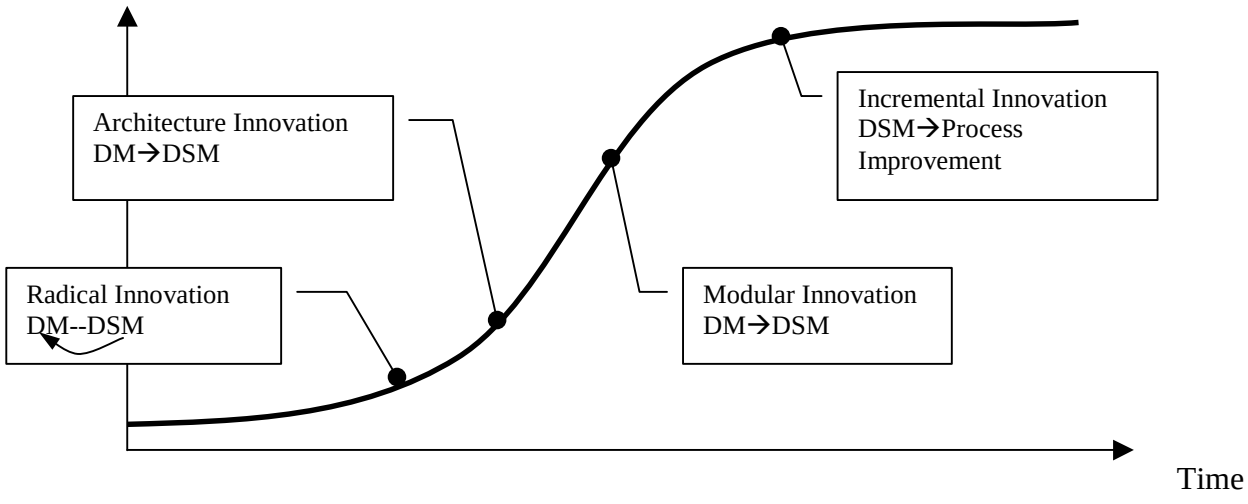


Figure 6-3 Which Matrix to Use at Which Phase of the Technology "S" Curve

In summary, we have proposed a hypothesis to associate the product innovation and the applicability of the matrix design methods. The goal of this hypothesis is to put the matrix transformation method into a framework to aid the practical decisions of the product development engineers and managers. This hypothesis should be tested with carefully selected case studies to meet the condition of each quadrant. The case studies should follow through the product design from concept development until at least the end of the manufacturing of the product, or even better until the end of the product life. Compare the new learning about system interactions with those predicted at early stage of the design process, and then judge the completeness of the prediction in each quadrant in Figure 6-2.

6.2.1.4.2 What Makes Two DSM's Similar

In the JNJ OCD case study, the requirement DSM was compared to the DSM built by the experts using traditional DSM construction techniques. The two DSM's looked different, but gave very similar information about the system. Therefore, the measurements of the similarity of the DSM should include not only all of the marks at the same location, but also others such as the topology of the system, the suggestion on the important system elements, etc. For more details on the measures used in the JNJ case study, please see section 5.3.1.7. In future research, it is worth studying how to compare the two DSM's and what the metrics should be to judge whether two DSM's are similar. Future research can look into this question from more case studies in other companies. The math literatures on matrix manipulations should also be reviewed.

6.2.1.4.3 Use the Product Interactions to Lead the Process and Organization Design

A hypothesis this thesis took on was that the product interactions should drive the interactions among the design tasks and the interactions among the people in the organization (see section 1.2.3.1.1). Therefore, this thesis focused on predicting the interactions among the design variables in the product. Although in both case studies, the people interactions were studied by associating the design parameters with the responsible people or teams, the interactions among people due to the task assignments were not captured. Future research should look into how we might relate the interactions among the product elements with the

interactions among the tasks. If this association can be made in a systematic manner, then we can even predict the interactions among the tasks and the people from requirements.

6.2.2 Managing System Level Knowledge in An Organization

6.2.2.1 Existing Practices in Managing System Level Knowledge (Q2-a)

Q2-a: What has been done in managing system level knowledge?

The existing knowledge management work was surveyed. The discussions regarding this question only exist in Chapter 2. Therefore, the summary regarding this research question at the end of Chapter 2 is still valid, and will not be repeated here. The readers may turn to section 2.4 for the summary.

6.2.2.2 System Level Knowledge Management Framework (Q2-b)

Q2-b: Is there a better way to capture, store, and represent system level knowledge?

A framework for managing system level knowledge is proposed in Chapter 3 (Figure 3-1). A list the types of system level knowledge is also listed in section 3.1.2. They are *What, How, When, Where, Who, Which, Why, and Rules*. Guru Prasanna (2000) tested the framework using the following examples:

- Ford's design documents for throttle body
- Ford's throttle body design DSM's [Dong (1999)]
- CVC MOCVD project design documents
- CVC ESC project design documents
- CVC ESC design DSM (see Figure 4-8)

Prasanna found the categories for system level knowledge was complete for the documents listed above. In addition, it was also found that the knowledge management framework could serve as a basis for finding out what types of knowledge a company is lack of, and help the company to improve its knowledge management effort.

Yet, Prasanna only tested documented design knowledge in the above examples, which falls in the FR and DP domains in Figure 3-1. The classifications of knowledge should be also tested in other domains such as marketing, manufacturing, product testing, and field service. In addition, more case studies for different types of products are also needed. The usefulness of the framework was also not tested by Prasanna's thesis. Therefore, although the results from Prasanna's thesis are encouraging, we still need to do more testing of the knowledge management framework.

6.2.2.3 The Best Source of System Interaction Knowledge (Q2-c)

Q2-c: What are the best sources of information for predicting system interactions?

The goal of this question was to find out whether the existing design documents in companies are capable of capturing system level knowledge. Both the CVC case study and the JNJ OCD case study showed that the requirements documents were the best source among all documents for information to predict system interactions. In the CVC case study, the requirements document successfully predicted system interactions using the matrix transformation method. Since the CVC engineers did not produce other documents to capture system interaction information, there was no comparison with other system engineering techniques in the CVC case study. On the other hand, the CVC case study showed that requirements documents are the most common document produced in an engineering project. The matrix transformation method enabled us to obtain system interaction information using the most basic information in engineering projects without adding documentation effort.

JNJ OCD engineers produced other system engineering documentation to capture the system interactions, including function flow diagram, mechanical interface documents, etc. However, the discussions in section 5.3.3.1 showed that requirements document captured most of the system interactions contained within the rest of the documentation combined. Therefore, again requirements documents were the best source of information for predicting system interactions. Furthermore, we learned from the JNJ case study that in order for a

requirements document to be a good source of system interaction knowledge, the requirements document must include the following pieces:

- The flow-down of higher system level requirements to lower level subsystem level requirements is traceable and documented.
- The requirements on all subsystems and components in the design are documented.
- Requirements are updated as additional system interactions are learned.

The observations from two case studies are not conclusive, but are very encouraging. We can rely on well-written requirements document and the matrix transformation method to predict most of the system interactions, while other existing system engineering documentation techniques only capture a small amount of the total system interactions. However, based on the discussion on questions *Q1-c*, the matrix transformation cannot predict all of the system interactions. We must use other techniques to predict spatial types of interfaces. Datum Flow Chain is very suitable for this use. We must also be aware of the emergent properties of the systems and how they affect the fulfillment of the both the decomposable and indecomposable requirements.

6.2.2.4 Current Industry Status in Documenting System Level Knowledge (Q2-d)

Q2-d: How companies are doing with managing system level knowledge?

Despite the fact that the existing requirements document can predict most of the system interactions, both CVC and JNJ OCD case studies showed the system interactions knowledge was poorly documented in both companies (Figure 4-16 and Figure 5-32). Even the knowledge regarding requirements decomposition, which should be a part of the requirements document, was not properly documented (Figure 5-30). These observations further confirmed the need to improve the current practice in managing system level knowledge in companies.

6.2.2.5 Deployment of the Knowledge Management System (Q2-e)

Q2-e: How to encourage engineers to document system level knowledge?

Any knowledge management system requirements documentation work in order to capture the knowledge. As mentioned in section 3.1.1, it is important for the knowledge management system to deliver not only long-term benefits to the organization, but also short-term benefits to the engineers who spend time to document. The knowledge management framework proposed in Chapter 3 can provide immediate benefit on documentation when the matrix transformation method is applied. The diagram in Figure 6-1 shows that requirements documentation can be converted into critical knowledge regarding system interactions and project management. Lessons learned later in the project concerning system emergent properties can also be captured in DSM to update the action plan. Therefore, this knowledge management framework proposed in Chapter 3 shall be able to deliver both the long-term and short-term benefits to the organization.

6.2.2.6 Future Research Directions for Managing System Level Knowledge

The following topics are future research directions extended from the investigation regarding this set of research question:

- Other documents that concern the interactions among Process Variables (see Figure 3-1) shall also be investigated.
- Since system level knowledge includes not only the *How1* and *What*, the documentation that concern the *When*, *Which*, *Why2*, and *Rules* shall also be investigated.
- The rules of designing a system are explicit knowledge distilled from tacit knowledge. A large amount of design knowledge exists in tacit form. Knowledge management effort in an organization shall find ways to convert the tacit knowledge into explicit knowledge so that it can be easily reused by junior members of the team or programmed into knowledge based engineering applications.
- Since the knowledge management framework is able to follow the design process and produce not only short term but also long term benefits for the organization, we may implement the framework and the matrix transformation method in a software application. The software can be written in JAVA, ASP, and JAVA Script to be used

over web pages. The software can thus be independent of computer platforms, and have a good user interface design.

6.3 Summary

This chapter summarizes the progressed made in answering each research questions (see section 6.1). A matrix transformation method was proposed to bridge the Axiomatic Design's Design Matrix (DM) and Design Structure Matrix (DSM). This matrix conversion method takes the advantage of both DM and DSM, and produces a framework for continuously managing system level knowledge and benefit the organization in the short-term and long-term. From the understanding on these research questions, future research directions were also identified.

7 Conclusions and Future Work

7.1 Summary on the Research Questions

This thesis research started with two main topics, which include total eight research questions. This section summarizes the findings in this research regarding each of the research questions. For more details, please review the discussions in Chapter 6.

7.1.1 Obtaining System Interactions at Early Stage of the Design Process

The research questions under this topic are:

Q1-a: What methods have been used in the past to capture system level interactions? What are the strengths and weaknesses of existing methods? Is DSM a good way to predict system level interactions?

Q1-b: How to predict system interactions early? How to predict system interactions for new technology?

Q1-c: If we can predict system interactions, how complete is the prediction?

Literature search revealed that DSM is a good system analysis tool. It can help process improvement and manage organization communication. However, DSM is difficult to construct at early phase of the design process when system analysis is needed to make better-informed decisions about system design. DM can be built at early phase of the design process, but does not have the capability of system analysis. Therefore, in order to predict system interactions at early phase of the design process, and take advantage of DSM's system analysis and process improvement capabilities, a matrix transformation method was developed in this thesis to obtain a DSM from a DM.

The three steps of the matrix transformation are:

1. Build a DM.

2. Choose the diagonal elements as the output variables of the DM.
3. Rewrite the row headings into the same as the column headings. Then a DSM for interactions between the DP's is constructed.

The choice of the output variables implies that DM and DSM can be transformed from each other. Therefore, a more complete framework for predicting and managing system interactions is proposed in Figure 6-1. When little knowledge exists about the system interactions, we may construct a DM from requirements and obtain a DSM using the matrix transformation method. When we are only interested in the system interactions in one level of the system hierarchy, we may bypass the construction of DM and built a DSM from requirements directly. When emergent properties of the system are learned later in the product lifecycle, the contributing system interactions can be documented in the DSM and used to improve the design of the product and the management of the design process. The revised DSM can be converted back into a DM and show the effect of the emergent properties on the decomposable requirements.

The matrix transformation method was tested in two real engineering cases at CVC and Johnson and Johnson Ortho-clinical Diagnostics. In both case studies, the DSM's produced were useful in providing insights for the projects that even the experienced design engineers in the companies overlooked. Therefore, the matrix transformation method was feasible and useful in real engineering projects. It brought together the advantage of both the Axiomatic Design and Design Structure Matrix, so that we could predict the system interactions early in the design process and manage the inevitable design iterations.

However, the limitation of the matrix transformation method, like any prediction method, lacks the capability to predict the system interactions related to system emergent properties. The case in JNJ OCD further showed that not all requirements could be decomposed and hence the Axiomatic Design's DM could only show an incomplete picture of the system interactions.

Unfortunately, how incomplete the prediction DSM's really were was not investigated in this thesis research due to the difficulties in following up company projects during school semesters. However, the usefulness and the contribution of the prediction DSM's in both case studies were very positive signs for the practicality of this method.

7.1.2 Managing System Level Knowledge in An Organization

The research questions under this topic include:

Q2-a: What has been done in managing system level knowledge?
Q2-b: Is there a better way to capture, store, and represent system level knowledge?
Q2-c: What are the best sources of information for predicting system interactions?
Q2-d: How are companies doing with managing system level knowledge?
Q2-e: How to encourage engineers to document system level knowledge? Make recommendation to the management.

Literature search showed little has been done on understanding what system level knowledge is and how to manage it (Figure 2-15). The current industry practice surveyed in the author's thesis researches (Ford throttle body design, CVC ESC project, JNJ OCD OASIS project) revealed that companies do a poor job in capturing and managing system level knowledge. Companies put themselves at risk of competitiveness by relying on human experts' tacit knowledge to deal with system interactions.

This thesis proposed a framework for managing system level knowledge. As a part of the framework, a classification of system level knowledge is also recommended. Guru Prasanna's (2000) research showed that the knowledge classification was suitable for the design documents and the DSM's in Ford throttle body, CVC ESC, and CVC MOCVD projects. The knowledge classification can also serve as metrics to compare companies' practices and identify where improvements are needed.

In the JNJ CVC case study, we observed that using the matrix transformation method, the requirements documents become the best source for predicting system interactions, among all of the existing system engineering techniques. The matrix transformation framework is constructed to enable the documentation of the system level knowledge throughout the design process. The documentation of the requirements and system interactions can not only benefit the learning in the organization in the long-term, but also aid the planning of the project immediately. Therefore, there should be more incentives for documenting system level knowledge.

7.2 Contributions to Product Development Research and Practice

This thesis research contributes to both the academic research on product development and the current product development practices. The matrix transformation method bridges the Axiomatic Design and Design Structure Matrix researches, which is the contribution of this thesis to the academia. Being able to obtain a DSM from requirements added benefits to requirements management, which is already a challenging but important practice in the industries [Hooks and Farry (2001)]. The knowledge management framework can be applied to industries to improve their current status of the documentation of system level knowledge.

7.2.1 A Matrix Transformation Method to Bridge the Axiomatic Design, Design Structure Matrix, and Robust Design

The matrix transformation method developed in this thesis research bridges three existing design methods—Axiomatic Design, Design Structure Matrix (DSM), and Robust Design. This section first reviews the strengths and weaknesses of Axiomatic Design and DSM. Then how the matrix transformation method enables the above two design methods to complement each other is explained. Last, the role of robust design is introduced.

Axiomatic Design is the most advanced design theory for the design and development of large complex systems. The comparison among the representative design theories (Table 2-4) shows that Axiomatic Design is better than QFD by using the zigzagging design process to relate functional requirements to the physical design parameter. Axiomatic Design is

better than the classical European design theories by stressing the collaboration and relationship among the various departments in the organization. However, Axiomatic Design lacks the capability to deal with emergent properties of the system. This capability Axiomatic Design lacks is exactly the strength of Design Structure Matrix method. Table 2-3 and Table 3-1 have already summarized the comparison between the two methods. These two comparison tables also show that Axiomatic Design's Design Matrix (DM) can be constructed early in a design process or for a new product design, while DSM cannot.

The matrix transformation method enabled us to take advantage of both Axiomatic Design's DM and DSM. We can obtain a DSM from a DM. We can also transfer the interactions in the DSM back into DM, so as to show the impact of system interactions on the functional requirements. When we are in a position to change the design concept, we may use the requirements and DM to select the least complex system. When the ideal design in Axiomatic Design sense cannot be practically implemented, DSM and the system analysis tools associated with it can be used to plan for inevitable iterations in the system before the detailed design is carried out. As the design proceeds, more emergent system properties may be observed. The system interactions contributing to the emergent properties of the system can be documented in the DSM. DSM system analysis tools can be used to update the actions that should be taken for project management. The impact of these newly discovered system interactions on the functional requirements could be observed by reflecting the DSM marks back into DM.

Furthermore, the third design method that is to be introduced here—robust design [Taguchi (1993)]—can work with the matrix transformation method to make DM and DSM converge towards the same direction. The goal of DM and DSM are not so different as they may seem in Table 2-3. DM and DSM address the two sides of the same problem—managing system interactions. DM recognizes that if we can choose an uncoupled or decoupled engineering design, we may avoid system iterations altogether. DSM takes the existence of system iterations as a given fact and tries to minimize the impact of system iterations on design process by eliminating unnecessary iterations. Axiomatic Design's idea of avoiding system iterations altogether may be impractical for many existing product companies. Yet, the

matrix transformation method and the robust design method enable us to get closer to Axiomatic Design's goal. When inevitable iterations are identified using DSM partitioning techniques, the partitioned DSM can be converted back into the DM format. The requirements and the DP's that are involved in an iteration block are suitable for robust design study. Robust design method may be able to reduce or eliminate the effect of the off-diagonal marks for certain functional requirements, and therefore make the system less coupled. The end result is a product design that is the most uncoupled that the reality of the company allows. The rest of the coupling can be dealt with through project planning and management.

In summary, the matrix transformation method made it possible to combine the strengths of Axiomatic Design, Design Structure Matrix, and Robust Design. All three above methods have a rich body of research. Therefore, the matrix transformation method developed in this thesis enables these design methods to have even more comprehensive and bigger impact on the product development, especially for large complex systems.

7.2.2 A Requirements-driven Design Process

A requirements-driven design process is a process in which the tasks involved are scheduled for and directly support the achievement of design requirements. The matrix transformation method enables us to have a DSM built based on the requirements. The effect of DSM system analysis is to improve the design process. Re-sequencing the system elements so that they can be designed in sequence, parallel, or iteratively, eliminates the unnecessary rework in the design process. Associating the system elements in the DSM with responsible people or team enables us to have a more efficient cross-organizational communication. Hence the design process can move on more smoothly. Therefore, the design process developed based on the analysis of the DSM constructed from requirements is a requirements-driven design process.

The benefit of a requirements-driven design process is already seen in the CVC case study (see section 4.3.3.2). The DSM built from requirements identified system interactions that

were overlooked by the engineers if they were to follow how things were always done in the past. The requirements driven design process enables us to focus on the goal of the design activity—meet the requirements of the product, rather than follow the same path other people have taken.

The requirements-driven design process also encourages the documentation of requirements, because it enables us to improve the design process. Hooks has written a very interesting article on why requirements management is challenging in American Corporations [Hooks and Farry (2001), p.15]. Among many of the causes, one of the cause is that individual engineer does not understand why it is important to document requirements. Thus, a significant portion of Hooks and Farry (2001) was spent on talking about the impact of good requirements document on the project's success. The transformation between the DM and DSM introduced in this thesis research may finally enable the engineers to reap short-term benefits from documenting requirements, and hence do a better job at requirements management.

Therefore, the matrix transformation method added the benefit of design process planning and improvement to the existing list of the benefits of requirements management. This benefit of design process planning and improvement is a short-term benefit that can be felt and used by the engineers and managers in the project. Therefore, the engineers and managers will be even more motivated to do a good job in requirements management.

7.2.3 Managing System Level Knowledge

Very little research, except for a few DSM theses [Dong (1999), Glynn and Pelland (2000), and Bartowski (2000)], has been done on managing system level knowledge. This thesis research proposed a framework for managing system level knowledge. The contributions of this framework may be summarized as follows:

- This thesis proposed a way to classify system level knowledge in order to aid the capturing and representation of system level knowledge. Guru Prasanna's thesis

(2001) showed that the classification of the system level knowledge captured the documented design knowledge for Ford throttle body, CVC ESC, and MOCVD projects.

- The framework proposed in this thesis work for managing system level knowledge is associated with the most advanced design theory—Axiomatic Design. Therefore, the knowledge management effort can be carried out in parallel to the design process. In addition, the matrix transformation method enables the engineers to not only document the knowledge, but also use the DSM analysis tools to aid the design process planning. Therefore, this proposed knowledge management framework is both a documentation tool, and a process improvement method. In addition, using the Design Matrix in Axiomatic Design also enables us to combine DSM and robust design to design a better product. This proposed system level knowledge management tool is no longer a stand-alone documentation method.

7.2.4 Summary of Contributions

In summary, this thesis research has impact on both the existing product development research and practice. The matrix transformation method links three significant pieces of product design and development methods—the Axiomatic Design, the Design Structure Matrix, and the Robust Design. It enables us to take the strengths and overcome the weaknesses of each method. In addition, the matrix transformation method brings further benefits to requirements management effort. It enables the engineers to obtain immediate benefit on process and product improvements from documenting requirements. Therefore, the engineers can be more motivated in documenting requirements. Furthermore, the system level knowledge management framework is a part of the product development process, and is shown to be able to capture the documented design knowledge in several existing engineering projects. The matrix transformation method again enables the engineers to benefit immediately from documenting system level knowledge. Hence the engineers should be more motivated in documenting system level knowledge.

7.3 Future Research Directions

The learning from this thesis research has opened the door to more possible research directions. Below is a summary.

1. Test the framework in Figure 6-2 using different product development projects. Verify the hypothesis on when to use DM and when to use DSM. More details are in section 6.2.1.4.1.

2. Follow a product development project from the beginning to the end to see how the captured system interaction marks in the DSM change over the lifecycle of the product. Further verify the completeness and effectiveness of the DSM obtained at early phase of the design process through matrix transformation method.

3. The question of what makes two DSMs similar may be investigated. For more details, see Section 6.2.1.4.2.

4. This thesis took a product view of the system interactions. Future research may try to apply the matrix transformation technique build DSM's concerning project tasks, and the interactions among the people based on tasks' interactions. More discussions are in section 6.2.1.4.3.

5. Use the matrix transformation method to predict the interactions for the Process Variables (PV's) based on Design Parameters (DP's) (see Figure 2-13). The prediction of the interactions between the PV's should be able to aid the design and selection of the manufacturing process.

6. Use Datum Flow Chain (DFC) method to study the spatial relationship between components at early phase of the design process. The results of DFC studies can help to predict the spatial relationship that the requirements cannot predict using the matrix transformation method proposed in this thesis.

7. Extension of the knowledge management framework to software application. The latest web enabling technologies such as Java, Javascript, ASP, etc. should be used to create a computer platform independent environment.

References

Alexander, C. (1968*). Notes on the Synthesis of Form*, Harvard University Press, Cambridge, MA.

Allen, T. J. (1964). "The Use of Information Channels in Research and Development Proposal Preparation," *Working Paper, MIT Sloan School of Management*, No. 97-64.

Allen, T. J. (1977). Managing the Flow of Technology, Cambridge: MIT Press.

Allen, T. J. (1986). "Organizational Structure, Information Technology and R&D Productivity." *IEEE Transactions on Engineering Management*, EM-33, 4, pp212-217.

Allen, T. J. (1997). "Architecture and Communication Among Product Development Engineers," *Sloan Working Paper #3983.* Sloan School of Management: MIT.

Allen, T. J. and O. Hauptman (1987). "The Influence of Communication Technologies on Organization Structure: A Conceptual Structure for Future Research." *Communication Research*, 14(5): 575-587.

Allen, T. J. and O. Hauptman (1990). "The Substitution of Communication Technology for Organizational Structure in Research and Development," In Fulk, J. and Steinfield, C. W. (Eds.), Organizations and Communication Technology:275-294. Newbury Park CA: Sage.

Allen, T. J., D. M. Lee, and M. L. Tushman (1980). "R&D Performance as a Function of Internal Communication, Project Management, and the Nature of the Work," *IEEE Transactions on Engineering Management*, Vol. EM-27, no. 1, pp2-12, February.

Bartkowski, G. D. (2000). *Accounting for system level interaction in knowledge management initiatives.* MIT thesis for SDM program.

Bertalanffy, L. (1968). *General System Theory, Foundations, Development, Applications.* George Braziller Inc., New York, NY.

Blanchard, B. S. (1998). *System Engineering Management*. 2nd Edition. John Wiley and Sons, Inc. New York.

Boothroyd, G., P. Dewhurst, and W. Knight (1994). *Product Design for Manufacture and Assembly*. Marcel Dekker, Inc. New York, NY.

Borse, P., H. Akkermans, and J. Top (1996). "Engineering Ontologies." International *Journal of Human-computer Studies, Special Issue on Using Explicit Ontologies in KBS Development.* March.

Browning, T. R. (1998), *Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development*, Ph.D. Thesis (TMP), Massachusetts Institute of Technology, Cambridge, MA, 1998.

Browning, T. R. (1998b), "Use of Dependency Structure Matrices for Product Development Cycle Time Reduction", *Proceedings of the Fifth ISPE International Conference on Concurrent Engineering: Research and Applications*, Tokyo, Japan, July 15-17, pp. 89-96.

Browning, T. R. and S. D. Eppinger (1998), "A Model for Development Project Cost and Schedule Planning," M.I.T. Sloan School of Management, Cambridge, MA, Working Paper no. 4050, November.

Buede, D. M. (2000). *The Engineering Design of Systems, Models and Methods.* John Wiley and Sons, Inc. New York.

Carrascosa, M., S. D. Eppinger, and D. E. Whitney (1998), "Using the Design Structure Matrix to Estimate Product Development Time", *Proceedings of the ASME Design Engineering Technical Conferences (Design Automation Conference)*, Atlanta, GA, Sept. 13-16.

Chestnut, H. (1965). *Systems Engineering Tools.* John Wiley and Sons, Inc. New York.

Cho, S. (2001). *An Integrated Method for Managing Complex Engineering Projects Using the Design Structure Matrix and Advanced Simulation*, MIT MS Thesis, Mechanical Engineering.

Cho, S. and S. D. Eppinger (2001). "Product Development Process Modeling Using Advanced Simulation." *Proceedings of the 13th International Conference on Design Theory and Methodology (DTM 2001)*, September 9-12, 2001 Pittsburgh, Pennsylvania.

Clausing, D. (1994). *Total Quality Deployment, A Step-by-Step Guide to World-Class Concurrent Engineering*. ASME Press, New York.

Cohen, L. (1995). *Quality Function Deployment. How to Make QFD Work for You*. Addison-Wesley Publishing Company, Reading, Massachusetts.

Compton, W. David (1989). *Where No Man Has Gone Before: A History of Apollo Lunar Exploration Missions*. NASA SP 4212, p. 386-393. Washington D.C.

Dee, M., editor (2001a). *OASIS Architecture and Requirements Management.* Johnson and Johnso Ortho Clinical Diagnostics document OAS-SYS-ROB002, May 15.

Dee, Michael, editor (2001b). *OASIS Mechanical Interface Control Document (Draft).* Johnson and Johnson Ortho Clinical Diagnostics document OAS-SYS-ROB218, May 14.

Dee, Michael, editor (2001c). *OASIS System Product Requirements*. Johnson and Johnson Ortho Clinical Diagnostics document OAS-SYS-ROB001, March 29.

Dong, Q. (1999). *Representing Information Flow and Knowledge Management in Product Design Using Design Structure Matrix*. MIT Mechanical Engineering Master of Science degree thesis.

Eppinger, S. D., D. E. Whitney, R. Smith, and D. Gebala (1994). "A Model-based Method for Organizing Tasks in Product Development." Research in Engineering Design, Vol. 16, 1-13.

Eppinger, S. D., D. E. Whitney, R. P. Smith, and D. Gebala (1990), "Organizing the Tasks in Complex Design Projects," *ASME Conference on Design Theory and Methodology*, Chicago, IL, September, pp. 39-46.

Fine, C. (1998). *Clockspeed : Winning Industry Control in the Age of Temporary Advantage.* Perseus Books, Reading, Massachusetts.

Finger, S. and J. R. Dixon (1989). "A Review of Research in Mechanical Engineering Design Parts 1 & 2." *Research in Engineering Design*, 1.

Gershenson, J. A., D. V Khadilkar, and L. A Stauffer. "Organization and Managing Customer Requirements During the Product Definition Phase of Design." DE-vol. 68, *Design Theory and Methodology--DTM'94*, ASME 1994.

Getner, D. Bolt Beranek and Newman Inc. (1983). "Structure-Mapping: A Theoretical Framework for Analogy." *Cognitive Science*, 7, pp.155-170.

Gick, M. L. and K. J. Holyoak (1980). "Analogical Problem Solving." *Cognitive Psychology*, 12, pp. 306-355.

Glynn, S. V and T. G. Pelland (2000), *Information Flow & Knowledge Capture: Lessons for Distributed Integrated Product Teams.* MIT Thesis for SDM program.

Gonzalez-Zugasti, Javier P., Kevin Otto, John D. Baker. (1998) "A method for Architecting Product Platforms with an Application to Interplanetary Mission Design." Proceedings of 1998 ASME Design Automation Conferences, September 13-16, 1998, Atlanta, GA.

Gonzalez-Zugasti, J. P., K. Otto, and J. D. Baker (1999) "Assessing Value for Product Family Design and Selection." Proceedings of the 25th Design Automation Conference, 1999 ASME Design Engineering Technical Conferences, September 12-15, 1999, Las Vegas, Nevada.

Grady, J. O. (1993). System Requirements Analysis, McGraw-Hill, Inc., New York.

Griffin, A. and J. R. Hauser (1992). "Patterns of Communication Among Marketing, Engineering, and Manufacturing –A Comparison between Two New Product Teams." Management Science. 38: (3) 360-373, March.

Guindon, R. (1990) Microelectronics and Computer Technology Corporation. "Designing the Design Process: Exploiting Opportunistic Thoughts." *Human-Computer Interaction*, 1990, Volume 5, pp 305-344.

Hall, A. D. (1962). *A Methodology for Systems Engineering*. D. Van Norstand Company, Inc. Princeton, New Jersey.

Henderson, R. M. and K. B. Clark (1990). "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms." *Administrative Science Quarterly*, 35: p 9-30.

Hooks, I. and K. A. Farry (2001). *Customer-centered Products. Creating Successful Products Through Requirements Management*. AMACOM, Boston, MA.

Hubka V. (1980). Translated and edited by W. E. Eder. *Principles of Engineering Design*. Butterworth Scientific, Boston.

Hughes, A. C., and P. H. Thomas (2000). *Systems, Experts, and Computers.* The MIT Press, Cambridge, Massachusetts.

Hughes, T. P. (1998). *Rescuing Prometheus.* Vintage Books, a division of Random House, Inc. New York, NY.

Hutton, J. B. and G. Klein (1999). "Expert Decision Making." *INCOSE Systems Engineering*, January.

Kaufmann, W. J., and R. A. Freedman (2000). *Universe*, 5th Edition, W. H. Freeman Company.

Koch, C., and G. Laurent (1999). "Complexity and the Nervous System," *Science*, Vol 284, April.

Kuffner, T. A., and D. Ullman (1991). "The Information Requests of Mechanical Design Engineers." *Design Studies*, Vol. 12 No. 1, January. Butterworth-Heinemann ltd.

Mantripragada, R., and D. E. Whitney (1998). "The Datum Flow Chain," Research in Engineering Design. Vol 10, No. 1.

Martin, J. N. (1997). *Systems Engineering Guidebook: A Process for Developing Systems and Products*. CRC Press, New York.

Miller, G. A. (1956), "The Magic Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information." *The Psychological Review*, Vol. 63, pp81-97.

Mindell, D. A. (2000). "Opening Black's Box, Rethinking Feedback's Myth of Origin." *Technology and Culture*, Vol. 41, July.

Moder, J. J., E. W. Davis, and C. R. Phillips (1995). *Project Management with CPM, PERT and Precedence Diagramming*. 3rd edition. Blitz Publishing Company.

NASA Apollo Mission—Apollo 13, November 13, 2001. http://Science.ksc.nasa.gov/history/apollo/apollo-13/apollo-13.html

Noy, N. F. and C. D. Hafner (1997). "The State of the Art in Ontology Design: A survey and Comparative Review." American Association for Artificial Intelligence, 0738-4602-1997. http://www.aaai.org.

Otto, K., and K. Wood (2000). *Product Design: Techniques in Reverse Engineering and New Product Development.* Prentice Hall, Upper Saddle River, NJ.

Pahl, G. and W. Beitz (1995). Translated by K. Wallace, L. Blessing, and F. Bauert. Edited by K. Wallace. *Engineering Design, A Systematic Approach*. 2nd Edition. Springer-Verlag, New York.

Parker, G. G. and E. G. Anderson Jr. (2000) "From Buyer to Integrator: The Transformation of the Supply-chain Manager in the Vertically Disintegrating Firm." Submitted to a Special Issue of the *Journal of Production and Operations Management on Case-Based Research in Operations Management*, January 19.

Parrish, J. K., and L. Edelstein-Keshet (1999). "Complexity, Patter, and Evolutionary Trade-offs in Animal Aggregation," *Science*, Vol 284, April 1999.

Patil, S. (2000). *Mapping the product development process to IT solutions through use models*. MIT thesis for SDM degree.

Rechtin, E. (1991). *Systems Architecting*. Prentice Hall, New York, NY.

Ritchie, S. W. (1999). *Rescuing Endagngered Knowledge: A System Approach.* MIT Thesis for SDM degree.

Sage, A. P., and J. E. Armstrong Jr. (2000). *Introduction to Systems Engineering*. John Wiley and Sons, Inc. New York.

Sage, A. P., and W. Rouse (1999). Handbook for Systems Engineering and Management. John Wiley and Sons, Inc., New York.

Senge, P. M. (1994). *The Fifth Discipline: The Art and Practice of the Learning Organization*. Doubleday and Company Inc.

Senin, N., D. R. Wallace, and N. Borland (2000). "Distributed Object-based Modeling in Design Simulation Marketplace." Address correspondence to drwallac@mit.edu.

Service, R. F. (1999). "Exploring the Systems of Life," *Science*, Vol 284, April.

Sferro, P. R. (1999). Kamrani, K, editor. Direct Engineering: Toward Intelligent Manufacturing. Kluwer Acdemic, Boston, MA.

Simon, H. A. (1957). *Administrative Behavior: A Study of Decision-making Process in Administrative Organizations*, 2nd ed., MacMillian, New York, NY.

Simon, H. A. (1981). *The Science of the Artificial*. Second Edition. MIT Press, Cambridge, MA.

Simpson, T. (1998). *A Concept Exploration Method for Product Family Design*. Doctoral Thesis, Georgia Institute of Technology, September 1998.

Smith, R. P., and S. Eppinger (1997). "Identifying Controlling Features of Engineering Design Iteration." *Management Science*, vol. 43, pp. 276-293, 1997a.

Sosa, M. E. (2000). *The Effects of Product Architecture on Technical Communication in Product Development*. MIT Ph.D. Thesis, Mechanical Engineering Department.

Sterman, J. D. (1989). "Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamics Decision Making Experiment." *Management Science*, 35(3), p321-339.

Sterman, J. D. (2000). Business Dynamics—Systems Thinking and Modeling for a Complex World. Irwin McGraw-Hill, Boston, MA.

Steward, D. V., (1962). "On an Approach to Techniques for the Analysis of the Structure of Large Systems of Equations," *SIAM Review*, Vol. 4, No. 4, October.

Steward, D. V. (1965). "Partitioning and Tearing Systems of Equations", *SIAM Numerical Analysis*, ser. B, vol. 2, no. 2, pp. 345-365.

Steward, D. V. (1981). *Systems Analysis and Management, Structure, Strategy, and Design*. Petrocelli Books, Inc.

Stevens, R., P. Brook, K. Jackson, and S. Arnold (1998). *Systems Engineering: Coping with Complexity*. Prentice Hall Europe, New York.

Strang, G. (1986). *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, MA.

Systems Biology Internet sites, November 6, 2001:
http://www.systemsbiology.org/workwhat.html
http://www.icsb2001.org/what_is.html

Suh, N. P. (1990). "The Principles of Design." Oxford University Press, New York.

Suh, N. P. (1999). "A Theory of Complexity, Periodicity and the Design Axioms." *Research in Engineering Design*, Vol. 11, p.116-131.

Suh, N. P. (2000). *Axiomatic Design: Advances and Applications*. Oxford University Press.

Taguchi, G. (1993). *Taguchi on Robust Technology Development: Bringing Quality Engineering Upstream*. ASME Press, New York.

Tate, D. and M. Nordland (1995). "Synergies between American and European Approaches to Design." *Proceedings of the First World Conference on Integrated Design and Process Technology*, Society of Design and Process Science, 1995.

Tate, D. (1999). *A Roadmap for Decomposition: Activities, Theories, and Tools for System Design*. MIT PhD thesis for Mechanical Engineering Department.

Thebeau, R. E. (2001). Knowledge Management of System Interfaces and Interactions for Product Development. MIT SDM thesis.

Thome, B. (1993). *Systems Engineering: Principles and Practices of Computer-based Systems Engineering*. John Wiley and Sons, Inc. New York.

Tong, C. and D. Sriram, editors (1992). *Artificial Intelligence in Engineering Design, Volume I, II, and III*. Academic Press, Inc., Boston.

Ullman, D. G. (1995). "Engineering Decision Problems and Support Systems."*Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. Vol 9. pp427-438.

Ullman, D. G., Stauffer, and Dietterich (1983). "Toward Expert CAD." *Computers in Mechanical Engineering*, Nov-Dec. pp56-70.

Ullman D. G, D. Herling, and B. D'Ambrosio (1997). "What to do Next: Using Problem Status to Determine the Course of Action." Research in Engineering Design, Vol. 9, pp. 214-227.

Ulrich, K. T., and S. D. Eppinger (2000). *Product Design and Development*. Irwin McGraw-Hill, Boston, MA.

Ulschold M. and M. Gruninger (1996). "Ontologies: Principles, Methods, and Applications." *Knowledge Engineering Review*. Vol. 11, No. 2, June.

Warfield, J. N. (1973). "Binary Matrices in System Modeling." *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, pp. 441-449.

Warfield, J. N. (1976). *Societal Systems: Planning, Policy, and Complexity*. John Wiley and Sons.

Warfield, J. N. (1995). "Spreadthink: Explaining Ineffective Groups." *Sytems Research*, Vol. 12, No. 1, pp 5-14.

Warfield, J. N. (1999). "Twenty Laws of Complexity." *Systems Research and Behavioral Science*, 16, pp 3-40.

Walton, M. (1997). *Car: A Drama of the American Workplace*, Norton, New York.

Weist, J. D. and F. Levy (1990). *Management Guide to PERT-CPM*. 2nd Edition. Prentice Hall.

Weng, G., U. S. Bhalla, R. Lyengar (1999). "Complexity in Biological Signaling Systems," *Science*, Vol 284, April.

Werner, B. T. (1999), "Complexity in Natural Landform Patterns," *Science*, Vol 284, April.

Westman, H. R. (2001). *Systems Engineering Principles and Practice*. Artech House, Boston, MA.

Whitney, D. E. (1993). "Nippondenso Co. Ltd: A Case Study of Strategic Product Design." *Research in Engineering Design*, Vol 5. Pp.1-20.

Whitney, D. E., Q. Dong, J. Judson, C. Mascoli (1999). "Introducing Knowledge-based Engineering into an Interconnected Product Development Process." *Proceedings of the 1999 ASME Design Engineering Technical Conference*, September, Las Vegas, Nevada. DETC99-DTM8741.

Wilson, E. B. (1952), *An Introduction to Scientific Research*, McGraw-Hill Book Company, Inc. New York.

Wisnosky, D. E., A. W. Batteau (1990), “IDEF in Principle and Practice.” *GATEWAY*, May/June, pp. 8-11.

Wood, K. L. and K. N. Otto (2001), *Product Design, Techniques in Reverse Engineering and New Product Development*. Prentice Hall, Upper Saddle River, NJ.

Zimmer, C. (1999), “Life After Chaos”, *Science*, Vol 284, April.

Appendix A: OASIS Product Requirements

Note that not all OASIS product requirements are listed here. Only the requirements used as examples in this thesis are copied in this Appendix.

PRD1 The Oasis system shall perform Wet Assays to the following standards:

TDM Assays per 17-SP-72-002
Protein Assays per 17-SP-72-003
H.D. Lipoprotein Cholesterol and L.D. Lipoprotein Cholesterol Assays per 17-SP-72-004
%HbA1c Assay per 17-SP-72-005

PRD4 Sample integrity testing *should* require no increase in the volume of aspirated sample.

PRD6 The Operator shall be able to disable sample integrity checking for any particular sample.

PRD16 The Oasis system *should* be designed to maximize its sustained throughput rate against the US and European Core Lab Test Distributions as defined in *OASIS Timing and Throughput Model;* OAS-SYS-DA1004, with the following design goals for non-diluted samples:

Test Distribution	Design Goal (Average Tests/Hour)
X1 Lab	XXX
X2 Lab	XXX

PRD21 The system *should* be capable of performing all assays within specifications within both of the following intervals:

X minutes from a cold startup at an ambient lab temperature of 20°C (68°F)
X minutes from a cold startup at an ambient lab temperature of 15°C (59°F).

PRD34 The multiple levels of severity used on all Vitros products shall be utilized. In order of increasing severity, they are:

Level 1
Level 2
Level 3
Level 4
Level 5

PRD43 There shall be only one location where the Oasis system performs sample metering from a sample on an LAS automation track.

PRD44 The design *should* allow the standard OASIS system to be changed in the field to an OASIS AT configuration.

PRD46 The packaged system and components shall withstand the following non-operational environment without degrading performance.

Cold: -X C at Y RH.
Heat: X C at Y RH.
Humidity: Y RH at X°C.
Altitude: M to N m.

PRD48 The analyzer *should* have a total weight of less than X kg.

PRD49 The system shall utilize power sources with any combination of standard voltages from X to Y Vac and frequencies from M to N Hz, using regionally standard power cords.

PRD51 A system *should* require no more than X amperes at steady state.

PRD54 At least one UPS shall be available for all regions where the Oasis analyzer is sold.

PRD57 As packaged for shipping, the Oasis system shall perform normally after being subjected to shipping environments as specified in Performance Testing Protocol, Packaging and Shipping Containers, Test Standard 1001, dated 9/19/97

PRD58 Oasis system *should* be designed to meet the reliability targets defined in document OAS-SYS-ROB201, *Oasis Subsystem Reliability Allocation*.

PRD77 Access to system functions shall be based on security levels: General Operator, Key Operator, Field Engineer, and Remote Diagnostics.

PRD95 The system shall be able to read X% of all correctly oriented bar code labels on sample containers to a Y% confidence level.

PRD120 The system shall provide *internal storage for support* fluids having their own packaging systems (e.g. IR wash fluid, ERF fluid, Surfactant wash fluid).

PRD123 Reagents shall be available in 60 ml wedges and 20 ml bottles, capable of being packaged as two bottles, or one bottle and one wedge.

PRD134 The system shall have on-board capacity for at least X cartridges supporting both A-type and B-type slide cartridges.

PRD139 The system shall have the capacity to store the following consumables:

X Vitros tips
X Micro Volume tips
X Reaction cuvettes

PRD146 Protocols and other information required for processing assays shall be independent of system software versions or releases.

PRD187 All waste containers shall be accessible from the front of the system.

PRD256 The operator and/or field service person shall be able to exercise individual subsystem mechanisms through their full set of basic movements (e.g. extending a blade, rotating a metering probe, # of repetitions, etc.).

PRD269 The system shall be certifiable for the European operator safety standard: Low Voltage Directive 73/23/EEC; EN61010-1.

PRD280 The system shall be certifiable for the electromagnetic standard: EN 61000-4-4 (Electrical Fast Transient).

PRD287 Installation time for a standard Oasis system (not including system configuration and verification in the customer's lab environment with the customer's assays) *should* require less than A hours for a single body system, or less than B hours for a split body system.