Complexity analysis of human-machine interaction based on principles of axiomatic design

Lo, Shuan

2005

Lo, S. (2005). Complexity analysis of human-machine interaction based on principles of axiomatic design. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/6102>

<https://doi.org/10.32657/10356/6102>

Nanyang Technological University

Downloaded on 26 Apr 2023 08:47:39 SGT

**Complexity Analysis of Human-Machine Interaction
based on Principles of Axiomatic Design**

Lo Shuan

School of Mechanical & Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Master of Engineering

2005

SUMMARY

This research explores the use of Axiomatic Design (AD) concepts for analyzing coupling in human-machine systems. Coupling is a generic shaping factor of complexity, and coupling between user goals and user actions widens the gulf of execution of a human-machine system. By providing a general and rational criterion, this analytic method is proposed as a tool that design engineers can use for an early evaluation of usability.

A framework for modeling human-machine systems was constructed based on the concept of design domains. This framework consists of user goals, functional requirements, design parameters, and user actions. User goals and user actions describe a human-machine system from a user's perspective; functional requirements and design parameters describe from an engineering perspective. Including the system's structure in an analysis allows one to identify problems that lie in the inner workings of a system.

Design equations are used for representing the interactions between the design attributes, and the form of a design matrix is used for characterizing the degree of coupling. Several case studies were performed to understand the generalizability of the proposed method.

In the latter part of this thesis, the use of AD concepts to analyze interactions between usability guidelines was explored. Guidelines provide a set of generic design requirements, and coupling in design requirements results in complexity for software user interface designers. Nielsen's usability heuristics were used as a case study. The recommendations in the heuristics were distinguished as design goals and design solutions, and the dependencies between them were represented using a design equation. The feasibility of reducing coupling in the heuristics by manipulating the design matrix in the design equation and reengineering the heuristics was explored.

ACKNOWLEDGEMENT

I owe my greatest debt of gratitude to Professor Martin Helander for guiding me throughout this postgraduate program. With great patience, he taught me the basic skills for doing research, and through his own writings, he showed me how a complex idea can be expressed in simpler words. His encouragements have been my main source of motivation for completing this thesis.

I would like to thank Dr. Roger Jiao Jianxin for spending many hours to discuss my research with me. I am also grateful to Professor Nam Suh for answering some of my questions regarding Axiomatic Design. In addition, I wish to express my thanks to Professor John Long for giving me some constructive recommendations on writing this thesis.

I thank the staff and students who work in the Design Research Centre and the Centre for Human Factors and Ergonomics, in particular, Darius Alisantoso and Rudi. They have been helpful and encouraging to me, and their listening ears have helped to keep me sane throughout the course of this research. Finally, I want to express special gratitude to my family and Shihling for giving me this opportunity to pursue a postgraduate study.

This research is partially funded by a grant from the U.S. Air Force Research Laboratory, under agreement number FA5209-04-P-0320.

I am solely responsible for the arguments that are put forward in this thesis, and the views that are expressed in this thesis should not be attributed to those who are mentioned above.

TABLE OF CONTENTS

List of Figures	vi
------------------------------	-----------

List of Tables	ix
-----------------------------	-----------

Chapter 1. Introduction	1
--------------------------------------	----------

1.1 Increasing Importance of Usability	1
1.2 Lack of General Methods for Early Evaluation.....	2
1.3 Lack of Tools for Analyzing Guidelines Interactions.....	6
1.4 Axiomatic Design.....	9
1.5 Research Purpose and Objectives.....	11
1.6 Organization of Thesis	11
1.7 Chapter Summary	14

Chapter 2. Literature Review.....	15
--	-----------

2.1 Models of Human-Machine Interaction	17
2.1.1 Norman's Model of Human Action.....	17
2.1.2 Weir's Pyramid of Complexity.....	19
2.2 Analytic Usability Evaluation Methods	20
2.2.1 Hierarchical Task Analysis	21
2.2.2 Task Analysis for Error Identification.....	22
2.2.3 Goals, Operators, Methods, and Selection	24
2.2.4 Task-Action Grammar	26
2.2.5 User Interface Design with Matrix Algebra.....	27
2.2.6 Discussion	28
2.3 Complexity Methods in Human-Machine Interaction	29
2.3.1 Cognitive, Behavior, System, and Task Complexity	30
2.3.2 Cognitive Complexity Theory.....	31
2.3.3 Task-To-Action Model	32

2.3.4 Framework for Quantification of Complexity and Compatibility	35
2.4 Axiomatic Design	37
2.4.1 Axiomatic Design as a Framework for Managing Complexity of Design	37
2.4.2 Axiomatic Design Applications in Human-Machine Interaction	41
2.5 Effects of Coupling on Human Problem Solving	44
2.6 Chapter Summary	47
Chapter 3. Task Complexity Analysis.....	49
3.1 Overview	49
3.2 Duel-Domain Framework	50
3.2.1 Matrix Reordering Algorithms	57
3.2.2 Redundant Designs	60
3.2.3 Functional Analysis and Actual User Behavior	62
3.2.4 Hierarchical Structure of User Tasks	63
3.3 Quartet-Domains Framework	68
3.3.1 Microscope Workstation	72
3.3.2 Manual Single-Lens-Reflex Film Camera	74
3.3.3 Manual Point-and-Shoot Film Camera	78
3.4 Method Procedure	79
3.5 Analysis of a Process Control System	83
3.5.1 Introduction to Duress II	84
3.5.2 Analysis Procedure and Results	86
3.5.3 Discussion	96
3.6 Discussion	98
3.6.1 An Extension of Axiomatic Design	98
3.6.2 Comparison with Existing Methods	99
3.7 Chapter Summary	102
Chapter 4. Usability Guidelines Analysis	105
4.1 Role of Guidelines in Software Design	105
4.2 Technique for Decoupling Usability Guidelines	108
4.2.1 Identification of Dependencies	111

4.2.2 Reengineering the Usability Heuristics	117
4.3 Research Limitations	122
4.4 Chapter Summary	123
 Chapter 5. Conclusion and Recommendation	125
5.1 Axiomatic Design for Analyzing Human-Machine Interaction	125
5.2 Analyzing Interactions in Usability Guidelines.....	126
5.3 Limitations and Recommendations	127
References	129
 Appendix 1. Axiomatic Design	140
A1.1 Fundamentals of Axiomatic Design	140
A1.2 Advance Issues	144
A1.2.1 Hierarchical Structures in Axiomatic Design	145
A1.2.2 Distinguishing Abstraction Levels	150
 Appendix 2. Matrix Reordering Algorithms.....	152
A2.1 Steward's Algorithm	152
A2.2 King's Algorithm.....	154
 Appendix 3. List of Variables in Duress II.....	156
 Appendix 4. List of Acronyms	157

LIST OF FIGURES

Figure 1.1. Usability is a property of the interaction between user, task, product, and environment (adapted from Shackel, 1991).....	2
Figure 1.2. Classes of usability evaluation methods (adapted from Whitefield et al., 1991)...	4
Figure 1.3. Structure of thesis.....	13
Figure 2.1. Structure of literature review.....	16
Figure 2.2. An reinterpretation of Norman's (1988) model of human action.	18
Figure 2.3. Usability depends on the design of the human-machine interface, which is constrained by the design of the control mechanism, which is constrained by the design of the physical process (adapted from Weir, 1991).....	20
Figure 2.4. Example of a hierarchical task decomposition diagram (Annett, 2004).	22
Figure 2.5. A TAFEI diagram for changing the date on a digital watch (Baber and Stanton, 2004).....	23
Figure 2.6. A transition matrix for changing the date on a digital watch (Baber et al., 2004).	24
Figure 2.7. Example of a poor task-device mapping (Kieras et al., 1999).	32
Figure 2.8. Structure of an exemplary information system (Kang et al., 1998).	34
Figure 2.9. The complexity-compatibility quantification framework (Karwowski, 2003). ...	36
Figure 2.10. A constructive diagram can be used to represent the interactions between the requirements of a mechanical design problem (Alexander, 1964).	40
Figure 2.11. Example of an interaction matrix that shows interactions, which vary in strength, between components of a hospital design (Jones, 1980).	40
Figure 2.12. GUI for a 3×3 parameter design problem (Hirschi et al., 2002).	45
Figure 2.13. Effects of coupling on parameter design task completion time (Hirschi et al., 2002).	46
Figure 3.1. User task design can be represented by the mapping from the goal domain to the action domain.....	51
Figure 3.2. A water faucet that has two knobs for controlling, individually, the flow of hot and cold water.....	53

Figure 3.3. A water faucet that has a single lever for controlling the flow and the temperature of the water. Lifting the lever increases flow, and turning the lever towards the left raises temperature.	54
Figure 3.4. A user's lack of understanding of the interactions in a human-machine system may result in unnecessary task iterations.	63
Figure 3.5. User-task decomposition results in a goal hierarchy and an action hierarchy.	64
Figure 3.6. A master design matrix, which takes a tabular form, can be used to represent mappings across hierarchical levels.	65
Figure 3.7. UGs to UAs mapping for the traditional washing machine.	67
Figure 3.8. Quartet-domains model for analyzing user task designs.	69
Figure 3.9. A systematic procedure for analyzing a human-machine system based on the quartet-domains model.	80
Figure 3.10. Coupling between the functional (FR ^u s) and physical (DP ^u s) domains results in complexity for user. Complexity can be reduced by making localized changes within these two domains, but a modified system needs to be reanalyzed to ensure that the design changes do not introduce any new and unexpected interactions.	82
Figure 3.11. The multiplying effect of two or more semi-coupled mappings may result in overall full coupling. In this case, since complexity is an emergent property of the system, it is unlikely that the problem can be corrected by making localized design changes only.	82
Figure 3.12. A schematic diagram of the Duress II process control microworld which shows the various components and their topological connections (from Vicente, 1999).	85
Figure 4.1. Software design can be conceived as a mapping process across four design domains (adapted from Suh, 2001).	106
Figure 4.2. The role of usability guidelines in software design can be represented by adding two design domains to the Axiomatic Design framework.	107
Figure 4.3. van Welie et al.'s (1999) model of usability consists of four layers: components from the ISO definition of usability, usage indicators which are operational measures of usability, the means for improving the usage indicators, and designer's knowledge of usability.	112
Figure 4.4. Nielsen's (1994b) set of usability heuristics appears to be a coupled system. The dashed boxes indicate clusters.	114

Figure A1.1. Axiomatic Design's framework consists of four design domains (adapted from Suh, 2001)	141
Figure A1.2. The concept of zigzagging states that higher level FRs should not be decomposed into lower level FRs before the higher level DPs have been conceptualized (adapted from Suh, 2001).	142
Figure A1.3. Cognitive systems engineering typically uses a combination of a whole-part hierarchy and a means-ends hierarchy to represent work domains (from Rasmussen et al., 1994).	145
Figure A1.4. The triadic characteristic of a means-ends hierarchy.....	146
Figure A1.5. Each triplet of adjacent design domains has a “why-what-how” form of means-ends relationship.	147
Figure A1.6. The functional hierarchy of a lathe system (from Suh, 1990).....	147
Figure A1.7. The physical hierarchy of a lathe system (from Suh, 1990).....	148
Figure A1.8. The Axiomatic Design framework consists of a horizontal hierarchy and a vertical hierarchy.....	150
Figure A2.1. A simple production flow analysis matrix (Gallagher and Knight, 1986).	155
Figure A2.2. The reorganized production flow analysis matrix (Gallagher et al., 1986).....	155

LIST OF TABLES

Table 2.1	Definitions of complexity according to Rauterberg (1996).....	30
Table 3.1	Three general feedwater configuration strategies (adapted from Vicente, 1999). ..	86
Table 3.2	List of design variables in configuration strategy A.	87
Table 3.3	List of design variables in configuration strategy B.	90
Table 3.4	List of design variables in configuration strategy C.	92
Table 3.5.	A new set of functional requirements, design parameters, and user actions for configuration strategy B.	95
Table 4.1	Nielsen's (1994b) list of 10 usability heuristics.	110
Table 4.2	FR ^g s, DP ^g s, and Cs from Nielsen's (1994b) list of usability heuristics.	111
Table 4.3	A brief explanation for each of the dependencies.	115
Table 4.4	Reengineered set of FR ^g s and DP ^g s.....	122
Table A1.1	Decomposition of a Newcomen Steam Engine (Suh, 2001).	149
Table A1.2	Decomposition of a refrigerator (Suh, 2001).....	149

CHAPTER 1. INTRODUCTION

This chapter presents the background and motivation of this research. It also states the research purpose and objectives, and describes the overall structure of this thesis.

1.1 Increasing Importance of Usability

Technological advancement allows manufacturers to produce systems and products that can perform more functions and more sophisticated functions, and the pace of technological advancement has been accelerating in recent decades. However, while the functionality of products has been advancing, arguably the usability has not. Many researchers have stated that the systems and products that we use, for work and for recreation, are becoming too complex (e.g., Jensen, 2003; Perrow, 1984; Weir, 1991).

The complexity of modern technology is also fueled by manufacturers who use the number of functions that their products can provide as a sales argument. Management often assumes that building more functions into a product will make the product more competitive in the market. Consequently, designers have the tendency to build more functions into a product than what was originally conceived at the beginning of the development process (Norman, 1988).

As systems and products become increasingly difficult to use, customers are beginning to perceive usability as an important criterion for making purchasing decisions (Shackel, 1991). Consumers are showing less tolerance to products that are rich in functionality but poor in usability (Han et al., 2000). Hence, the design of usable products is becoming an important subject in academia as well as in industry.

Operationally, usability is defined as the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use (ISO 9241-11, 1998). Effectiveness refers to the accuracy and the completeness with which users achieve the specified goals. Efficiency refers to the resources expended in relation to the accuracy and the completeness with which users achieve the specified goals. Satisfaction is related to the comfort and acceptability of use.

It is important to note that usability is not a property of a product; rather, it is a property of the interaction between a product, a user, a set of tasks that he or she is trying to complete, and the environment (Shackel, 1991), see Figure 1.1.

Figure 1.1. Usability is a property of the interaction between user, task, product, and environment (adapted from Shackel, 1991).

1.2 Lack of General Methods for Early Evaluation

Gould et al. (1991) stated the following tenets for designing usable products:

- 1) Early focus on users. Design should begin with a good understanding of user needs.

- 2) Integrated design. To maintain consistency, different aspects of a design, such as the internal design and the user interface, should evolve concurrently.
- 3) Early and continual user testing. User feedback is important throughout a development process.
- 4) Iterative design. A single round of design and evaluation is not sufficient for ensuring usability.

Researchers in human-computer interaction (HCI) generally agree with these principles (e.g., Mayhew, 1999).

User testing is commonly considered the “gold standard” for usability assessment (Kieras, 1997). It refers to empirical techniques that are used for evaluating the usability of a product and involve the participation of potential end-users. However, user testing is not without problems. It is often slow and therefore incompatible with current product and software development schedules. Any design change, even one occurring in a very late iteration, can lead to new, unforeseen problems (Vicente, 1999). In addition, if a design has been substantially altered, it will be necessary to retest the design with a new set of users. Furthermore, a combination of iterative design and user-based evaluation can require much time and resources. Some usability problems may require changes at the upstream of design, and such changes are particularly costly.

Hence, there is a need for evaluation methods that can be used to identify potential usability problems while a design solution is still at a conceptual stage. If some of the problems can be identified in the early stages of design, changes can be made to the design concept before further development proceeds.

Whitefield et al. (1991) proposed a framework for classifying evaluation methods (see Figure 1.2). This framework is based on the type of user and computer components, either real or representational, that are used in the evaluation process. “Real computer” refers to the physical presence of a computer or an approximation of it, while “representational computers” refers to symbolic representations, such as specification models, notational models, and user’s mental representations. “Real users” refers to

actual or potential users, while “representational users” refers to descriptions or models of users.

		User	
		<i>Representational</i>	<i>Real</i>
Computer	<i>Representational</i>	Analytic Methods	User Reports
	<i>Real</i>	Specialist Reports	Observational Methods

Figure 1.2. Classes of usability evaluation methods (adapted from Whitefield et al., 1991).

Analytic methods are evaluation methods that do not involve real users and real systems. They usually require some expertise and the output is a prediction, rather than an actual measurement, of usability. Nevertheless, these methods are particularly useful during the early stages of design, because working prototypes and user subjects are not required.

Two examples of analytic usability evaluation methods are Goals, Operators, Methods, Selection (GOMS) and Task-Action Grammar (TAG) (Card et al., 1980; Payne and Green, 1989). For describing a task and the user’s knowledge, GOMS uses a notation with syntax and semantic similar to traditional programming languages. The results of a GOMS analysis can be used to predict human learning and performance of a user

interface design. For example, the number and the type of keystrokes can be used to predict the time required to perform a task, and the number of steps in the analysis can be used to predict learning time (Kieras, 2004).

TAG is a semi-formal method that analyzes the consistency of a user interface's input language, which facilitates learning, improves retention, and prevents negative transfer. Chapter 2 provides a more detailed review of existing analytic methods.

However, there are several deficiencies in the existing pool of analytic usability evaluation methods:

- 1) Preece et al. (1994) distinguished between structural and functional models of interaction design. Structural models describe the internal mechanisms of a device, while functional models represent the internalized procedural knowledge necessary for using a device. Design decisions in either one of the two models constrain the design options that are available in the other; the two models are coupled (Lee and Yoon, 2004).
However, most of the existing methods, such as GOMS and TAG, evaluate the functional model without considering the structural model. Therefore, these methods may be able to identify the presence of usability problems, but they are not able to identify the structural design decisions that lead to these problems.
- 2) Some of the methods, such as Hierarchical Task Analysis (HTA), do not provide an explicit and general criterion for judging the usability of a design (Annett et al., 1971). It is up to the analyst's knowledge and experience to decide whether a design is acceptable or not. Hence, judgments may be inconsistent.
- 3) Most of the existing methods, such as GOMS and TAG, are designed more for evaluating software-based systems and products. There is a lack of methods for general hardware-based products and process control systems.

Therefore, there is a need to develop new analytic usability evaluation methods to complement the existing pool of methods in the human factors and human-computer

interaction literature. In this research, an analytic method has been developed to meet the following requirements:

- 1) Analyzes and identifies potential usability problems in a set of user tasks. The framework, procedure, and notation of the method should assist a design engineer to analyze the structure of a set of tasks and to identify potential problems.
- 2) Identifies the engineering design decisions that result in those potential problems. The framework of the method should enable a design engineer to identify the underlying structure of a system or product that leads to potential usability problems at the user interface.
- 3) Facilitates rational discussions among members of a design team by providing an explicit notation and a systematic framework. The basic idea of the design methodology movement is to develop systematic external methods and tools to carry out logical design analysis better, and to unburden the designer to engage in the creative aspects of idealization (Cross, 1984). Furthermore, researchers have reported that it is usually difficult for designers and users to reason reliably about user interfaces (Thimbleby, 2004). An explicit notation and a systematic framework would also facilitate teaching and further development of the method.
- 4) Facilitates rational and consistent evaluations by providing a general criterion for judging the usability of a design. Providing a general criterion for evaluation would lessen the dependency on the experience of individual design engineers.

1.3 Lack of Tools for Analyzing Guidelines Interactions

Usability guidelines refer to design and evaluation principles or rules that should be observed in order to achieve usability (Vanderdonckt, 1999). They are widely used to as a source of knowledge for developing and evaluating software user interface (Henninger, 2001).

Previous publications, based on psychological theories as well as practical experience, have produced a large number of user interface guidelines. One of the largest collections of publicly available guidelines was published by Smith and Mosier (1986); it contains 944 guidelines. Other examples of guidelines are Bastien and Scapin (1993), Mandel (1998), Mayhew (1992), Nielsen (1994a; 1994b), and Shneiderman (1987). The International Organization for Standardization published a set of 17 standards for design and evaluation of dialogues for visual display terminals (Helander, 2005). In the industry, major software and computer companies, such as Apple, IBM, and Microsoft have also published their own sets of guidelines (Apple, 2002; IBM, 2003; Microsoft, 2003).

While it is generally accepted that usability guidelines cannot replace user testing, guidelines can play a role in improving the quality of the iterative design process and lead to a reduction of the number of iterations in the design-evaluate-redesign cycle of user interface development (Strong, 1994). According to de Souza and Bevan (1990), guidelines can be used in at least three different manners:

- First, they can be used as a compilation of human-computer interaction (HCI) knowledge and provide human factors professionals with an authoritative source of advice for designers.
- Second, they can be used as means of transferring knowledge to designers in educational or training courses.
- Third, they can be used as a direct source of reference and guidance for designers during the design process.

However, while usability guidelines aim to improve usability, the guidelines themselves have been reported to be difficult to use (Carter, 1999). One major problem that is often reported is conflicting recommendations (e.g., Grammenos et al., 2000; Mayhew, 1992; Reed et al., 1999). In addition, guidelines tend to state what is to be done, without explaining why it should be done or how it can be done (Carter, 1999). Thimbleby (1990) reported that some designers were often skeptical of guidelines, because they seemed either trivial (too low level) or difficult to implement (too abstract).

A survey was conducted by Mosier and Smith (1986) on the use of their guidelines by designers. The results show that, in spite of the fact that most respondents considered the compilation useful, they reported problems in:

- 1) locating relevant guidelines among the 944,
- 2) choosing which guidelines to use, and
- 3) translating general guidelines into specific design rules.

de Souza et al. (1990) carried out an experiment to evaluate the effectiveness of a set of guidelines for menu interface design. Three designers were given a week to study the guidelines. They then spent one day using the guidelines to redesign a menu interface. The results showed that the designers had difficulties with 91% of the guidelines.

Therefore, there is a need to improve the usability of usability guidelines; Vanderdonckt (1999) referred to this type of studies as *meta-ergonomics*. Two general approaches that have been used to improve the usability of guidelines are as follows:

- Development of software tools that work with guidelines. For example, Vanderdonckt (1995) reported on the development of SIERRA (System Interactive for ERgonomic Realization of Applications), which is a program that assists retrieval of guidelines from a large database. Grammenos et al. (2000) reported on the development of Sherlock, which is a software program that facilitates automatic inspection of conformance to low-level guidelines.
- Formulating and presenting guidelines in more usable forms. For example, a group of researchers proposed the concept of pattern language, which originates from the discipline of architecture (Alexander et al., 1977). A pattern captures an exemplar of good design; it describes a common design problem and the core of the solution to that problem. A set of related patterns forms a pattern language. Patterns are similar to guidelines, but some researchers advocate the use of patterns because they adopt a more problem-oriented perspective (Henninger 2001; van Welie et al., 2000).

Alexander (1964, p.2) stated that a typical design problem has requirements that must be met, but there are interactions between the requirements, which make the design problem complex. Understanding and reducing the interactions helps to reduce the (perceived) complexity of the design problem. In the context of software user interface design, guidelines can be perceived as a set of generic design requirements (Tetzlaff and Schwartz, 1991). Alm (2003) pointed out that, because existing guidelines are highly coupled, it is not possible for designers to consciously keep track of the interconnections between the many variables or to calculate all the consequences and constraints that may emerge from putting all of the guidelines together. Hence, the usability of guidelines can be improved identifying and reducing their interactions.

In the HCI literature, there is a lack of techniques for analyzing interactions in usability guidelines. Hence, to complement the existing studies that aim to improve the usability of guidelines, an analytic tool has been developed in the latter part of this thesis. This tool has been developed to meet the following requirements:

- 1) Identifies and supports the representation of potential interactions in a set of guidelines.
- 2) Supports the elimination of interactions between guidelines by providing a mechanism for analyzing and restructuring guidelines.

1.4 Axiomatic Design

Suh (1990) presented a design methodology that is known as *Axiomatic Design* (AD). AD provides a framework for conceptualizing design problems (National Academy of Sciences, 2002).

Reported AD applications include product design, manufacturing system design, material engineering, software design, and organization design (El-Haik and Tate, 2002; Suh, 2001; Tate, 2000). According to Suh (personal communication, 2004), the

National Aeronautics and Space Administration (NASA) in United States is training some of its engineers in AD for the purpose of designing the next generation of space shuttles. In recent years, there are a few studies that apply AD in human factors engineering and human-computer interaction (HCI) (Helander and Jiao, 2002; Helander and Lin, 2000; Helander and Lin, 2002; Karwowski, 2003; Quill et al., 2001).

As a continuation of the previous research efforts, this study adapted and developed AD into both an analytic usability evaluation method and a tool for analyzing interactions in guidelines. The motivations for using AD are as follows:

- 1) AD provides a versatile framework for modeling different types of design problems. This versatility, which may allow one to adapt AD to human factors and HCI design problems, has been demonstrated by diversity of existing applications of AD.
- 2) Coupling is one of the most important topics in AD; the Independence Axiom of AD states that a good design is one that maintains the independence of the functional requirements. Previous research shows that coupling is a generic shaping factor of complexity, and it is a familiar concept in engineering (Endsley et al., 2002; Sadun, 2001). By providing a general and rational criterion for evaluation, a human factors method that is based on AD may be used by design engineers who are not experts in human factors engineering and cognitive psychology.
- 3) AD has been used for product design, machine design, manufacturing system design, software design, and organization design. A human factors method that is based on AD would speak the same language as designers who apply AD in these other fields. In addition, a human factors method that is based on AD may be able to benefit from theoretical advancements and development of software tools in AD. On the other hand, developing a human factors method that is based on AD would contribute to the overall development of AD.
- 4) AD offers a scheme for representing and analyzing functional coupling in design. It also provides a notation and criterion for characterizing the degree of

coupling in a design. Hence, AD may be useful for analyzing interactions in usability guidelines.

1.5 Research Purpose and Objectives

The purpose of this research is to explore how AD principles can be adapted and developed into 1) a method for analyzing user control tasks in human-machine interaction and 2) a tool for analyzing coupling in software usability guidelines.

The research objectives are as follow:

- 1) Develop a model of user tasks based on the concept of design domains.
- 2) Investigate how user control tasks can be represented using design equations.
- 3) Determine the implications of uncoupled design, decoupled design, and coupled design in the context of user control tasks.
- 4) Demonstrate the application of the proposed method.
- 5) Explore how design equations can be used for representing the structure of a set of guidelines.
- 6) Investigate how design matrix can be used as a mechanism for reengineering an existing set of highly coupled guidelines into a set of less coupled guidelines.

1.6 Organization of Thesis

The organization of this thesis is as follows. In Chapter 2, a literature review in the following subjects is presented:

- 1) Models of human-machine interaction,
- 2) Analytic usability evaluation methods,
- 3) Complexity methods in human-machine interaction,
- 4) Axiomatic Design, and

5) Effects of coupling on human problem solving.

In Chapter 3, a proposed framework and corresponding method for analyzing coupling in user control tasks are presented and discussed. An exploratory study of using a similar framework to analyze coupling in software usability guidelines is reported in Chapter 4. The conclusion of this research and a recommendation for future work are stated in Chapter 5. Figure 1.3 shows a schematic illustration of the organization of this thesis.

Figure 1.3. Structure of thesis.

1.7 Chapter Summary

The systems and products that we use, for work and for recreation, are becoming increasingly complex. Consequently, the design of usable products is an important subject in academia as well as in industry.

Previously, researchers stated that iterative design and user-based evaluation are two important principles for designing usable products. However, a combination of iterative design and user-based evaluation can be particularly costly. Hence, there is a need for early evaluation methods too.

Analytic methods can be particularly useful at early design stages, because they do not require working prototypes and end-users. However, existing analytic methods are inadequate. For the purpose of supplementing the existing pool of methods, this research proposed to develop an analytic tool based in the concepts of Axiomatic Design (AD), and the requirements for such a method have been stated. AD provides a framework for conceptualizing design problems, and it has been applied in many different fields.

For software user interface, usability guidelines are widely used to as a source of knowledge for design and evaluation. However, the guidelines have been reported to be difficult to use. For the purpose of improving the usability of guidelines, this research proposed to develop a tool for analyzing the interactions between guidelines, and the requirements for such a tool have been stated.

CHAPTER 2. LITERATURE REVIEW

Figure 2.1 shows an overview of the organization of this chapter. The first part of the literature review examines existing knowledge in human-machine interaction, and this includes 1) models that describe human actions and the complexity of human-machine systems and 2) existing methods for analyzing and evaluating human-machine systems.

The second part of this review discusses about the complexity of design. Interactions between design requirements result in complexity for designers. The results of an experiment demonstrate the detrimental effects of coupling on human-problem solving, and the Axiomatic Design (AD) is presented as a framework for dealing with this type of complexity.

Figure 2.1. Structure of literature review.

2.1 Models of Human-Machine Interaction

This section presents two models from the human factors literature that contributed to the development of the analytic evaluation method in this research. First, Norman's (1988) *model of human action* provides a framework for describing and studying human-machine interaction from a user's perspective. The "gulf of execution" in this model forms the scope of the proposed method. Following that, Weir's (1991) *pyramid of complexity* presents a system's perspective of human-machine interaction. This model illustrates how the complexity that a user faces is related to the complexity of the machine, and it provides further support for the need to include the system's structure in a usability analysis.

2.1.1 Norman's Model of Human Action

Norman (1988) presented a model that structures human actions into the following stages (see Figure 2.2):

- 1) perceiving the state of the system,
- 2) interpreting the perception,
- 3) evaluating the interpretations,
- 4) forming the goals,
- 5) forming the intention to act,
- 6) specifying the sequence of actions, and
- 7) executing the action sequence.

Figure 2.2. A reinterpretation of Norman's (1988) model of human action.

In this model, a user builds an understanding of the existing state of a system based on his or her perception. When there is a difference between the perceived system state and the desired system state, user attempts to eliminate the difference by executing a sequence of actions. Each of these stages is affected by different aspects of design. For example, the physical arrangement of controls affects the execution of actions, while the information design in displays affects the perception of feedback.

Based on this model, the term “gulf of execution” refers to the difference between the intentions of a user and the actions that are allowed by a system (Norman, 1988). When the gulf of execution is wide, it means that the user is not able to perform his or her intended actions directly.

Norman's (1988) model can be perceived as a functional model; it describes a human-machine interaction from the human's perspective. It does not describe the structure of the machine component in the human-machine system.

2.1.2 Weir's Pyramid of Complexity

Weir (1991) presented a structural model of human-machine system, which consists of the following layers:

- 1) A set of physical processes that enables the system to achieve its goals within a set of environmental conditions.
- 2) A set of control mechanisms, hardware and software, that enables the machine to govern the physical process.
- 3) A human-machine interface that enables the human operator to control the machine.

The complexity at each level percolates upward (see Figure 2.3), and the overall complexity is revealed to the user through the human-machine interface in terms of cognitive and physical requirements.

There are two implications in Weir's (1991) model that are particularly useful to this research. First, by modeling a system into several layers, one can be more specific in describing the locus of complexity. Identifying the exact location of complexity is particularly important for suggesting how and which part of the system should be improved. Second, to deal with the complexity at the human-machine interface, we may need to manage the complexity within the machine.

Figure 2.3. Usability depends on the design of the human-machine interface, which is constrained by the design of the control mechanism, which is constrained by the design of the physical process (adapted from Weir, 1991).

2.2 Analytic Usability Evaluation Methods

This section presents a review of several common analytic methods. The methods are as follows:

- 1) Hierarchical Task Analysis (HTA),
- 2) Task Analysis for Error Identification (TAFEI),
- 3) Goals, Operators, Methods, Selections (GOMS),
- 4) Task-Action Grammar (TAG), and
- 5) User Interface Design with Matrix Algebra (UIDMA).

It should be noted that HTA is more of a task description method, rather than an evaluation method. This is because HTA does not measure nor predict the usability of a

design. UIDMA is a relatively new method and its applications have not been well reported. Nevertheless, it is included in this review, because the use matrix algebra in this method makes it of interests to research in Axiomatic Design.

2.2.1 Hierarchical Task Analysis

Hierarchical Task Analysis (HTA) is a systematic method for describing the structure of a set of tasks. It is based on the notion that complex tasks can be expressed in terms of a hierarchy of goals (what a person is seeking to achieve) and sub-goals that are nested within higher order goals. Each goal and its corresponding actions (the means of achieving the goals) are represented as an operation (a verb), and the actions themselves may be redefined as sub-goals.

The general steps for carrying out a HTA are as follows (Annett, 2004):

- 1) Decide the purpose of the analysis. Typical purposes include improving the usability of an existing design, or analyzing safety, or developing a training program.
- 2) Define the operator's top goals and the criteria that will be used to decide whether the goals are met.
- 3) Collect data. Typical sources of information include interviews with experts and direct observation. Construct a decomposition table or diagram to represent the operator's tasks. Figure 2.4 shows an example of a HTA decomposition diagram, which presents a section of the goal hierarchy for an acid distillation plant operator's tasks. This decomposition process can go on indefinitely, so the analyst has to apply a *stopping rule* to decide when the tasks are basic enough, which depends on the purpose of the analysis (Dix et al., 1998).
- 4) Validate the way the tasks are decomposed by checking with the stakeholders of the analysis.
- 5) Identify the important parts of the tasks and generate hypotheses concerning their performance.

According to Annett (2004), the purpose of HTA is to investigate performance problems in a task. However, HTA itself does not provide any concrete criteria for evaluating a task nor a set of diagnostic tools. Hence, HTA is more useful as a tool for recording and communicating a task description (Stanton and Young, 1998). The process of performing a HTA may help an analyst to observe and think more carefully about a user's task activity, but conclusions from a decomposition diagram depend on the expertise of the analyst.

Figure 2.4. Example of a hierarchical task decomposition diagram (Annett, 2004).

2.2.2 Task Analysis for Error Identification

Task Analysis for Error Identification (TAFEI) is a method that can be used to identify possible user errors (Baber and Stanton, 1994). The main steps in carrying out a TAFEI are as follows:

- 1) Perform a HTA to model the human side of the interaction.
- 2) Construct state space diagrams (SSDs) to represent the behavior of the artifact.

- 3) Map the plans in the HTA to the SSD to form a TAFEI diagram.
- 4) Construct a transition matrix to display state transitions during device use.

Figure 2.5 shows an example of a TAFEI diagram. The three columns correspond to the three sub-goals in the HTA diagram for changing the date on a digital watch.

Figure 2.5. A TAFEI diagram for changing the date on a digital watch (Baber and Stanton, 2004).

A transition matrix is constructed based on the results of the HTA and the SSD to display the state transitions involved in the human-machine interaction (see Figure 2.6). If a transition is impossible, a “-” is entered into the cell (there is no such cases in the figure). If a transition is possible and progresses towards the goal state, an “L”, which stands for legal transition, is entered into the cell. If a transition is possible but does not progress towards the goal state, an “I”, which stands for illegal transition, is entered into the cell.

TAFEI is based on the notion that usability may be improved by making illegal transitions impossible thereby limiting the user to performing desirable actions (Stanton and Young, 1999). Illegal transitions are treated as potential user errors, and it is up to the analyst to conceive design solutions to remove them.

From states	To States				
		0	1	2	3
	0	I	L	I	I
	1	L	I	L	I
	2	I	L	L	I
	3	I	I	I	I

Figure 2.6. A transition matrix for changing the date on a digital watch (Baber et al., 2004).

Since TAFEI uses states and transitions to describe the behavior of a device, it is more suitable for analyzing interaction tasks that are discrete and step-by-step, rather than tasks that are continuous and concurrent. In addition, it is difficult to perform TAFEI when there are a large number of possible states in the system and when the states are difficult to identify.

2.2.3 Goals, Operators, Methods, and Selection

Goals, Operators, Methods, and Selection (GOMS) is a task analysis method, which uses a notation with syntax and semantic similar to traditional programming languages, for describing a task and user's knowledge (Card et al., 1980). A GOMS model consists of the following elements:

- 1) Goals - what a user wants to achieve.
- 2) Operators – the actions that a user interface allows a user to take.
- 3) Methods – sequences of sub-goals and operators that can accomplish a goal.
- 4) Selection – personal rules that a user follows in deciding which method to use in a particular instance.

There are several variants of GOMS model, such as the Keystroke-Level Model, the CPM-GOMS Model, and the NGOMSL (Card et al., 1980; Gray et al., 1993; Kieras, 1988). A typical GOMS model consists of a single high-level Goal that is decomposed into a sequence of unit tasks, and all of which can be further decomposed down to the level of Basic Operators (Dix et al., 1998). The general steps in constructing a GOMS model are as follows:

- 1) Choose the top-level user's Goals.
- 2) Write the top-level Method.
- 3) Recursively expand the Method hierarchy.
- 4) Document and check the analysis.

A GOMS model is a description of general methods that are used for accomplishing a set of tasks. If the model is correct, a user should be able to execute a series of *task instances* or specific tasks by executing the steps in the model using specific values (Kieras, 1997). However, there are always more than one way to decompose a task; hence, it relies on the analyst to make judgments about how users view the task in terms of their natural goals, how they decompose the task into sub-tasks, and what are the natural steps in the user's methods (Kieras, 2004).

The results of a GOMS analysis can be used to predict human learning and performance for a user interface. For example, the number and type of keystrokes can be used to predict the time required to perform a task, and the number of steps in the analysis can be used to predict learning time (Kieras, 2004).

It should be noted that GOMS does not identify the top-level user goals. Methods for identifying these goals include task analysis such as HTA, interviews with potential users, observation of users of similar or existing systems or intuition on the part of the analyst (Kieras, 2004).

A typical GOMS model describes user tasks and user knowledge in the form of production rules. One disadvantage is that these rules can be tedious and time-

consuming to formulate when the tasks are complex. Moreover, one task can be described by different set of rules, and it is difficult to verify which set of rules resembles the actual user behavior.

2.2.4 Task-Action Grammar

Task-Action Grammar (TAG) is a semi-formal method that analyzes the consistency of a user interface's input language (Schiele et al., 1990). Consistency is important, because it allows a user to make generalizations.

TAG expresses the how-to-do-it knowledge of a task as a *schema* that consists of rules (Schiele et al., 1990). Two or more tasks can share a single schema if the rules for performing the tasks are the same. The notion is that the number of rules required for a task is an indication of the difficulty of executing that task, and the number of schemas required by the input language of a user interface for performing the tasks that the system supports gives an indication of its consistency; the fewer the number of schemas, the easier it is for a user to generalize (Grant and Mayes, 1991).

TAG uses grammatical notations to model the relationships between task and the actions required to accomplish the tasks (Schiele et al., 1990). Tasks are defined by *values of features*. The actions to perform a task are described by a *feature grammar*. Because the grammatical specification of a user interface is rather laborious, to construct a complete TAG representation of a large system is beyond the resources that most people are prepared to commit (Schiele et al., 1990). Therefore, TAG is only suitable for comparing parts of alternative designs and when the input languages are relatively similar.

2.2.5 User Interface Design with Matrix Algebra

Thimbleby (2004) proposed a formal method for reasoning about the design of user interfaces with “push buttons” and “point and clicks”. This method is based on the notion that users and systems interact according to relevant laws of linear algebra; hence, a user’s task can be expressed as a matrix. This method is referred to as User Interface Design with Matrix Algebra (UIDMA) in this thesis.

Thimbleby (2004) claimed that using matrices as a notation has the following advantages:

- 1) Matrices are standard mathematical objects, with a history going back to the nineteenth century. Hence, they constitute an established and well-defined notation.
- 2) Matrices are easy to work with. Hence, designers can work out user interface issues using matrices easily, and practical tools that work with matrices can be built.
- 3) Matrix algebra has structure and properties. Hence, designers and human-computer interaction (HCI) specialists can use it to reason about what is possible and not possible in general ways.

UIDMA uses a unit vector e_s of length N , where N is the total number of possible states, to represent the state of a finite state machine (FSM). e_s is a vector of “0”s with a “1” at the position corresponding to the state number s . When a transition occurs, the FSM goes into a new state, and the transition is represented by a matrix B , where B is an $N \times N$ matrix of “0”s and “1”s. Hence, finding the new state amounts to a matrix multiplication, which is e_s times B .

As pointed out by Thimbleby (2004), UIDMA does not predict individual preferences, motivation, pleasure, learning or human errors, which are the concerns of psychological theories. On the other hand, UIDMA has the potential to uncover other types of

usability problems, such as task procedural problems, which may be missed by methods that are driven from the psychological realism.

Although Thimbleby (2004) claimed that UIDMA is an easy method to use, in the author's opinion, further studies are required to demonstrate its usefulness for designers who are not familiar with either linear algebra or the formal methods' concept of theorem proving.

2.2.6 Discussion

Two general conclusions can be made from the above review. First, the use of hierarchical structures to model a set of user tasks is a common and well-accepted technique. All of the above methods involve the use of some form hierarchical structures. Even in UIDMA, there is a form of abstraction in terms of matrices and sub-matrices. Therefore, the usefulness of representing a set of tasks in terms of a hierarchical structure should be taken into consideration in the development of new methods.

Second, there are several general deficiencies in this set of analytic methods:

- 1) As stated earlier above, HTA and UIDMA do not provide a criterion for design evaluation. Any conclusions regarding the usability of a design is based on the expertise of the analyst, and different analysts may have different opinions of a design.
- 2) GOMS and TAG provide evaluation criteria, such as the number of steps or schemas that is required to complete a set of tasks. However, these methods are used mainly for evaluating software-based user interface.
- 3) Similarly, applications of TAFEI and UIDMA are restricted to systems that can be represented by a limited number of discrete states, such as "push-button machines".

The above review of existing analytic methods is not exhaustive. However, the methods that were covered originate from many different disciplinary roots, such as psychology, work-studies, and computing, and they share many common characteristics with other existing analytic methods. For instance, TAG's notion of using a mathematical-based grammar to specify human-machine interaction is also found in many formal methods in HCI (Harrison and Thimbleby, 1990). Therefore, the methods that were covered can be deemed as representative of existing analytic methods.

In conclusion, for a new analytic method to be useful, the following requirements should be considered during its development:

- 1) Provide a general criterion for design evaluation. A method that does not provide a criterion for evaluation will be dependent on the subjective judgments of individual design engineers.
- 2) Applicable to general hardware-based systems that are not of a FSM nature, such as those that allow concurrent and continuous control.
- 3) If necessary, hierarchical structures should be used to describe complex user tasks, because their usefulness has been widely accepted.

It should be noted that the purpose of developing new methods is to complement, rather than to replace, existing methods. Due to the diversity of human-machine systems and the wide scope of usability, it may not be likely or meaningful to have a method that is applicable to all domains.

2.3 Complexity Methods in Human-Machine Interaction

This section presents another review of several methods for human-machine interaction. These methods belong to a special class, because they share the common claim of measuring the complexity of a human-machine system. Based on this review, the differences between these methods and the method that has been developed in this thesis will be discussed in Section 3.6.

2.3.1 Cognitive, Behavior, System, and Task Complexity

Rauterberg (1992) proposed a framework and method, which makes use of McCabe's (1976) complexity measure for analyzing human-computer interaction (HCI). This framework distinguishes between cognitive complexity (CC), behavior complexity (BC), system complexity (SC), and task complexity (TC) (see Table 2.1).

Table 2.1 Definitions of complexity according to Rauterberg (1996).

Type of Complexity	Definition
Cognitive complexity (CC)	The complexity of a user's mental model of a dialog system, which depends on the number of known dialog contexts and known dialog operations.
Behavior complexity (BC)	The complexity of a user's observable behavior, which can be estimated by analyzing his recorded concrete task solving process.
System complexity (SC)	The complexity of an interactive system.
Task complexity (TC)	The complexity of a task structure, which depends on the minimum amount of knowledge that is necessary for solving the task. The complexity of a given task is constant.

According to Rauterberg (1996), when a user learns to solve a specific task with a given system, his or her BC decreases until it is equal to the TC, and his or her CC increases until it is equal to the SC. Rauterberg argued that the complexity of an observable task solving space is constrained by the structure of the system that is used. Consequently, Rauterberg assumed that

$$BC - TC = SC - CC \quad (2.1)$$

Hence,

$$CC = SC + TC - BC \quad (2.2)$$

Therefore, the complexity of a user's cognitive structure for solving a problem or task can be derived if the complexity of his behavior (BC), the complexity of the system (SC), and the complexity of the task (TC) are known. BC depends on the number of states and transitions that a user goes through in his task solving process. SC depends on the total number of possible states and transitions in the system. TC is estimated as the minimum of all observed BCs among test subjects.

Rauterberg's (1996) method is similar to TAFEI in the sense that it describes a system in terms of states and transitions. Hence, it is only suitable for analyzing interaction tasks that are discrete and step-by-step, rather than tasks that are continuous and concurrent. Further studies are required to validate the assumptions that the mathematical formulae in this method are based on. For example, Equation 2.2 would be incorrect if there are other variables that have not been taken into account. Lastly, it should be noted that, unlike the other methods that were reviewed, this method is dependent on empirical data (for deriving BC and TC).

2.3.2 Cognitive Complexity Theory

Kieras and Polson (1985; 1999) presented a *Cognitive Complexity Theory* (CCT), which is based on GOMS, for analyzing the complexity of a system from a user's perspective. CCT uses two parallel descriptions: one to describe the user's knowledge of how to use a system, which is known as the *user's job-task representation*; the other to describe the behavior of the system itself, which is known as the *device representation*.

Kieras et al. (1985) suggested that a good design is one that has a one-to-one mapping between the use task representation and the system representation. An example is illustrated in Figure 2.7, which shows the task-device mapping for a text-editing software. The left hand side of the figure, which is abstracted from the software's generalized transition network, shows a hierarchical arrangement of the major functional components of the software. The user's goal structure does not correspond to

the system's structure; hence, the correct order of actions may appear to be confusing to the user and results to user errors. Kieras et al. noted that this problem could be solved by changing either the system's design or the user's goal structure. The user's goal structure can be redefined through training and documentation.

Figure 2.7. Example of a poor task-device mapping (Kieras et al., 1999).

2.3.3 Task-To-Action Model

Kang and Seong (1998) proposed a *Task-To-Action Model* (TTA), which describes the work procedure for a human operator who deals with information on a computer display. This model is based on the Hick-Hyman law and the maximum channel capacity concept of information theory (Wickens and Hollands, 2000).

Kang et al.'s (1998) method for measuring complexity involves the following measures:

- 1) operation complexity,
- 2) transition complexity, and
- 3) screen complexity.

Each of these measures is briefly discussed below.

Operation complexity is related to the underlying structure of the screens in a system (see Figure 2.8), and it depends on the operational profile, which is calculated using the following formula:

$$p_i^{OP} = \frac{OP_i}{\sum_{j=1}^{n_{OP}} OP_j} \quad (2.3)$$

where OP_i is the number of anticipated usage instances for the i^{th} path by an operator and n_{OP} is the total number of possible paths in the system.

Operational profile is at a maximum when each path is equally probable. The ratio of a system's operational profile to the maximum operational profile indicates the relative operation complexity. A system with a small number of frequently accessing paths has low operation complexity. Contrary, a system with a large number of equally probable paths has high operation complexity.

Transition complexity is related to the number of possible next screens, and it depends on the mental load profile, which is calculated using the following formula:

$$p_i^{TR} = \frac{TR_i}{\sum_{j=7}^{n_{TR}+6} TR_j} \quad (2.4)$$

where TR_i is the number of screens in the i^{th} category and n_{TR} is the number of categories.

Figure 2.8. Structure of an exemplary information system (Kang et al., 1998).

The screens in each category have the same number of buttons or menus. Mental load profile is at a maximum when every TR_i equals to one. The ratio of a system's mental load profile to the maximum mental load profile indicates the system's relative transition complexity. Transition complexity increases as the number of possible next screen increases.

Screen complexity is related to the overall consistency of the screens in a system, and it depends on the relative frequency of screen design types, which is calculated using the following formula:

$$p_i^{TSC} = \frac{SC_i}{\sum_{j=1}^{n_{SC}} SC_j} \quad (2.5)$$

where SC_i is the number of screens in the i^{th} design type and n_{sc} is the total number of screen design types.

Relative frequency of screen design types is at a maximum when the frequency of every screen design type is equals to one. The ratio of a system's relative frequency of screen design types to the maximum relative frequency of screen design types indicates the system's screen complexity. When a system is designed with high consistency across screens, it has relatively low screen complexity. Contrary, when there are a large number of screen design types, the system has high screen complexity.

This method measures the complexity of display navigation while considering the information structure of the displays and the consistency across the display designs. However, it is not clear how an analyst can predict the number of anticipated usage by an operator for each path (OP_i), especially during the early design phase. Moreover, many systems allow the operator to switch to another task without the need to return to the main screen, which is unlike the case shown in Figure 2.8.

2.3.4 Framework for Quantification of Complexity and Compatibility

Karwowski (1997) proposed a framework for defining a system's complexity and compatibility. In this framework, the complexity of an ergonomic system (ψ) is defined as a function of the following variables:

- The number of components in the system (N_c).
- The number of interactions within the system (N_I).
- The strength of these interactions (K_I).

The compatibility of an ergonomic system (θ) is defined as a function of the following variables:

- The inherent incompatibility between the components of the system (I_p).
- The strength of the interactions between the components of the system (K_{pq}).

- The significance of the interactions between the components of the system (α_p).

According to Karwowski (1991), a compatible system is one that requires a minimal level of human interaction to achieve the required level of fit. Using office chair design as an example, Karwowski (2003) explained that a chair is compatible with the user if it fits the user well and is comfortable; a typical solution is to increase fit is to increase the adjustability of the chair. However, increasing the degrees of freedom of the chair also increases the complexity of the chair, and the user may not be able to properly adjust the variety of chair features to ensure his comfort.

Using a scale of '0' to '1' to measure both ψ and θ , an ergonomic system can be represented by its compatibility and complexity coordinates in a chart (see Figure 2.9). An ideal design is one that is high in compatibility but low in complexity.

Figure 2.9. The complexity-compatibility quantification framework (Karwowski, 2003).

Karwowski (2003) presented several mathematical formulae for calculating the *total adjacency*, the *total reachability*, and *total connectivity* of a system. These formulae are not reproduced in this thesis, because they are not directly relevant to the discussion.

This method considers the number of components as one of the main contributing factors of complexity. However, not every component in a system is related to user goals, and many studies suggest that users often treat the internal workings of a system as a black box (Preece et al., 1994). Hence, there is a need to differentiate between complexity in the structural model (how a system functions) and complexity in the functional model (how a user operates a system).

In its current form, this framework is rather abstract. Further research is required to develop this framework into a readily applicable method for engineers and designers.

2.4 Axiomatic Design

The content of this section is as follows. First, a discussion on the use of Axiomatic Design (AD) as a framework for managing complexity in design will be presented. Following that is a summary of previous AD applications in human factors engineering and human-computer interaction (HCI) will be presented.

2.4.1 Axiomatic Design as a Framework for Managing Complexity of Design

Suh (1990) presented a design methodology that is known as *Axiomatic Design* (AD). The fundamental concepts of AD can be summarized as follows:

- 1) The concept of design domain.
- 2) The concept of design hierarchy.
- 3) The concept of zigzagging.

4) The Independence Axiom.

5) The Information Axiom.

Using these concepts, AD offers a method for representing design problems in a formal manner and two criteria for design evaluation (National Academy of Sciences, 2002). Appendix A1 discusses AD in more details.

AD is given its name because Suh (1990, p.47) claimed that it is based on a set of axioms, and “axioms are fundamental truths that are always observed to be valid and for which there are no counterexamples or exceptions”. However, this research conceives AD as based on a set of stratagems for dealing with complexity; more specifically, AD offers techniques for reducing complexity of design.

Definitions of complexity from different fields of research show that that the number of parts in a system and the extensiveness of interaction between the parts are two generic and principal sources of complexity (Steward, 1981; Weng et al., 1999; Woods, 1988; Yates, 1978). Complexity that is due to a large number of parts can be referred to as *component complexity*, and the complexity that is due to a large number of relations can be referred to as *relational complexity* (Miller, 2000).

Component and relational complexities are related to a limited information processing capacity. People can attend to and mentally manipulate only four independent pieces of information at the same time (Halford et al., 1997). Therefore, when the number of parts and relations in a system is excessive, people experience great difficulty in predicting the effects of an action or tracing the implications of a disturbance in a system (Döner, 1996).

A powerful tool for managing complexity is abstraction, because abstraction reduces the number of parts and relations that a person needs to think of simultaneously (Ossher, 1987). Several concepts in AD can be perceived as forms of abstraction. The concept of design domains reduces complexity by guiding a designer to focus on design attributes that belong to two adjacent design domains at any one time; each design domain

represents one aspect of a design problem. The concept of design hierarchy reduces complexity by limiting the level of detail at each design level.

The Independence Axiom can be perceived as principle for reducing component and relational complexities in design. When a design is highly coupled, it is not possible for the designer to focus only on just a few critical parts in the design. This is because the property of a coupled design as a whole can be very different from the properties of its individual parts. Avoiding coupling through a proper selection of design parameters at each abstraction level helps to reduce the complexity of the design process and design artifact.

The empirical effects of coupling on human problem solving are demonstrated by an experiment that was conducted by Hirschi and Frey (2002). This experiment and its results will be discussed in Section 2.5.

Before Suh's (1990) introduction to AD, many researchers already pointed out the need to manage interactions in design. Alexander (1964, p.2) stated that a typical design problem has requirements that must be met, but the interactions between the requirements make the requirements difficult to meet. He suggested that designers should use a constructive diagrams or an interaction matrix, such as the one shown in Figures 2.10 and 2.11, to represent, understand, and manage the interactions between the design requirements or system components.

According to Jones (1992), the interaction matrix is one of the most useful design aids that have emerged in the search for systematic design methods. In addition, Steward (1981) presented a design-structure-matrix method (DSM) that is based on Alexander's ideas. However, the constructive diagram, the interaction matrix, and DSM have two major limitations. First, a problem arises when the requirements do not belong to the same hierarchical level. In this case, the design problem will appear to be more coupled than it actually is.

Figure 2.10. A constructive diagram can be used to represent the interactions between the requirements of a mechanical design problem (Alexander, 1964).

	1	2	3	4	5	6	7	8	9	10
1 lobby	-	2	0	2	0	0	0	0	0	1
2 waiting space	-	-	2	0	2	0	0	2	0	0
3 treatment room	-	-	-	2	2	2	0	0	2	0
4 side ward	-	-	-	-	0	1	0	0	0	0
5 consulting room	-	-	-	-	-	1	0	1	0	0
6 office	-	-	-	-	-	-	1	0	1	0
7 nurses' w.c.	-	-	-	-	-	-	-	0	0	0
8 patients' w.c.	-	-	-	-	-	-	-	-	0	0
9 medical store	-	-	-	-	-	-	-	-	-	0
10 cleaners' store	-	-	-	-	-	-	-	-	-	-

Figure 2.11. Example of an interaction matrix that shows interactions, which vary in strength, between components of a hospital design (Jones, 1980).

Second, there are some ambiguities regarding the definition of interactions. Alexander (1963) stated, as cited in Jones (1992), two design requirements are said to interact if

“whatever you do about meeting one makes it either harder or easier to meet the other, and if it is in the nature of the two requirements that they should be so connected, and not accidental”. Thus, Alexander’s notion of interaction in design problems is independent of design solutions.

However, there is a need to distinguish between goal conflicts and plan conflicts; plan conflicts arises not through the nature of the goals themselves but through the plans selected to achieve these goals (Wilensky, 1983). Since design problems are widely recognized as being ill defined or ill-structured problems, there is no definitive formulation of a design problem (Ahmed et al., 2003; Caldenfors, 1998; Goel and Pirolli, 1992). Consequently, conflicts in design requirements emerge in the process of problem solving because the formulations of the design problem depend on the method selected for solving them (Cross, 1994). A designer may well be able to find particular sets of sub-solutions that are able to minimize conflicts between design requirements (Jones, 1992). Therefore, the existence of interaction between two requirements is not deterministic.

AD overcomes these two limitations using the concept of design hierarchy and the concept of design domains. By distinguishing between design requirements and design solutions, an interaction between two requirements becomes something that can be determined and eliminated.

2.4.2 Axiomatic Design Applications in Human-Machine Interaction

AD has been reported to be applicable in various fields, such as product design, manufacturing system design, material engineering, software design, and organization design (El-Haik and Tate, 2002; Suh, 2001; Tate, 2000). In recent years, some researchers applied AD in human factors engineering and human-computer interaction (HCI). A summary of these studies is presented below.

Helander and Lin (2000) used AD to design a microscope workstation. Suitable *design parameters* (DPs) were selected to maintain the independence of the *functional requirements* (FRs), which were related to the adjustability of the workstation. The results showed that the height of adjustable chairs introduces coupling and it is better to use an adjustable footrest.

In addition, Helander et al. (2000) showed that a modified definition of information content is more suitable for comparing adjustability features in workstation design. This modification involved a re-definition of system range and design range. In addition, Helander et al. reported that the two design axioms fit well with design methodology in ergonomics, because they place a strong emphasis on user requirements.

Quill et al. (2001) applied AD to information visualization. The primary difficulty of their design problem was the presentation of apparently conflicting information to the user. To overcome this difficulty, Quill et al. matched the FRs to specific DPs, and translated the hierarchical breakdown of FRs into hierarchical representations in the display. The improved design received positive user feedback. In addition, Quill et al. reported that the decoupled characteristic of the improved design allowed the developers to easily add in user personalization features to the software.

Helander and Lin (2002) demonstrated how AD could be used for biomechanics design of hand tools and for anthropometric design of workplaces. The study showed that the use of the Independence Axiom could avoid time-consuming iterative improvements of design solutions. In one of the case studies, it was demonstrated that the number of design iterations could be reduced if the environment or machine was designed first, followed by the operator's workstation and the task. A design sequence that begins from the operator's task, which is commonly accepted in the human factors community, is actually sub-optimum for this case.

Helander and Jiao (2002) extracted FRs and DPs from Nielsen's (1994b) 10 usability heuristics and constructed a design matrix to represent the dependencies between the

FRs and the DPs. The highly coupled design matrix suggested that the heuristics are not independent from each other. Helander et al. concluded that the usability heuristics lead to coupled design and evaluation processes, and they suggested a reformulation of the heuristics by manipulating the design matrix to reduce the degree of coupling.

The above studies suggest that AD concepts may be useful in several aspects of human factors engineering and human-computer interaction. Two of these studies are of particular interest to this research:

First, Helander and Lin's (2000) study suggests a possibility of applying AD's concepts in ergonomics evaluation. However, further research is required to understand whether AD's concepts can be integrated with existing knowledge in human factors into a systematic and analytic usability evaluation method. More specifically, the following issues need to be explored:

- 1) Can AD concepts be used to construct a functional model and a structural model of human-machine interaction? If so, how can one represent the relationships between the two models?
- 2) How to adapt AD's concepts into a method for predicting the gulf of execution for a human-machine system?
- 3) How to adapt AD concepts into a method that can be used to identify the percolation of complexity from within a machine design to its human-machine interface and thereby suggest design changes for reducing complexity for the user?

Second, Helander and Jiao's (2002) study shows that AD's framework can be used to analyze interactions between usability guidelines. However, further research is required to understand how the design matrix can be used as a mechanism for reengineering a set of coupled guidelines and how the results of such a reengineering process would be like.

2.5 Effects of Coupling on Human Problem Solving

When Axiomatic Design (AD) was first introduced by Suh (1990), he did not provide a detailed explanation to why coupling in design should be avoided. However, researchers in other fields have pointed out that coupling is a generic shaping factor of complexity (e.g., Endsley et al., 2002). In a more recent publication, Suh (2005) explained that coupling results in *time-independent real complexity*.

The empirical results from an experiment further show that the size of a problem and the degree of coupling within the problem have a detrimental effect on people's problem solving performance (Hirschi and Frey, 2002). A description of the experiment is as follows.

Twelve subjects were asked to adjust the input variables, which were controlled by slider bars on a simple graphical user interface (GUI), until the output variables, which were indicated by gauges in the GUI, fell with specified ranges. Figure 2.12 shows the GUI that was used for a three output variables and three input variables (3×3) parameter design task. Other GUIs were created for 2×2 , 4×4 , and 5×5 problems by adding or removing slider bars and output gauges.

The output variable display gauges were not designed to update themselves continuously as the inputs were varied. Instead, the positions of the output variable indicators were updated only after the subjects pressed the "refresh plot" button, which was at the lower right-hand side of the GUI (see Figure 2.12).

The number of input-output-variable pairs varied from two to five. The type of mapping between the input variables and the output variables was altered randomly across each test between one-to-one relationships to many-to-many relationships; the subjects were not informed about this. In addition, the subjects were not allowed to use pens, papers, or computational aids.

Figure 2.12. GUI for a 3×3 parameter design problem (Hirschi et al., 2002).

The results of the experiment can be summarized as follows:

- The subjects completed the uncoupled tasks in a sequential process.
- The subjects completed the coupled 2×2 problems in a fairly direct manner with few missteps after the underlying dependencies among inputs and outputs were determined.
- There was a modest but statistically significant increase in the time required to complete the coupled 2×2 tasks as compared with uncoupled 2×2 tasks.
- The solution procedures employed by the subjects for the coupled 3×3 , 4×4 , and 5×5 problems involved more iteration, a greater fraction of non-converging moves, and much more time.

In the pilot test, subjects were also presented with coupled 6×6 problems, but a substantial fraction of the subjects became too frustrated and would not complete the tasks (Frey et al., 2002).

Figure 2.13 shows that the time taken by the subjects to complete an uncoupled design task increased linearly with the problem size. On the other hand, the time taken to complete coupled design tasks increased geometrically with the problem size. Therefore, the results of this experiment is in agreement with the notion that coupling between problem variables has a significant and detrimental effect on human problem solving performance.

Figure 2.13. Effects of coupling on parameter design task completion time (Hirschi et al., 2002).

The results of this experiment provide empirical evidence for the detrimental effects of coupling in human problem solving. It should be noted that the results of this experiment could be different if the change in readings on the display gauges was instantaneous; the subjects would not need to click on the “refresh plot” button after every adjustment, and the underlying dependencies between input and output variables might be more obvious. However, the general pattern of geometrically increasing problem solving time for coupled problems is unlikely to change, because it is an effect that is due to the general limitations of people’s cognitive capacity.

2.6 Chapter Summary

First, two models in human-machine interaction were reviewed. Norman's (1988) model distinguishes human actions into execution and evaluation stages. User's difficulties in the execution stages constitute the gulf of execution. This model can be perceived as a functional model of human-machine interaction. Weir's (1991) model describes a human-machine system in layers, and the complexity at each layer percolates upwards. For a poorly designed system, the resultant complexity emerges to the user at the user interface. This model can be perceived as a structural model of human-machine interaction.

Following that, several existing analytic methods and complexity methods were reviewed. These methods originate from diverse fields, and they are deemed as an acceptable representation of existing analytic and complexity evaluation methods. These reviews facilitate a comparison, in a later part of this thesis, with the method that has been developed in this research.

Suh's (1990) Axiomatic Design (AD) was presented as a method for dealing with the complexity of design. The concept of design domains and the concept of design hierarchy were conceived as different forms of abstraction. The Independence Axiom was conceived as a principle for minimizing complexity during a decomposition process.

Recent applications of AD in human factors engineering and human-computer interaction were discussed; Helander and Lin's (2000) and Helander and Jiao's (2002) studies were found to provide two specific directions for further research.

Finally, an experiment by Hirschi and Frey (2002) was reviewed. Subjects were asked to perform parameter design tasks, and the design tasks varied in the number of parameters and the degree of coupling between the parameters. The results show that coupling has significant and detrimental effects on human problem-solving

performance, and they support AD's argument that coupling in design should be avoided.

CHAPTER 3. TASK COMPLEXITY ANALYSIS

This chapter reports on the development of a method for analyzing human-machine interaction. This method is based on Norman's (1988) model of human action, Weir's (1991) pyramid of complexity, and Suh's (1990) axiomatic design (AD). First, an overview of the development process and a skeleton framework for describing a set of user tasks will be presented. Following that is a description of an extended framework that includes a system's structural model. At the end of this chapter, several case studies are used to exemplify the proposed method, followed by a discussion on the implications of this research.

3.1 Overview

In developing the methodology that is proposed in this thesis, the author iterated between reviews of the literature, conceptualizations, and verifications with case studies. The underlying framework of the proposed method is largely based on Suh's (1990) Axiomatic Design (AD) and Helander and Lin's (2000) study.

The literature review presented in Chapter 2 provided ideas on how to integrate the AD framework with some of the existing concepts in human-machine interaction. For example, Preece et al's (1994) concept of functional and structural models and Weir's (1991) pyramid of complexity indicated that, to be effective in identifying the locus of complexity in a human-machine system, the proposed framework should include descriptions of both the user tasks and the system's structure.

The final methodology that is presented in this thesis has been refined by several case studies. These studies helped to 1) verify whether the framework is sufficient for its purpose and 2) facilitate a better understanding of the generalizability of the method.

3.2 Duel-Domains Framework

The literature review shows that user tasks are usually described in terms of goals and actions, and to narrow the gulf of execution, a design should enable a user to map the goals to the actions in a simple and direct manner (Annett, 2004; Kieras, 1988; Norman, 1988). A skeleton model for representing user task design was constructed by synthesizing these concepts with Suh's (1990) concept of design domains (see Figure 3.1).

This model consists of two design domains: a goal domain and an action domain. In the Axiomatic Design (AD) framework, each design domain represents a certain aspect of design. Similarly, each of the two domains in Figure 3.1 represents one aspect of a user task design. The goal domain contains the goals that a user wants to achieve. These *user goals* (UGs), which are similar to what Pribeanu and Vanderdonckt (2002) referred to as functional goals, are expressed in terms of user's desired system states. The action domain contains the *user actions* (UAs) that a designer plans for the user to perform to achieve the UGs. Together, the UGs and UAs constitute a set of *user tasks* (UTs).

The goal domain is linked to the action domain through a means-ends relationship, which has two perspectives. To an interaction designer, the UAs are the results of his or her design decisions and the solutions to the design problem. To a user, the UAs are the means that he or she uses to achieve the goals. The directness of the mapping affects usability and determines the gulf of execution (Kieras et al., 1985; Norman, 1988; van Welie, 1998).

Figure 3.1. User task design can be represented by the mapping from the goal domain to the action domain.

Based on the concepts of component and relational complexities, a set of UTs is less complex when there is a one-to-one mapping, and it is more complex when there is a many-to-many mapping. Hence identifying the interactions between UGs and UAs can help to predict usability problems. Furthermore, reducing the degree of coupling between UGs and UAs helps to reduce the complexity of a set of UT, minimize the gulf of execution, and simplify a user's mental model of the product.

Using AD's notation, a design equation can be used to represent the mapping from the goal domain to the action domain:

$$\{UG\} = [U]\{UA\} \quad (3.1)$$

where $\{UG\}$ is a *vector of user goals*, $\{UA\}$ is a *vector of user actions*, and $[U]$ is referred to as a *user-task design matrix*.

It should be noted that the mapping from UGs to UAs can also be represented using a directed graph. However, as the number of vector components or nodes increases, a design equation becomes easier to work with than a directed graph. A design equation is also easier to manipulate in computers for analysis. Furthermore, the physical form of a

design matrix provides a useful visual summation and a qualitative metric for characterizing the degree of coupling.

As with AD, the shape of the user-task design matrix can be used to identify the degree of coupling:

- 1) coupled,
- 2) uncoupled, and
- 3) semi-coupled or decoupled.

A *coupled user-task design* is one in which there are dependencies on both sides of the diagonal of $[U]$; such a matrix is known as a *full matrix*, and there is a many-to-many mapping (see Equation 3.2). An *uncoupled user-task design* is one in which all the non-diagonal elements of $[U]$ are zeros; there is a one-to-one mapping (see Equation 3.3). A *semi-coupled user-task design* is one in which $[U]$ is a triangular matrix; there is a one-to-many mapping (see Equation 3.4).

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \end{Bmatrix} = \begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \end{Bmatrix} \quad (3.2)$$

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \end{Bmatrix} = \begin{bmatrix} X & & \\ & X & \\ & & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \end{Bmatrix} \quad (3.3)$$

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \end{Bmatrix} = \begin{bmatrix} X & & \\ X & X & \\ X & X & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \end{Bmatrix} \quad (3.4)$$

The design of a water faucet can be used to exemplify the difference between a coupled and an uncoupled user-task design. This example was first suggested by Sohlenius (personal communication, 1996). It also demonstrates the detrimental effects of coupling on product usability. Consider a water faucet that has two knobs: one for

controlling hot water, another for controlling cold water (see Figure 3.2). The UGs and UAs for such a design can be stated as follows:

UG_1 = Desired water flow rate.

UG_2 = Desired water temperature.

UA_1 = Turn hot water knob.

UA_2 = Turn cold water knob.

The design equation can be formulated as

$$\begin{Bmatrix} UG_1 \\ UG_2 \end{Bmatrix} = \begin{bmatrix} X & X \\ X & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \end{Bmatrix} \quad (3.5)$$

Figure 3.2. A water faucet that has two knobs for controlling, individually, the flow of hot and cold water.

This is a coupled user-task design, because $[U]$ is a full matrix. Turning either of the knobs affects both UGs. For instance, when the water temperature is too hot, a user can lower the temperature by turning either of the knobs (to reduce the hot-water flow rate or increase the cold-water flow rate). However, both actions will result in an unintentional change in the combined flow rate. Thus, the new combined flow rate becomes too high or too low. The user will need to perform several adjustments of both controls to achieve the desired water temperature and flow rate.

Consider another water faucet design that has a single handle for controlling the flow rate and the water temperature (see Figure 3.3). While the UGs for this design can be assumed to be the same as the two-knobs design, the UAs are as follows:

UA_1 = Pivot handle up/down.

UA_2 = Pivot handle left/right.

Figure 3.3. A water faucet that has a single lever for controlling the flow and the temperature of the water. Lifting the lever increases flow, and turning the lever towards the left raises temperature.

The design equation can be formulated as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \end{Bmatrix} = \begin{bmatrix} X & \\ & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \end{Bmatrix} \quad (3.6)$$

This is an uncoupled user-task design, because $[U]$ is a diagonal matrix. Such a design enables the user to achieve the goals in a straightforward manner. The functional independence of the design relieves the user from the need to perform control functions mentally.

The above example illustrates that a coupled user-task design can be a modular design, while a physically integrated design can be an uncoupled user-task design. Therefore, it is important to understand that coupling is analyzed with respect to the user goals, rather

than the design of the physical controls. This is in agreement with Suh's (1990) distinction between functional and physical coupling.

A semi-coupled user-task design is more complex than an uncoupled design, but it is less complex than a coupled design. An example of a semi-coupled design, first suggested by Helander and Lin (2000), is an adjustable microscope workstation that consists of the following main components:

- 1) a height adjustable operator's chair,
- 2) a height adjustable worktable, and
- 3) a microscope with an adjustable eyepiece, which is placed on the worktable.

The primary UGs for such a design, from a physical ergonomics perspective, can be stated as follows:

UG_1 = Feet are supported when the thighs are parallel to the floor.

UG_2 = Elbows are supported when the shoulders are relaxed.

UG_3 = Microscope eyepiece in front of eyes when the back is relaxed.

To achieve these UGs, the user has to adjust the workstation and the corresponding UAs can be stated as follows:

UA_1 = Adjust chair height.

UA_2 = Adjust table height.

UA_3 = Adjust eyepiece.

Therefore, the design equation is as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \end{Bmatrix} = \begin{bmatrix} X & & \\ X & X & \\ X & X & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \end{Bmatrix} \quad (3.7)$$

This is a semi-coupled design, because $[U]$ is a triangular matrix. A semi-coupled design is more complex than an uncoupled design, because there is only one correct

sequence for performing the tasks. In Equation 3.7, the correct action sequence is UA_1 , followed by UA_2 , then UA_3 . That is, the user should adjust the chair height first. Subsequently, although the support at the elbows is affected by both the chair height and the table height, only the table height needs to be adjusted, because the chair height has already been set. Lastly, only the microscope eyepiece needs to be adjusted, because the chair height and the table height have already been decided. Therefore, an uncoupled design, which has a diagonal design matrix, is simpler than a semi-coupled design -- there are no such constraints on the operation sequence.

A semi-coupled user-task design becomes coupled if the proper *user-task sequence* is disregarded. For example, $[U]$ in Equation 3.7 becomes a full matrix when the action sequence is changed to UA_3 first, followed by UA_1 , then UA_2 :

$$\begin{Bmatrix} UG_3 \\ UG_1 \\ UG_2 \end{Bmatrix} = \begin{bmatrix} X & X & X \\ & X & \\ & X & X \end{bmatrix} \begin{Bmatrix} UA_3 \\ UA_1 \\ UA_2 \end{Bmatrix} \quad (3.8)$$

In this case, the user will not be able to satisfy all of the goals during the first attempt, and iterative adjustments become necessary. This is because adjusting the chair height (UA_1) after the microscope eyepiece has been adjusted (UA_3) will ruin the previous goal of placing the eyepiece at the eye level (UG_3). Similarly, adjusting the table height (UA_2) after the eyepiece has been adjusted will move the eyepiece away from the eye level.

Therefore, a user-task design is characterized with respect to the UGs and the sequence of the UTs. When the correct user-task sequence is not followed, a semi-coupled design becomes coupled, and this is represented by the change in the form of the user-task design matrix.

In a user-task design equation, $[U]$ shows whether there is coupling between the UTs, and coupling potentially leads to usability problems. The scope of this method lies

between the intention and the execution stages in Norman's (1988) model of human action. This type of usability problems, which is related to task procedures, is referred to as structural usability problems. It should be noted that coupling potentially leads to usability problems, but an uncoupled design may still be difficult in the following circumstances

- User has a different set of UGs from those that were assumed by the designer.
- User is not aware of the available UAs. For instance, the location of the controls is not obvious.
- User is not able to perform the UAs. For instance, the controls are physically difficult to manipulate.
- System's feedback is poor.

3.2.1 Matrix Reordering Algorithms

The above workstation example shows that the form of $[U]$ is dependent on the components in the user goal vector $\{UG\}$, the components in the user action vector $\{UA\}$, and the order of these vector components. A semi-coupled user-task design becomes a coupled design if the proper user-task sequence is disregarded. On the other hand, it may be possible to reorder a full matrix into a triangular matrix and thereby revealing the proper sequence.

In the existing AD literature, two different algorithms have been used for reordering a design matrix. The first algorithm, which is based on Steward (1981), was presented by Suh (1990). This algorithm reorders a square matrix into a lower-left triangular matrix whenever it is possible. The second algorithm, which is based on King (1980), was used by Helander and Jiao (2002). This algorithm clusters the non-zero elements in any matrix, as into blocks along the diagonal of the matrix much as possible and with a skew towards the upper left corner. The two algorithms have been reproduced in Appendix 2.

For the purpose of comparison between the algorithms, consider the following design equations:

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \\ UG_4 \\ UG_5 \end{Bmatrix} = \begin{bmatrix} X & & & & \\ X & X & X & & \\ X & & X & & \\ & X & X & X & X \\ & X & X & & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \\ UA_4 \\ UA_5 \end{Bmatrix} \quad (3.9)$$

Applying Steward's algorithm to Equation 3.9 results in

$$\begin{Bmatrix} UG_1 \\ UG_3 \\ UG_2 \\ UG_5 \\ UG_4 \end{Bmatrix} = \begin{bmatrix} X & & & & \\ X & X & & & \\ X & X & X & & \\ & X & X & X & \\ & X & X & X & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_3 \\ UA_2 \\ UA_5 \\ UA_4 \end{Bmatrix} \quad (3.10)$$

On the other hand, applying King's algorithm to Equation 3.9 results in

$$\begin{Bmatrix} UG_2 \\ UG_3 \\ UG_1 \\ UG_4 \\ UG_5 \end{Bmatrix} = \begin{bmatrix} X & X & & & X \\ X & & & & X \\ & & & & X \\ X & X & X & X & \\ X & X & X & & \end{bmatrix} \begin{Bmatrix} UA_3 \\ UA_2 \\ UA_5 \\ UA_4 \\ UA_1 \end{Bmatrix} \quad (3.11)$$

Equations 3.10 and 3.11 show that Steward's algorithm turns the original $[U]$ into a triangular matrix and reveals the proper user-task sequence for a semi-coupled user-task design, but King's algorithm does not; King's algorithm merely defines the clusters. Therefore, for revealing a proper user-task sequence, Steward's algorithm should be used instead of King's.

However, it should be noted that King's algorithm may be useful when a design matrix does not have an equal number of rows and columns, because Steward's algorithm can only be performed on a square matrix. In this case, iteration is unavoidable and there is no specific correct operating sequence; nevertheless, performing King's algorithm can help to identify the clusters of dependencies.

The usefulness of Steward's algorithm can be exemplified using the workstation design that has been described previously. Suppose an analyst formulates the UGs and UAs in the following order:

UG_1 = Feet are supported when the upper part of the legs are close to horizontal.

UG_2 = Microscope eyepiece in front of eyes when the back is relaxed.

UG_3 = Elbows are supported when the shoulders are relaxed.

UA_1 = Adjust chair height.

UA_2 = Adjust microscope height.

UA_3 = Adjust table height.

The design equation will be formulated as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \end{Bmatrix} = \begin{bmatrix} X & & \\ X & X & X \\ X & & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \end{Bmatrix} \quad (3.12)$$

Applying Steward's algorithm to Equation 3.12 results in

$$\begin{Bmatrix} UG_1 \\ UG_3 \\ UG_2 \end{Bmatrix} = \begin{bmatrix} X & & \\ X & X & \\ X & X & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_3 \\ UA_2 \end{Bmatrix} \quad (3.13)$$

Hence, the microscope workstation is actually a semi-coupled design with a particular user-task sequence, and this is revealed by performing Steward's algorithm.

This example shows that when a set of UGs and UAs results in a square but full design matrix, Steward's matrix reordering algorithm should be applied to investigate whether the matrix can be made triangular. It should be noted that reordering a design matrix does not change the physical design of the product; rather, it reflects a change in the planned sequence of the user tasks.

3.2.2 Redundant Designs

A redundant design in AD is defined as a design that has more functional requirements (FRs) than design parameters (DPs) (Suh, 2001). According to the third theorem of AD, the redundancy characteristic of a design can be removed by fixing appropriate design parameters. This concept applies to user-task designs too. For the purpose of illustration, consider a workstation design that consists of the following components:

- A height adjustable footrest.
- A height adjustable chair.
- A height adjustable worktable.

The primary UGs for such a workstation design, from an office ergonomics perspective, are as follows:

UG_1 = Feet are supported when the upper part of the legs are close to horizontal.

UG_2 = Elbows are supported when the shoulders are relaxed.

To achieve these UGs, users of different sizes have to make adjustments to the workstation, and the UAs can be stated as follows:

UA_1 = Adjust footrest height.

UA_2 = Adjust chair height.

UA_3 = Adjust table height.

Therefore, the design equation is as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \end{Bmatrix} = \begin{bmatrix} X & X & \\ & X & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \end{Bmatrix} \quad (3.14)$$

This is a redundant design, because there are more UAs than UGs in the design equation. The characteristic of such a redundant design changes according to the attribute that will be held constant. If user chooses to fix the height of the footrest, then the workstation becomes a semi-coupled user-task design:

$$\begin{Bmatrix} UG_1 \\ UG_2 \end{Bmatrix} = \begin{bmatrix} X & \\ X & X \end{bmatrix} \begin{Bmatrix} UA_2 \\ UA_3 \end{Bmatrix} \quad (3.15)$$

If user chooses to fix the height of the table, then the workstation also becomes a semi-coupled (but different) design:

$$\begin{Bmatrix} UG_1 \\ UG_2 \end{Bmatrix} = \begin{bmatrix} X & X \\ & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \end{Bmatrix} \quad (3.16)$$

If user can chooses to fix the height of the chair, then the workstation becomes an uncoupled design:

$$\begin{Bmatrix} UG_1 \\ UG_2 \end{Bmatrix} = \begin{bmatrix} X & \\ & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_3 \end{Bmatrix} \quad (3.17)$$

Therefore, a redundant user-task design can be uncoupled if the appropriate variables are held constant.

In AD, an uncoupled design is more desirable than a redundant design. However, Suh (1990) also noted that redundant designs are common in situations where safety and

reliability are critical. From a product usability perspective, a redundant user-task design has the potential to provide user with greater flexibility; users can achieve a set of UGs using different sets of UAs. However, redundancy also means that users need to decide which set of UAs to choose or which particular UAs to disregard. Hence, further studies are required to understand the effects of redundancy on product usability and user's perceived complexity.

3.2.3 Functional Analysis and Actual User Behavior

The dual-domains model in Figure 3.1 is an engineering model rather than a psychological model. The UGs and UAs constitute goals and solutions for a designer during a design process. Similarly, they constitute goals and solutions for a user; task performance, by definition, is a goal-directed behavior (Annett, 2004). However, users may not conceive of their tasks in terms of explicit goals and actions. Furthermore, researchers have reported that the strategy that people actually use in performing a task -- even in a very simple one -- can be very complex (Kieras and Meyer, 2000; Rasmussen et al., 1994).

Therefore, the proposed method aims to provide a functional analysis, rather than a behavioral description, of user tasks (see Annett, 2004). Given a set of UGs and UAs and a planned user-task sequence, the proposed method predicts usability by identifying coupling, but this is different from predicting the actual user behavior. For instance, for semi-coupled user-task designs, the proposed method is able to reveal the correct sequence for performing the user tasks. However, in actual usage, a user may not understand the correct sequence and ends up in performing multiple iterations.

For example, the UGs and UAs for the microscope workstation design described above are as follows:

UG_1 = Feet are supported when the upper part of the legs are close to horizontal.

UG_2 = Elbows are supported when the shoulders are relaxed.

UG_3 = Microscope eyepiece in front of eyes when the back is relaxed.

UA_1 = Adjust chair height.

UA_2 = Adjust table height.

UA_3 = Adjust microscope height.

As shown previously, this is a semi-coupled user-task design. However, if a user does not understand the interactions, he or she will require several iterations to satisfy all the UGs (as illustrated in Figure 3.4). Suh (2005) referred to this type of complexity, which is caused by a lack of knowledge of the correct procedure for operating a decoupled design, as *time-independent imaginary complexity*.

Action Sequence		User Goals		
		Feet are supported (UG_1)	Elbows are supported (UG_2)	Microscope eyepiece at eye level (UG_3)
Step 0	Initial Situation	✗ (not satisfied)	✗	✗
Step 1	Adjust chair height (UA_1)	✓ (satisfied)	✗	✗
Step 2	Adjust microscope height (UA_3)	✓	✗	✓
Step 3	Adjust table height (UA_2)	✓	✓	✗
Step 4	Adjust chair height (UA_1)	✗	✗	✓

Figure 3.4. A user's lack of understanding of the interactions in a human-machine system may result in unnecessary task iterations.

3.2.4 Hierarchical Structure of User Tasks

As shown by the literature review, hierarchical structure is a useful technique for describing user tasks, and it is used in many human factors and human-computer interaction (HCI) methods. Hierarchical structures can also be used with the proposed method. The UGs in the goal domain constitute a *goal hierarchy*, and the UAs in the

action domain constitute an *action hierarchy*. *User-task decomposition* refers to the process of decomposing higher level UGs and UAs into lower levels.

The attributes in the goal and action hierarchies are related in a general manner, which can be explained using Figure 3.5. UG_1 and UA_1 constitute a high-level user task (UT), and they have a “what-how” relationship; “what” refers to what a user wants to achieve, “how” refers to how the user can achieve it. The attributes in a higher level can be perceived as the “why”s, because they provide the rational for selecting the attributes in the lower level.

Figure 3.5. User-task decomposition results in a goal hierarchy and an action hierarchy.

In a design process, the designer usually starts by identifying a set of high-level UGs and selecting the corresponding UAs. Subsequently, the high-level UTs are decomposed according to the designer’s intentions and the design constraints. Through this decomposition process, the designer is transforming his or her design intents, which are expressed by higher level design equations, into a realizable detailed design, which is expressed by the lowest level design equation. This decomposition process is a problem and solution reframing process, which is based on the concepts of design hierarchy and

zigzagging in AD. However, such a hierarchical concept is also analogous to the idea of expanding the method hierarchy in GOMS and the top-down approach in HTA.

In the above sections, a microscope workstation design was analyzed at one particular hierarchical level. Actually, the analysis can begin from a higher hierarchical level as well:

UG_0 = Good work posture.

UA_0 = Adjust workstation.

For the purpose of representing and analyzing mappings across several hierarchical levels, a *master design matrix* can be used (Suh, 2005) (see Figure 3.6). For the workstation example, the set of UTs is uncoupled at the first hierarchical level, but it is semi-coupled at the second hierarchical level.

		UA_0		
		UA_1	UA_2	UA_3
UG_0	UG_1	X		
	UG_2	X	X	
	UG_3	X	X	X

Figure 3.6. A master design matrix, which takes a tabular form, can be used to represent mappings across hierarchical levels.

Analyzing a design at various hierarchical levels is useful, because a design that appears to be uncoupled at a lower level may actually be coupled at a higher level, and ultimately, the satisfaction of the higher-level UGs is most important. According to Weir (1991), some researchers perceive that a good interaction design is one that is a one-to-one mapping from *operator control actions* to *actions on the domain*. Weir criticized this perception as simplistic, and he used a washing machine design example to support his case:

The traditional type of washing machine comes with four basic functions: heat, wash, rinse, and spin. The user selects each of these functions in turn to perform a washing cycle; hence, it has a *one-to-one mapping from user actions to domain actions*. The modern type of washing machine has the same basic functions but it is easier to use because it provides a range of pre-determined cycles; it has a *one-to-many mapping from user actions to domain actions*.

Weir's (1991) observations can be discussed using the concept of goal and action hierarchies. The user's high-level goal is to wash his garments using a desired cycle, which consists of a desired duration and sequence for each part of the cycle and the right water temperature. Hence, UG_1 can be stated as "desired washing cycle", and the corresponding UA_1 for the traditional washing machine can be stated as "control washing operation". Assuming that the user interface has four buttons for activating and terminating each of the basic functions, the UT can be decomposed into the following UGs and UAs:

UG_{11} = Heat stage.

UG_{12} = Wash stage.

UG_{13} = Rinse stage.

UG_{14} = Spin stage.

UA_{11} = Depress heat button.

UA_{12} = Depress wash button.

UA_{13} = Depress rinse button.

UA_{14} = Depress spin button.

Figure 3.7 illustrates the master design matrix.

		UA_1			
		UA_{11}	UA_{12}	UA_{13}	UA_{14}
UG_1	UG_{11}	X			
	UG_{12}		X		
	UG_{13}			X	
	UG_{14}				X

Figure 3.7. UGs to UAs mapping for the traditional washing machine.

For the modern washing machine, UA_1 can be stated as “select pre-determined cycle type”. Consequently, there is no need to decompose the UT because of the automatic functionality of the washing machine. Hence, the design equation is as follows:

$$\{UG_1\} = [X]\{UA_1\} \quad (3.18)$$

This discussion shows that the decoupling concept that is advocated in this research is different from Weir’s (1991) one-action-one-control design concept, which may or may not facilitate a one-to-one mapping between UGs and UAs. Furthermore, it is important to consider abstraction levels when one analyzes user tasks; doing so shows that the traditional and the modern washing machines have uncoupled UT designs. Which is a better design depends on what UGs a user really has.

This discussion also shows that users can have multiple levels of intent, but automation can make the sub-intents irrelevant (cf. Beyer and Holtzblatt, 1998). The more direct is the system, the better. When a UT is decomposed to very low levels, usability decreases with the growing number of UAs that a user needs to perform.

3.3 Quartet-Domains Framework

The framework presented in Section 3.2 can be used to analyze the degree of coupling in a set of user tasks. The goal and action domains can be perceived as a functional model of a user interaction design. However, the cause of coupling may lie in the inner workings of a system -- the structural model. In other words, couplings between UGs and UAs may be caused by how the product is designed to function. Therefore, it is necessary to extend the dual-domains framework to include a structural model. Otherwise, an analyst can only tell whether a set of user tasks is coupled or not, but not identify the cause of the coupling and suggest appropriate design changes.

In Suh's (1990) Axiomatic Design (AD), a product structure is represented by a functional domain, a physical domain, and their interrelationships. The functional domain contains a minimum set of functional requirements (FRs) that characterizes the functions of a product. The physical domain contains design parameters (DPs) that characterize the solution that will satisfy the FRs. DPs can be physical concepts, physical components, or engineering variables of a system.

Functional and physical attributes may belong to different aspects of design; some attributes may belong to more than one aspect. For the purpose of usability analysis, only FRs and DPs that are related to UGs and UAs are considered. These FRs and DPs are referred to as FR^u s and DP^u s. A variable-illumination ceiling lamp can be used to illustrate the characteristics of UG, FR^u s, DP^u s, and UAs:

- UGs are formulated in terms of user's desired state of the system, such as "desired amount of light" -- what the user wants to achieve.
- FR^u s are formulated in terms of engineering functions of the product, such as "provide illumination adjustability" -- what the product should be able to accomplish.
- DP^u s are formulated in terms of engineering parameters of the product, such as "variable resistance" -- how the engineer plans to satisfy the FR^u s.

- UAs are formulated in terms of user operations, such as “rotate light switch” -- what the user has to perform to achieve the UGs.

A usability analysis framework consisting of four design domains is illustrated by Figure 3.8. The goal domain and the action domain, together, characterize the tasks that a user has to perform, while the functional and physical domains, together, characterize the product structure. The mapping across the domains represents the design decisions.

Figure 3.8. Quartet-domains model for analyzing user task designs.

The mapping from the goal domain to the functional domain represents decision-making regarding the functions of a product. As pointed out by Kieras (1997), sometimes usability problems are not caused by poor designed user interfaces, but they are caused by poor selection of functions. The mapping from the functional domain to the physical domain represents engineering design decisions, and it is well discussed in the AD literature (e.g., Suh, 1990; Suh, 2001). The mapping from the physical domain to the action domain represents decision-making regarding the design of user interface.

Design equations can be used to represent the relationships between UGs and FR^u s, between FR^u s and DP^u s, and between DP^u s and UAs:

$$\{UG\} = [A]\{FR''\} \quad (3.19)$$

$$\{FR''\} = [B]\{DP''\} \quad (3.20)$$

$$\{DP''\} = [C]\{UA\} \quad (3.21)$$

Hence,

$$\{UG\} = [A][B][C]\{UA\} \quad (3.22)$$

and

$$[U] = [A][B][C] \quad (3.23)$$

where $[U]$ is the design matrix in Equation 3.1.

Therefore, the overall complexity of a set of user tasks (UTs) is determined by the decisions that are made in each of the facets of design. The relationships between the UGs and the UAs determine the directness of an interaction, but the UAs are also constrained by the specification of the product functions, the conceptualization of the engineering design solutions, and the design of the user interface.

Simon (1969) stated a parable that is commonly referred to as “Simon’s Ant”: to predict an ant’s path, it is neither sufficient to understand the ant’s goals, nor to map the sand dunes; one must understand both in relation to the other. Analogously, to identify and understand structural usability problems, we need to map the relationships between all the design domains, which include the goal domain, the functional domain, the physical domain, and the action domain. Usability problems can be caused by poor user interface design or poor engineering design. By considering both aspects of design, we will achieve a more complete understanding.

The design equations express the design decisions and the designer's intentions. Consequently, coupling between different pairs of adjacent design domains has different implications:

- Coupling between the goal and functional domains can be conceptualized as the result of the functional specification of the product.
- Coupling between the functional and physical domains can be conceptualized as the result of engineering conceptual design.
- Coupling between the physical and action domains can be conceptualized as the result of user interface design.

Identifying the locus of coupling is important, because it serves as a guide for improving the design.

One may question the locus of coupling for the two-knobs water faucet design that has been described in Section 3.2. There appears to be two alternative answers to this question, which of these is correct depends on how the original designer conceived the FR^us. If the FR^us and DP^us were as follows:

FR_1^u = Combined-flow-rate adjustability.

FR_2^u = Temperature adjustability.

DP_1^u = Orifice size of the hot-water valve.

DP_2^u = Orifice size of the cold-water valve.

Then coupling lies between the functional and physical domains:

$$\begin{Bmatrix} FR_1^u \\ FR_2^u \end{Bmatrix} = \begin{bmatrix} X & X \\ X & X \end{bmatrix} \begin{Bmatrix} DP_1^u \\ DP_2^u \end{Bmatrix} \quad (3.24)$$

If the FR^us were specified as follows:

FR_1^u = Cold-water-flow-rate adjustability.

FR_2^u = Hot-water-flow-rate adjustability.

Then coupling lies between the goal and functional domains:

$$\begin{Bmatrix} UG_1 \\ UG_2 \end{Bmatrix} = \begin{bmatrix} X & X \\ X & X \end{bmatrix} \begin{Bmatrix} FR_1^u \\ FR_2^u \end{Bmatrix} \quad (3.25)$$

Therefore, the coupling of the two-knobs water faucet design lies between either the goal and functional domains, or the functional and physical domains. Which is the case depends on the designer's original intentions. In the first case, the designer selected a suboptimum set of FR^u s; in the second case, the designer selected a good set of FR^u s but a suboptimum set of DP^u s.

Three examples are presented in the following sub-sections to illustrate how different products can be analyzed based on the quartet-domains model.

3.3.1 Microscope Workstation

The microscope workstation example, which has been presented in Section 3.2, can be used to illustrate coupling between the goal and functional domains. As stated earlier, the UGs can be stated as follows:

UG_1 = Feet are supported when the thighs are parallel to the floor.

UG_2 = Elbows are supported when the shoulders are relaxed.

UG_3 = Microscope eyepiece in front of eyes when the back is relaxed.

To fit different users, the workstation consists of an adjustable chair, an adjustable worktable, and a microscope that is placed in the adjustable worktable and had an adjustable eyepiece. Hence, the FR^u s that were selected by the designer of the workstation can be stated as follows:

FR_1^u = Chair-height adjustability.

FR_2^u = Table-height adjustability.

FR_3^u = Microscope-eyepiece-height adjustability.

The DP^u 's that were selected by the designer can be identified as:

DP^u_1 = Chair-height adjustment mechanism.

DP^u_2 = Table-height adjustment mechanism.

DP^u_3 = Microscope-eyepiece-height adjustment mechanism.

As stated earlier, the corresponding user actions (UAs) are:

UA_1 = Adjust the chair-height adjustment mechanism.

UA_2 = Adjust the table-height adjustment mechanism.

UA_3 = Adjust the microscope-eyepiece-height mechanism.

Therefore, the design equations are as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \end{Bmatrix} = \begin{bmatrix} X & & \\ X & X & \\ X & X & X \end{bmatrix} \begin{Bmatrix} FR^u_1 \\ FR^u_2 \\ FR^u_3 \end{Bmatrix} \quad (3.26)$$

$$\begin{Bmatrix} FR^u_1 \\ FR^u_2 \\ FR^u_3 \end{Bmatrix} = \begin{bmatrix} X & & \\ & X & \\ & & X \end{bmatrix} \begin{Bmatrix} DP^u_1 \\ DP^u_2 \\ DP^u_3 \end{Bmatrix} \quad (3.27)$$

$$\begin{Bmatrix} DP^u_1 \\ DP^u_2 \\ DP^u_3 \end{Bmatrix} = \begin{bmatrix} X & & \\ & X & \\ & & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \end{Bmatrix} \quad (3.28)$$

Hence,

$$[U] = \begin{bmatrix} X & & \\ X & X & \\ X & X & X \end{bmatrix} \quad (3.29)$$

Analyzing the microscope workstation design based on the quartet-domains model shows that the one-to-many mapping is caused by the selection of FR^u s. Therefore, one could try to improve the design by selecting a different set of FR^u s to uncouple the design. A possible solution, as suggested by Helander and Lin (2000) consists of an adjustable footrest, an adjustable worktable, and a separate adjustable microscope table. The FR^u s of this design can be stated as:

FR''_1 = Footrest-height adjustability.

FR''_2 = Table-height adjustability.

FR''_3 = Microscope-table-height adjustability.

The mapping from the original set of UGs to this new set of FR^u s can be represented as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \end{Bmatrix} = \begin{bmatrix} X & & \\ & X & \\ & & X \end{bmatrix} \begin{Bmatrix} FR''_1 \\ FR''_2 \\ FR''_3 \end{Bmatrix} \quad (3.30)$$

This example demonstrates coupling between goal and functional domains and how the quartet-domains model can be used for design analysis. It also illustrates how a design can be improved by selecting a different set of functions. Commonly, engineers try to improve a design by making changes in the physical domain of a design, but sometimes the problems originate from the functional specification.

3.3.2 Manual Single-Lens-Reflex Film Camera

In this sub-section, a camera is used to illustrate coupling between the functional and physical domains. The model that was studied is a Nikon FM2, but the design of this camera is typical of manual single-lens-reflex (SLR) film cameras. For photo taking, the key UGs in the operating the camera can be stated as follows:

UG_1 = Desired image sharpness.

UG_2 = Desired level of depth-of-field.

UG_3 = Desired amount of motion blur.

UG_4 = Desired image brightness.

UG_5 = Desired image captured.

Based on the user interface design, the UAs were identified as follows:

UA_1 = Rotate lens focusing ring.

UA_2 = Rotate aperture ring.

UA_3 = Rotate shutter speed ring.

UA_4 = Press shutter release.

Subsequently, the mapping between the UGs and the UAs were represented as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \\ UG_4 \\ UG_5 \end{Bmatrix} = \begin{bmatrix} X & & & \\ X & X & & \\ & & X & \\ & X & X & \\ X & X & X & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \\ UA_4 \end{Bmatrix} \quad (3.31)$$

Since the number of UAs is less than the number of UGs, this is a coupled user-task design. In order to identify the locus of coupling, there is a need to study the product structure, and thereby the mappings between the other design domains. The UGs were translated into the following FR^u s:

FR^u_1 = Provide user control for focusing picture.

FR^u_2 = Provide user control for adjusting depth-of-field.

FR^u_3 = Provide user control for adjusting degree of motion blur.

FR^u_4 = Provide user control for adjusting degree of exposure.

FR^u_5 = Provide user control for exposing frame.

The DP^u s were identified as follows:

DP^u_1 = Distance between lens and film.

DP^u_2 = Aperture size.

DP^u_3 = Shutter speed.

DP^u_4 = Shutter release mechanism.

Therefore, the design equations are as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \\ UG_4 \\ UG_5 \end{Bmatrix} = \begin{bmatrix} X & & & & \\ & X & & & \\ & & X & & \\ & & & X & \\ X & X & X & X & X \end{bmatrix} \begin{Bmatrix} FR^u_1 \\ FR^u_2 \\ FR^u_3 \\ FR^u_4 \\ FR^u_5 \end{Bmatrix} \quad (3.32)$$

$$\begin{Bmatrix} FR^u_1 \\ FR^u_2 \\ FR^u_3 \\ FR^u_4 \\ FR^u_5 \end{Bmatrix} = \begin{bmatrix} X & & & & \\ X & X & & & \\ & & X & & \\ & X & X & & \\ & & & X & \end{bmatrix} \begin{Bmatrix} DP^u_1 \\ DP^u_2 \\ DP^u_3 \\ DP^u_4 \end{Bmatrix} \quad (3.33)$$

$$\begin{Bmatrix} DP^u_1 \\ DP^u_2 \\ DP^u_3 \\ DP^u_4 \end{Bmatrix} = \begin{bmatrix} X & & & \\ & X & & \\ & & X & \\ & & & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \\ UA_4 \end{Bmatrix} \quad (3.34)$$

Equation 3.32 shows that the mapping from the UGs to the FR^us is a one-to-many relationship. This implies that there is a specific correct sequence for satisfying the UGs: UG_1 , UG_2 , UG_3 , and UG_4 should precede UG_5 .

Equation 3.33 shows that the locus of coupling lies between the functional and the physical domains. Two design parameters, DP^u_2 and DP^u_3 , are used to satisfy three functional requirements, FR^u_2 , FR^u_3 , and FR^u_4 . As a result of the coupling, the product may not be able to satisfy the FR^us simultaneously. This results in difficulties for the user in achieving the goals.

Since the number of FR^u s is one more than the number of DP^u s, adding an appropriate DP^u , and hence a corresponding UA, may decouple the user-task design. This implies changes to the underlying structure, rather than only the user interface, of the camera. Two examples of a suitable DP^u are a variable power electronic flash and a film with variable ISO speed. In fact, some recent digital cameras allow the user to manipulate the sensitivity of the photoreceptors. Equation 3.35 can be used to explain why the addition of either of these two features is beneficial to the user.

$$\begin{Bmatrix} FR_1^u \\ FR_2^u \\ FR_3^u \\ FR_4^u \\ FR_5^u \end{Bmatrix} = \begin{bmatrix} X & & & & \\ & X & & & \\ & & X & & \\ & & & X & X \\ & & & & X \end{bmatrix} \begin{Bmatrix} DP_1^u \\ DP_2^u \\ DP_3^u \\ DP_{new}^u \\ DP_4^u \end{Bmatrix} \quad (3.35)$$

where DP_{new}^u = Electronic flash power or variable ISO film speed.

It is important to note that the focus of this method is the satisfaction of user goals. Hence, a design solution that is uncoupled for one set of user goals may be decoupled or coupled for another. For example, the SLR camera will appear to be less complex, if a user is willing to sacrifice some of the goals that were assumed in the analysis. More specifically, Equation 3.36 shows that if a user is willing to sacrifice the depth-of-view of a picture (UG_2), the set of user tasks becomes decoupled, even though the physical design of the camera remains the same. This observation may explain to why some camera manuals instruct the user to choose between aperture-priority and shutter-priority strategies -- they are asking the users to change their goals, so as to overcome the limitation of the cameras.

$$\begin{Bmatrix} UG_1 \\ UG_3 \\ UG_4 \\ UG_5 \end{Bmatrix} = \begin{bmatrix} X & & & \\ & X & & \\ & & X & X \\ X & X & X & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_3 \\ UA_2 \\ UA_4 \end{Bmatrix} \quad (3.36)$$

Different classes of users, such as amateurs and experts, will have different sets of UGs for a product. For example, usually, only professional photographers would be interested in controlling the depth-of-field, such as in a situation where he or she wants the background as well as the close foreground to be blurred, so that there are less distractions in the picture. Hence, a design may need to be analyzed with respect to different sets of UGs, and one design may or may not be suitable for different classes of users, and Sometimes, coupling may be avoided by hiding some of the more advanced functions from the novice users or by specifying variations of the design (cf. Jensen, 2003).

3.3.3 Manual Point-and-Shoot Film Camera

A Voigtländer Vitomatic I 35mm manual camera can be used to illustrate coupling between the physical and action domains. This camera has a user interface design that is different from most manual cameras: it has a “universal setting ring” and a “shutter speed ring”. The universal setting ring controls the aperture, while the shutter speed ring controls both aperture and shutter. Moreover, the two controls are physically coupled -- turning one of them beyond a certain point will turn the other one as well. Hence, there are couplings between the DP^u s and the UAs :

$$\begin{Bmatrix} DP_1^u \\ DP_2^u \end{Bmatrix} = \begin{bmatrix} X & X \\ X & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \end{Bmatrix} \quad (3.37)$$

where DP_1^u = aperture size, DP_2^u = shutter speed, UA_1 = rotate universal setting ring, and UA_2 = rotate shutter speed ring.

This example illustrates coupling between the physical and the action domains, which is due to the design of the user interface. Note that the design of the user interface is not

directly specified in the design equations; it is the actions that are allowed by the user interface that are stated.

3.4 Method Procedure

Based on the above case studies, a generalized procedure for performing an analysis based on the quartet-domains framework is proposed as shown in Figure 3.9.

Figure 3.9. A systematic procedure for analyzing a human-machine system based on the quartet-domains model.

To begin an analysis, specify the UGs, FR^us, DP^us, and UAs. Following that, formulate the design matrices $[A]$, $[B]$, and $[C]$ by identifying the interactions between the design attributes based on observation and, if available, the designer's knowledge. It should be noted that the proposed method does not provide the set of high-level goals for beginning an analysis; they are provided from sources external to this method. The ability to specify a suitable set of user goals for design and analysis depends on the expertise of the designer and his understanding of the targeted users. Interviews with potential users and observations of users interacting with similar or existing systems are also helpful to establish goals. When there is more than one set of UGs or UAs, there is a need to perform an analysis for each set of UGs or UAs.

Subsequently, calculate the user-task design matrix $[U]$, which is the product of $[A]$, $[B]$, and $[C]$. If $[U]$ is a diagonal matrix, then this is an uncoupled user-task design and there are no constraints on the user-task procedure. If $[U]$ is a triangular matrix, then this is a semi-coupled user-task design and there is a specific correct sequence for performing the tasks. If $[U]$ is a full matrix (that is, the matrix is neither diagonal or triangular) and it is square matrix (which is a matrix that has an equal number of rows and columns), then perform Steward's matrix reordering algorithm to determine whether $[U]$ can be transformed into a triangular matrix or not. If this is not possible, or if $[U]$ is not a square matrix, this is a coupled user-task design and there will be potential usability problems.

If $[U]$ is a full matrix, based on matrix algebra, there are two possible causes. First, the mapping between one or more pairs of adjacent design domains may be coupled; that is, at least one of $[A]$, $[B]$, or $[C]$ is a full matrix (see Figure 3.10). In this case, the system may be improved by performing changes within the coupled pair of design domains. For instance, if the mapping between the functional and physical domains is coupled, the system may be improved by selecting a different set of DP^us, which is the case in the manual SLR camera study. However, design changes have a cascading effect towards the right side of the quartet-domains model. For instance, changes to the FR^us

of a human-machine system may lead to a need for changes in the DP^u s, which may in turn lead to a need for changes in the UAs .

The second possible cause is that the mapping between two or more pairs of adjacent design domains is semi-coupled and the multiplying effect results in full coupling (see Figure 3.11). In this case, coupling is an emergent property of the system, and any improvements usually require changes to the entire, instead of a specific aspect, of the human-machine system.

Figure 3.10. Coupling between the functional (FR^u s) and physical (DP^u s) domains results in complexity for user. Complexity can be reduced by making localized changes within these two domains, but a modified system needs to be reanalyzed to ensure that the design changes do not introduce any new and unexpected interactions.

Figure 3.11. The multiplying effect of two or more semi-coupled mappings may result in overall full coupling. In this case, since complexity is an emergent property of the

system, it is unlikely that the problem can be corrected by making localized design changes only.

Although a systematic methodological procedure has been presented above, it should be noted that there is some flexibility regarding the analysis. For example, an analyst could choose to analyze directly, without considering the functional and physical domains, the mapping from the goal domain to the action domain. This is possible if the interactions between UGs and UAs are easy to identify. Subsequently, if the set of user-task design is identified as coupled, the analyst can further investigate the problem by including the functional and physical domains in the analysis. Therefore, the details of the procedure can be customized to the context of the analysis.

3.5 Analysis of a Process Control System

Several case studies, which involve existing designs and non-existing conceptual designs, have been presented in the previous sections. These products include water faucets, office workstation, microscope workstation, and manual SLR cameras. One may criticize these consumer products as “trivial systems”, but one should also note that to develop formalized human-machine interaction methods, it is necessary to start simple in order to address problems and issues one at a time (Dix, 2003).

This section presents a case study that involves a more “sophisticated” human-machine system – a process control microworld. Microworlds are computer-based simulations that fit the definition of complex problem solving tasks -- situations that are dynamic, time-dependent, and complex (Quesada et al., 2005).

The purpose of this case study is to further investigate the generalizability of the proposed method. More specifically, the objective is to determine the applicability and

the limitations of the proposed method in modeling and analyzing a typical small-scale process control system.

The structure of this section is as follows. First, an introduction to the microworld that was studied will be provided. Following that, the analysis procedure and results will be presented. At the end of this section, there will be a discussion on the implications of this study.

3.5.1 Introduction to Duress II

Duress II (Dual Reservoir System Simulation) is a thermal-hydraulic microworld that was designed to be representative of industrial processes (see Figure 3.12). Quesada et al. (2005) provided the following description of Duress II:

Duress II has two redundant feedwater streams that can be configured to supply water to either, both, or neither of two reservoirs. Participants control eight valves and two heaters to satisfy the goals of the system: to keep each of the reservoirs at a prescribed temperature (e.g., 40 C and 20 C, respectively) and to satisfy the current water output demand (e.g., 5 liters per second and 7 liters per second, respectively). The system is determined by the laws of conservation of mass and energy, and its state can be described using 37 variables (energies, volumes, valve positions, etc.) that are saved into log files.

Duress II is publicly available (Cognitive Engineering Laboratory at the University of Toronto, 2004). Appendix 3 provides a list of variables in Duress II.

Each of the two main feedwater streams has a maximum capacity of 10 liters per second. Depending on the output demands, there are three general configurations that can be utilized (see Table 3.1). Vicente (1999) referred to these configurations as the single strategy (A), the decoupled strategy (B), and the full strategy (C). (Since the

pumps must be on for there to be any flow at all, their operations are usually not included in an analysis.)

Figure 3.12. A schematic diagram of the Duress II process control microworld that shows the various components and their topological connections (from Vicente, 1999).

However, it is important to note that Vicente's (1999) classification of coupling is different from the one that has been used throughout this thesis -- Vicente's classification is based on the physical arrangement and independent of user goals. The classification in this thesis is based on the mapping between design domains and is stated with respect to a specific set of user goals.

Duress II has been used by several researchers in a number of studies (e.g., Hajdukiewicz et al., 2004; Howie and Vicente, 1998; Miller and Vicente, 2001). Most of these studies focused on how to design better displays for process control systems, and they dealt primarily with the evaluation of the state of the system. In this study, the method was used to model and analyze the user control tasks for Duress II and derive an appropriate set of user actions. Using Norman's (1988) concepts, we can refer to these problems as related to the "gulf of evaluation" and the "gulf of execution", respectively.

Table 3.1 Three general feedwater configuration strategies (adapted from Vicente, 1999).

Strategy	Schematic Illustration	Description
A		<ul style="list-style-type: none"> One pump (P_A) and its connected input valves (VA, $VA1$, and $VA2$) are used to feed both reservoirs. Feasible when the sum of output demands is less than 10 liters per second -- the maximum capacity of either pump.
B		<ul style="list-style-type: none"> Each pump and one of its connected input valves ($VA1$ and $VB2$) are used to feed different reservoirs. Feasible when the sum of output demand for each reservoir is less than or equal to 10 liters per second. Necessary when the sum of output demands is greater than 10 liters per second.
C		<ul style="list-style-type: none"> Both pumps (PA and PB) and their connected valves are used to feed the reservoirs. At least one pump feeds both reservoirs. ($VA1$, $VB1$, and $VB2$ are assumed to be in use for this case study.) Feasible when the sum of output demands is less than or equal to 20 liters per second. Necessary when the one of the output demands is greater than 10 liters per second.

3.5.2 Analysis Procedure and Results

The general procedure of the analysis was as follows:

- 1) Identify the top-level UGs.

- 2) Based on the schematic diagram of Duress II, identify the FR^u s, DP^u s, and UAs for the respective configuration strategy.
- 3) Identify the interactions and formulate the design matrices $[A]$, $[B]$, and $[C]$.
- 4) Calculate the user-task design matrix $[U]$.

The analysis was performed for each of the three configuration strategies, and the results are as follows.

The UGs , FR^u s, DP^u s, and UAs for configuration strategy A (in Table 3.1) were identified and listed in Table 3.2.

Table 3.2 List of design variables in configuration strategy A.

User Goals	Functional Requirements	Design Parameters	User Actions
UG_1 : Desired output water flow rate at reservoir 1.	FR^u_1 : Control of input water flow rate at reservoir 1.	DP^u_1 : Orifice size of valve VA.	UA_1 : Control orifice size of valve VA.
UG_2 : Desired output water flow rate at reservoir 2.	FR^u_2 : Control of output water flow rate at reservoir 1.	DP^u_2 : Orifice size of valve VA1.	UA_2 : Control orifice size of valve VA1.
UG_3 : Desired temperature in reservoir 1.	FR^u_3 : Control of input water flow rate at reservoir 2.	DP^u_3 : Orifice size of valve VO1.	UA_3 : Control orifice size of valve VO1.
UG_4 : Desired temperature in reservoir 2.	FR^u_4 : Control of output water flow rate at reservoir 2.	DP^u_4 : Orifice size of valve VA2.	UA_4 : Control orifice size of valve VA2.
		DP^u_5 : Orifice size of valve VO2.	UA_5 : Control orifice size of valve VO2.
	FR^u_5 : Control of heat input at reservoir 1.	DP^u_6 : Heater setting at reservoir 1.	UA_6 : Control heater at reservoir 1.
	FR^u_6 : Control of heat input at reservoir 2.	DP^u_7 : Heater setting at reservoir 2.	UA_7 : Control heater at reservoir 2.

The design equations for representing strategy A were formulated as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \\ UG_4 \end{Bmatrix} = \begin{bmatrix} X & X & & & & \\ & & X & X & & \\ X & X & & & X & \\ & & X & X & & X \end{bmatrix} \begin{Bmatrix} FR_1^u \\ FR_2^u \\ FR_3^u \\ FR_4^u \\ FR_5^u \\ FR_6^u \end{Bmatrix} \quad (3.38)$$

$$\begin{Bmatrix} FR_1^u \\ FR_2^u \\ FR_3^u \\ FR_4^u \\ FR_5^u \\ FR_6^u \end{Bmatrix} = \begin{bmatrix} X & X & & X & & & \\ X & X & X & X & & & \\ X & X & & X & & & \\ X & X & & X & X & & \\ & & & & & X & \\ & & & & & & X \end{bmatrix} \begin{Bmatrix} DP_1^u \\ DP_2^u \\ DP_3^u \\ DP_4^u \\ DP_5^u \\ DP_6^u \\ DP_7^u \end{Bmatrix} \quad (3.39)$$

$$\begin{Bmatrix} DP_1^u \\ DP_2^u \\ DP_3^u \\ DP_4^u \\ DP_5^u \\ DP_6^u \\ DP_7^u \end{Bmatrix} = \begin{bmatrix} X & & & & & & \\ & X & & & & & \\ & & X & & & & \\ & & & X & & & \\ & & & & X & & \\ & & & & & X & \\ & & & & & & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \\ UA_4 \\ UA_5 \\ UA_6 \\ UA_7 \end{Bmatrix} \quad (3.40)$$

Hence,

$$[U] = \begin{bmatrix} X & X & X & X & & & \\ X & X & & X & X & & \\ X & X & X & X & & X & \\ X & X & & X & X & & X \end{bmatrix} \quad (3.41)$$

Equation 3.41 shows that configuration strategy A results in a coupled user-task design. There is coupling and redundancy between the goal and functional domains, and between the functional and physical domains. The causes were identified as follows:

- 1) The output flow rates are constrained by the input flow rates once the reservoirs have been depleted.
- 2) The input flow rate for each reservoir is determined by a function of the orifice size of valves VA, VA1, and VA2.
- 3) The temperature in each of the reservoir depends on the heat input, the input flow rate, and the output flow rate (see Equation 3.4.2).
- 4) Valve VA leads to redundancy between the functional and the physical domains (see Equation 3.43).

Hence, although there is a one-to-one mapping between the physical and action domains, the overall design is coupled.

The UGs and FR^us for configuration strategies B and C (in Table 3.1) were the same as in strategy A. The DP^us and UAs in strategy B were identified and listed in Table 3.3.

Table 3.3 List of design variables in configuration strategy B.

User Goals	Functional Requirements	Design Parameters	User Actions
UG_1 : Desired output water flow rate at reservoir 1.	FR_1^u : Control of input water flow rate at reservoir 1.	DP_1^u : Orifice size of valve VA.	UA_1 : Control orifice size of valve VA.
UG_2 : Desired output water flow rate at reservoir 2.	FR_2^u : Control of output water flow rate at reservoir 1.	DP_2^u : Orifice size of valve VA1.	UA_2 : Control orifice size of valve VA1.
UG_3 : Desired temperature in reservoir 1.	FR_3^u : Control of input water flow rate at reservoir 2.	DP_3^u : Orifice size of valve VO1.	UA_3 : Control orifice size of valve VO1.
UG_4 : Desired temperature in reservoir 2.	FR_4^u : Control of output water flow rate at reservoir 2.	DP_4^u : Orifice size of valve VB.	UA_4 : Control orifice size of valve VB.
		DP_5^u : Orifice size of valve VB2.	UA_5 : Control orifice size of valve VB2.
		DP_6^u : Orifice size of valve VO2.	UA_6 : Control orifice size of valve VO2.
	FR_5^u : Control of heat input at reservoir 1.	DP_7^u : Heater setting at reservoir 1.	UA_7 : Control heater at reservoir 1.
	FR_6^u : Control of heat input at reservoir 2.	DP_8^u : Heater setting at reservoir 2.	UA_8 : Control heater at reservoir 2.

The design equations for representing strategy B were formulated as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \\ UG_4 \end{Bmatrix} = \begin{bmatrix} X & X & & & & \\ & & X & X & & \\ X & X & & & X & \\ & & X & X & & X \end{bmatrix} \begin{Bmatrix} FR_1^u \\ FR_2^u \\ FR_3^u \\ FR_4^u \\ FR_5^u \\ FR_6^u \end{Bmatrix} \quad (3.42)$$

$$\begin{Bmatrix} FR_1^u \\ FR_2^u \\ FR_3^u \\ FR_4^u \\ FR_5^u \\ FR_6^u \end{Bmatrix} = \begin{bmatrix} X & X & & & & \\ & X & X & X & & \\ & & & X & X & \\ & & & X & X & X \\ & & & & & X \\ & & & & & & X \end{bmatrix} \begin{Bmatrix} DP_1^u \\ DP_2^u \\ DP_3^u \\ DP_4^u \\ DP_5^u \\ DP_6^u \\ DP_7^u \\ DP_8^u \end{Bmatrix} \quad (3.43)$$

$$\begin{Bmatrix} DP_1^u \\ DP_2^u \\ DP_3^u \\ DP_4^u \\ DP_5^u \\ DP_6^u \\ DP_7^u \\ DP_8^u \end{Bmatrix} = \begin{bmatrix} X & & & & & & & \\ & X & & & & & & \\ & & X & & & & & \\ & & & X & & & & \\ & & & & X & & & \\ & & & & & X & & \\ & & & & & & X & \\ & & & & & & & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \\ UA_4 \\ UA_5 \\ UA_6 \\ UA_7 \\ UA_8 \end{Bmatrix} \quad (3.44)$$

Hence,

$$[U] = \begin{bmatrix} X & X & X & & & & & \\ & & & X & X & X & & \\ X & X & X & & & & X & \\ & & & X & X & X & & X \end{bmatrix} \quad (3.45)$$

The DP^us and UAs in configuration strategy C were identified and listed in Table 3.4.

Table 3.4 List of design variables in configuration strategy C.

User Goals	Functional Requirements	Design Parameters	User Actions
UG_1 : Desired output water flow rate at reservoir 1.	FR_1^u : Control of input water flow rate at reservoir 1.	DP_1^u : Orifice size of valve VA.	UA_1 : Control orifice size of valve VA.
UG_2 : Desired output water flow rate at reservoir 2.	FR_2^u : Control of output water flow rate at reservoir 1.	DP_2^u : Orifice size of valve VA1.	UA_2 : Control orifice size of valve VA1.
UG_3 : Desired temperature in reservoir 1.	FR_3^u : Control of input water flow rate at reservoir 2.	DP_3^u : Orifice size of valve VO1.	UA_3 : Control orifice size of valve VO1.
UG_4 : Desired temperature in reservoir 2.	FR_4^u : Control of output water flow rate at reservoir 2.	DP_4^u : Orifice size of valve VB.	UA_4 : Control orifice size of valve VB.
		DP_5^u : Orifice size of valve VB1.	UA_5 : Control orifice size of valve VB1.
		DP_6^u : Orifice size of valve VB2.	UA_6 : Control orifice size of valve VB2.
		DP_7^u : Orifice size of valve VO2.	UA_7 : Control orifice size of valve VO2.
	FR_5^u : Control of heat input at reservoir 1.	DP_8^u : Heater setting at reservoir 1.	UA_8 : Control heater at reservoir 1.
	FR_6^u : Control of heat input at reservoir 2.	DP_9^u : Heater setting at reservoir 2.	UA_9 : Control heater at reservoir 2.

The design equations for describing strategy C were formulated as follows:

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \\ UG_4 \end{Bmatrix} = \begin{bmatrix} X & X & & & & \\ & & X & X & & \\ X & X & & & X & \\ & & X & X & & X \end{bmatrix} \begin{Bmatrix} FR_1^u \\ FR_2^u \\ FR_3^u \\ FR_4^u \\ FR_5^u \\ FR_6^u \end{Bmatrix} \quad (3.46)$$

$$\begin{Bmatrix} FR_1^u \\ FR_2^u \\ FR_3^u \\ FR_4^u \\ FR_5^u \\ FR_6^u \end{Bmatrix} = \begin{bmatrix} X & X & & X & X & X \\ X & X & X & X & X & X \\ & & & X & X & X \\ & & & X & X & X & X \\ & & & & & & X \\ & & & & & & & X \end{bmatrix} \begin{Bmatrix} DP_1^u \\ DP_2^u \\ DP_3^u \\ DP_4^u \\ DP_5^u \\ DP_6^u \\ DP_7^u \\ DP_8^u \\ DP_9^u \end{Bmatrix} \quad (3.47)$$

$$\begin{Bmatrix} DP_1^u \\ DP_2^u \\ DP_3^u \\ DP_4^u \\ DP_5^u \\ DP_6^u \\ DP_7^u \\ DP_8^u \\ DP_9^u \end{Bmatrix} = \begin{bmatrix} X & & & & & & & & \\ & X & & & & & & & \\ & & X & & & & & & \\ & & & X & & & & & \\ & & & & X & & & & \\ & & & & & X & & & \\ & & & & & & X & & \\ & & & & & & & X & \\ & & & & & & & & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \\ UA_4 \\ UA_5 \\ UA_6 \\ UA_7 \\ UA_8 \\ UA_9 \end{Bmatrix} \quad (3.48)$$

Hence,

$$[U] = \begin{bmatrix} X & X & X & X & X & X & & & \\ & & & X & X & X & X & & \\ X & X & X & X & X & X & & X & \\ & & & X & X & X & & & X \end{bmatrix} \quad (3.49)$$

Equations 3.45 and 3.49 show that strategies B and C result in coupled designs too. This is not unexpected, because strategies B and C share the same set of UGs and FR^us with strategy A. But on top of that, strategies B and C have higher degrees of redundancy, because each of them involves additional DP^us. Strategy C has nine DP^us, and it has

higher coupling and redundancy than strategy B, which has eight DP^us; strategy A has seven DP^us.

It was observed that the user-task design matrices in Equations 3.41, 3.45, and 3.49 have more columns than rows, and some of the columns are identical. Hence, one possible approach for reducing redundancy is to integrate or eliminate some of the “how”s (UAs) that determine the same “what”s (UGs) in the matrices. For instance, a design engineer can work towards the integration of UA_1 , UA_2 , and UA_3 , and the integration of UA_4 , UA_5 , and UA_6 for configuration strategy B (see Equation 3.45). One possible method for achieving this is as follows:

- Integrate the controls of input flow rates with the corresponding controls of output flow rates to reduce coupling and redundancy between the goal and functional domains. That is, let the orifice size of valve VA1 equals to the orifice size of valve VO1, the orifice size of valve VB2 equals to the orifice size of valve VO2, during the normal operations of the feedwater system.
- Fix the orifice size of valve VA and VB during the normal operations to reduce redundancy between the functional and physical domains.

Table 3.5 lists the new set of FR^us and DP^us and a corresponding set of new UAs, assuming that the one-to-one mapping between the physical and action domains is maintained. Equations 3.50 to 3.53 represent the mapping between the new design attributes.

Table 3.5. A new set of functional requirements, design parameters, and user actions for configuration strategy B.

Functional Requirements	Design Parameters	User Actions
FR_1^u : Control of input and output water flow rate at reservoir 1.	DP_1^u : Orifice size of valves VA1 and VO1.	UA_1 : Control orifice size of valves VA1 and VO1.
FR_2^u : Control of input and output water flow rate at reservoir 2.	DP_2^u : Orifice size of valves VB2 and VO2.	UA_2 : Control orifice size of valves VB2 and VO2.
FR_3^u : Control of heat input reservoir 1.	DP_3^u : Heater setting of reservoir 1.	UA_3 : Control heater at reservoir 1.
FR_4^u : Control of heat input at reservoir 2.	DP_4^u : Heater setting of reservoir 2.	UA_4 : Control heater at reservoir 2.

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \\ UG_4 \end{Bmatrix} = \begin{bmatrix} X & & & \\ & X & & \\ X & & X & \\ & X & & X \end{bmatrix} \begin{Bmatrix} FR_1^u \\ FR_2^u \\ FR_3^u \\ FR_4^u \end{Bmatrix} \quad (3.50)$$

$$\begin{Bmatrix} FR_1^u \\ FR_2^u \\ FR_3^u \\ FR_4^u \end{Bmatrix} = \begin{bmatrix} X & & & \\ & X & & \\ & & X & \\ & & & X \end{bmatrix} \begin{Bmatrix} DP_1^u \\ DP_2^u \\ DP_3^u \\ DP_4^u \end{Bmatrix} \quad (3.51)$$

$$\begin{Bmatrix} DP_1^u \\ DP_2^u \\ DP_3^u \\ DP_4^u \end{Bmatrix} = \begin{bmatrix} X & & & \\ & X & & \\ & & X & \\ & & & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \\ UA_4 \end{Bmatrix} \quad (3.52)$$

Hence,

$$\begin{Bmatrix} UG_1 \\ UG_2 \\ UG_3 \\ UG_4 \end{Bmatrix} = \begin{bmatrix} X & & & \\ & X & & \\ X & & X & \\ & X & & X \end{bmatrix} \begin{Bmatrix} UA_1 \\ UA_2 \\ UA_3 \\ UA_4 \end{Bmatrix} \quad (3.53)$$

Equation 3.53 shows that the modified configuration results in a semi-coupled user-task design.

It should be noted that the purpose of the above discussion is only to illustrate a possible and straightforward approach for reducing the coupling and redundancy of configuration A. The feasibility of such design changes depend on many other engineering considerations (such as the capacity of the pumps, maintainability, and cost); making design decisions for a specialized system, such as the feedwater system that is simulated by Duress II, will require domain expertise.

3.5.3 Discussion

This study shows that the method that has been developed in this research is capable of modeling and analyzing a process control system such as the feedwater system simulated in the Duress II microworld. The design equations provided a powerful visual summary of the structural differences between the three configuration strategies.

Through the identification of the loci of coupling and redundancy, the design equations were used to make indirect suggestions on how to improve the systems. It was noted that the formulation of the design equations in the analysis was rather easy, because the behavior of Duress II is governed by relatively simple physical laws; modeling more complicated and more dynamic systems would be more difficult. There is an extensive literature discussing the properties of Duress II, but the possibility of simplifying the

control design has not been explored in the past (e.g., Hajdukiewicz et al., 2004; Howie and Vicente, 1998; Miller and Vicente, 2001).

The proposed method can be used to *analyze* an existing design and suggest what interactions can be removed to simplify the design. However, design *synthesis* has not become easier through this method; it is still equally difficult to come up with good design solutions.

This study also demonstrates the advantages of the analytic characteristic of the proposed method. It is important to have models that can make predictions from a technical analysis of a task without the need to fit data against an empirical study of users executing the task; thereby, they can be used early in a design process (John, 2003). For large systems such as process control systems, early evaluations are even more important since the costs of late design changes for such systems may be unaffordable. Furthermore, user-based evaluations may not be able to pinpoint the cause of usability problems as well as the proposed method. This is because users do not think of a human-machine system in terms of user goals, functional requirements, design parameters, user actions, and their interrelationships.

One may argue that a trained usability specialist should be able to reason about the process and draw conclusions that are similar to those in the case studies that have been presented in this thesis. However, one purpose of developing systematic external methods is to make explicit certain design knowledge and formalize it so that a design can be critically and rationally examined.

Furthermore, the idea of “giving the methods away” to those with little formal education in ergonomics is a re-occurring theme (Stanton and Young, 2003). In the author’s opinion, the proposed method is rather straightforward; the concept of decoupling a problem is not new to engineering, and it does not require deep understanding of human factors engineering and cognitive psychology (Sadun, 2001).

Hence, the proposed method has the potential to be accepted by the general population of design engineers.

3.6 Discussion

This section discusses this research's contribution to Suh's (1990) Axiomatic Design (AD) methodology and compares the proposed method with existing analytic and complexity methods in human-machine interaction.

3.6.1 An Extension of Axiomatic Design

The method that has been developed in this research is largely based on the concepts of AD. However, there are several important differences between the proposed method and the existing knowledge in AD applications. First, the proposed method is not conceived as "axiomatic". Instead, it has bases in Norman's (1988) model of human action, the concept of abstraction as a technique for dealing with complexity of design, and Hirschi and Frey's (2002) empirical evidence of the detrimental effects of coupling on human problem solving. While the generalizability of the Independence Axiom in all contexts of design may be arguable, the negative effects of coupling between user goals and user actions are rather obvious.

Secondly, although Suh (1990) presented an AD framework that consists of a customer domain, a functional domain, a physical domain, and a process domain, most of the previous applications of AD, including Helander and Lin (2000), utilized the functional and physical domains only. By introducing a goal domain and an action domain, the proposed method demonstrated how four design domains can be used concurrently.

Furthermore, the distinctions between user goals and functional requirements and between design parameters and user actions advanced the original concepts in Helander et al.'s (2000) study. Without the use of four design domains, it is impossible to include both a functional model (which is a description of how to accomplish a task) and a structural model (which is a description of the inner structure of a system) of human-machine interaction and identify specific loci of coupling.

Lastly, one of the advantages of using matrix equations as a notation is the ability to multiply two equations. This is useful, because we are interested in 1) the interactions between any two design domains, instead of only between adjacent design domains, and 2) the emerging characteristics of two or more one-to-many (semi-coupled) relationships. This idea was noted by Suh (1990), but it was almost never utilized in the previous applications of AD. The proposed method exploits this characteristic of matrix equations and demonstrates its usefulness.

3.6.2 Comparison with Existing Methods

Several analytic methods were reviewed in Section 2.2. A comparison between these methods and the method that has been developed in this research reveals the following main differences. Firstly, the proposed method is an analytic technique, rather than a descriptive technique. Unlike Hierarchical Task Analysis (HTA), the proposed method provides a general and rational criterion for evaluation, and this reduces the dependency on the subjective perception of individual design engineers.

Secondly, the proposed framework includes both a functional model and a structural model. The functional model, which is formed by user goals and user actions, describes human-machine interaction from a user's perceptive. The structural model, which is formed by functional requirements and design parameters, describes the structure of the machine from engineering's perspective. On the other hand, methods such as HTA, Goals, Operators, Methods, Selection (GOMS), and Task-Action Grammar (TAG) do

not consider the underlying structure of a system. Hence, they are not able to suggest design changes when the cause of a usability problem lies in the design of the inner workings of the machine.

Lastly, task Analysis for Error Identification (TAFEI) and User Interface Design with Matrix Algebra (UIDMA) use states and transitions to describe the behavior of a device. Hence, they are more suitable for analyzing systems that are discrete, step-by-step, and have limited and easily identifiable states. On the other hand, the grammatical notation of TAG may be more suitable for computer systems. The proposed method has a different scope; it can be applied to interaction tasks that are simultaneous and continuous.

A review of existing complexity methods was presented in Section 2.3. A comparison between these methods and the method that has been developed in this research draws the following conclusions. Firstly, each of the complexity methods, including the proposed method, has different definitions and measures of complexity. Among the methods that were covered in the review, Kieras and Polson (1985; 1999) Cognitive Complexity Method (CCT) appears to be comparable, in some aspects, to the proposed method. Both methods are related to the task procedural aspects of usability. In addition, CCT includes a device representation and a user's job-task representation, which are analogous to the functional-physical domains and the goal-action domains in the proposed method.

Furthermore, CCT has a similar concept of a one-to-one mapping between the user's job-task representation and the system's representation. However, the proposed method uses design equations instead of product rules. Product rules may appear to be a more natural notation for describing user behavior, but they are less general than design equations; it would be more tedious to describe the usage of the SLR camera and the operation of the Duress II microworld using product rules. While the design equations specified a general set of goals and actions, a product rules system needs to consider all

possible operation conditions. Whether product rules or design equations are a better notation may depend on the type of human-machine system and the purpose of analysis.

Secondly, each of the complexity methods appears to be more suitable to a particular type of user interface. For instance, Rauterberg's (1992; 1996) method can be applied to computer dialogue systems, while Task-To-Action Model (TTA) can be applied to user navigation in an information system. The scope of the proposed method does not include these systems; on the other hand, Rauterberg's method and TTA do not appear to be applicable to the type of systems that were studied in this research.

Lastly, Karwowski's (1997) framework considers complexity of a human-machine system as a function of the number of components, the number of interactions, and the strength of these interactions. In this sense, it shares some similarities with the proposed method. However, unlike the proposed method, Karwowski's framework does not consider user goals. Hence, it is limited to the structural model of human-machine interaction.

There is a special class of methods in human-computer interaction (HCI) that is referred to as *formal methods*, which includes Communicating Sequential Processes (CSP) and Z (Hoare, 1978; Spivey, 1988). These methods usually involve the use of visual or algebraic expressions, and they assume that the transformation from a detailed design to its implementation is the same as the mapping from one mathematical representation to another (Dix et al., 1998). More specifically, a formal method is one that displays the following characteristics (Wing, 1990):

- Based on mathematics.
- Provide means of proving a software specification is realizable.
- Provide means of proving that a software system has been correctly implemented.
- Provide means of proving the properties of a software system without necessarily running it.

The method that has been proposed in this research is *formalized* in the sense that it has a rather systematic procedure. However, it is not strictly formal in the mathematical sense; natural language is used to express the design requirements. Nevertheless, it does share certain similarities with formal methods in HCI. First, the use of mathematical equations as a form of notation allows an analyst to apply metrics to predict the usability of a design, which is the hallmark of formalism in mathematics (Dix, personal communication, 2003). Second, the use of functional and physical domains in describing a design reveals the inner workings of a design. Third, the process of describing the design using the proposed methodology encourages designers to think about their systems in detail, and to consider issues that could otherwise be missed. Fourth, the proposed method uses an abstract model of the system or product for analysis.

3.7 Chapter Summary

This chapter presents a framework and method, which is based on the principles of Suh's (1990) Axiomatic Design (AD), for analyzing complexity in human-machine interaction. The method is proposed as an analytic method for identifying couplings between the goals that a user wants to achieve and the control actions that are designed into a user interface. This type of coupling reduces usability, because it widens the "gulf of execution".

The skeleton framework consists of a goal domain and an action domain. The mapping from the goal domain to the action domain is represented using a user-task design equation. However, to include the structural model into the analysis, the framework was augmented with a functional domain and a physical domain. Using the four design domains, the locus of coupling in a human-machine system can be identified. The analysis proceeds from a set of user goals to a set of user actions. Coupling between different pair of adjacent domains have different implications for design.

The proposed method uses design equations as a notation for representing the interactions between the design domains. Based on the form of the user-task design matrix, a design can be characterized uncoupled, semi-coupled, or coupled. Uncoupled designs provide a straightforward, one-to-one mapping between the user goals (UGs) and the user actions (UAs). Semi-coupled or decoupled designs provide a one-to-many mapping between the UGs and the UAs, and they have a specific operation procedure. Coupled designs provide a many-to-many mapping between the UGs and the UAs.

Several issues regarding the proposed framework and method were discussed. These issues include

- 1) matrix-reordering algorithms,
- 2) redundant designs,
- 3) actual user behavior, and
- 4) user task hierarchies.

Several case studies, which were conducted during the development process of the proposed method, were presented. These studies include

- 1) two water faucets,
- 2) a microscope workstation,
- 3) an office workstation,
- 4) a manual SLR camera, and
- 5) a manual point-and-click camera.

These studies illustrate the application and capabilities of the proposed method and the different types of coupling.

To further understand the generalizability of the proposed method, a case study that involved the Duress II microworld was conducted. This study shows that the proposed method was able to identify the locus of coupling and redundancy in a small-scale process control system and distinguish the structural differences between three configuration strategies for operating the system.

Finally, the proposed method was compared with the methods that were reviewed in Chapter 2. A comparison between this research and previous applications of AD was also presented.

CHAPTER 4. USABILITY GUIDELINES ANALYSIS

As stated earlier in this thesis, guidelines provide a rich source of human-computer interaction (HCI) knowledge. But researchers have reported that they are difficult to use. Since guidelines can be conceived as a set of generic design requirements, understanding and eliminating interactions in guidelines can help to reduce the complexity of guidelines.

This chapter reports on the development of a technique, which is based on Suh's (1990) Axiomatic Design (AD), for analyzing interactions in usability guidelines. This research is a follow up on Helander and Jiao's (2002) study. First, a framework for describing the role of guidelines in software user interface design will be presented. Following that, a technique for decoupling guidelines will be illustrated using a case study. The limitations of this research will be discussed at the end of the chapter.

4.1 Role of Guidelines in Software Design

Software design can be conceived as a mapping process across four design domains: the customer domain, the functional domain, the physical domain, and the process domain (Suh, 2001). Suh stated that between each pair of adjacent design domains, there is a "what-how" relationship. However, we can enrich the description by stating that each triplet of adjoining domains has a "why-what-how" relationship (see Figure 4.1).

Figure 4.1. Software design can be conceived as a mapping process across four design domains (adapted from Suh, 2001).

Design begins by identifying the customer needs (CNs) and specifying these needs as functional requirements (FRs). In Quality Function Deployment (QFD), CNs are referred to as the “voice of customers” (Clausing, 1994). FRs refer to a set mutually independent functional software output and requirements that characterizes the design problem. To satisfy the FRs, a designer conceives a set of a set of software attributes, which are referred to as design parameters (DPs). The DPs are mapped onto a set of algorithms, modules, or program codes, which are referred to as process variables (PVs), for implementation.

Building on top of this framework, software usability guidelines can be perceived as a collection of *generic FRs* (FR^gs) and *generic DPs* (DP^gs) that are related to the usability of a software program. Providing this database to designers is helpful in two ways:

- 1) CNs tend to be abstract and difficult to capture. Furthermore, translating CNs to FRs depends largely on the experience of the designer (Suh, 2001). Providing a set of FR^gs allows a designer to make use of previous design experience.

- 2) Similarly, providing a set of DP^gs enables a designer to, rather than conceive a completely new set of DPs, reuse DPs that have been proven as effective in the past.

We can extend AD's framework for software design by adding two design domains to represent the role of usability guidelines (see Figure 4.2). Guidelines contain generic design goals and generic design solutions, which can be selected and adapted according to a design context.

Figure 4.2. The role of usability guidelines in software design can be represented by adding two design domains to the Axiomatic Design framework.

Design is subjectable to constraints, and there are two general types of constraints in design: 1) input constraints and 2) system constraints (Suh, 2001). Input constraints are specific to the overall design goals; all proposed design solutions must satisfy them. For instance, cost is usually considered as an input constraint. System constraints are

specific to a given design solution; they are the result of higher-level design decisions. For instance, the selection of the software-operating platform (higher-level decision) will result in certain constraints on the look of the user interface of a software program (lower-level decisions). In a latter part of this chapter, the author argues that some usability guidelines are recommended constraints, rather than design goals or design solutions.

4.2 Technique for Decoupling Usability Guidelines

Based on the above framework, AD concepts can be adapted into a technique for analyzing interactions in usability guidelines. This section illustrates the technique using Nielsen's (1994b) usability heuristics as a case study.

Nielsen (1994a) derived a list of nine usability heuristics based on a factor analysis of 209 usability problems. Subsequently, Nielsen (1994b) added a heuristic on "help and documentation" to the list (see Table 4.1). Nielsen (2004) claimed that this list of heuristics provide a broad explanatory coverage of common usability problems and can be employed as general principles for user interface design and evaluation.

First, each of the usability heuristics was studied carefully to identify and extract any phrases that implied a design goal (FR^g s), a design solution (DP^g s), or a design constraint (C). Following that, the dependencies between the extracted FR^g s and DP^g s were determined, and a design equation was formulated to represent their interactions. Since the design matrix showed a coupled system, Steward's matrix re-ordering algorithm was applied to cluster the dependencies into blocks to reveal the loci of coupling.

Subsequently, an attempt was made to reengineer the FR^g s and DP^g s to achieve a new and less coupled set of recommendations. The basic notion of the reengineering process

was to examine and reformulate the FR^gs and DP^gs that were highly coupled with each other. One approach that was used to reformulate the FR^gs and DP^gs was to split them according to two different aspects of design: 1) the visual presentation aspect and 2) the behavior and interaction techniques aspect. The analysis and reengineering cycle was repeated several times until the results were deemed as acceptable.

Table 4.1 Nielsen's (1994b) list of 10 usability heuristics.

1. Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

2. Match between system and the real world

The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world convention, making information appear in a natural and logical order.

3. User control and freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

4. Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

6. Recognition rather than recall

Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. Flexibility and efficiency of use

Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

4.2.1 Identification of Dependencies

The analysis technique begins by classifying the recommendations in the usability guidelines and identifying the interactions. First, each of the usability heuristics were studied carefully to identify and extract any phrases that could be conceived as a FR^g , DP^g , or C (see Table 4.2).

It should be noted that a similar concept of distinguishing between design goals and design solutions exists in usability patterns (Henninger 2001; van Welie et al., 2000) and van Welie et al.'s (1999) layered model of usability (see Figure 4.3). This type of distinction overcomes the limitations of Alexander's (1964) tools for analyzing interactions between design requirements (refer to Section 2.5.3). In addition, it makes design recommendations more goal-oriented, which may be useful for designers (Henninger, 2001).

Table 4.2 FR^g s, DP^g s, and Cs from Nielsen's (1994b) list of usability heuristics.

Heuristic	FR^g s	DP^g s	Cs
1	FR^g_1 : Keep users informed about system status.	DP^g_1 : System feedback within reasonable time.	-
2	FR^g_2 : Match system to real world.	DP^g_2 : User's language.	-
3	FR^g_3 : Provide user control and freedom.	DP^g_3 : Emergency exits.	-
4	FR^g_4 : Maintain consistency.	DP^g_4 : Platform conventions.	-
5	FR^g_5 : Prevent errors.	-	-
6	FR^g_6 : Facilitate recognition rather than recall.	DP^g_6 : Visible objects, actions, and options.	-
7	FR^g_7 : Provide flexibility and efficiency of use.	DP^g_7 : Tailorable accelerators.	-
8	FR^g_8 : Provide aesthetic appeal.	-	C_8 : Minimalist design.
9	FR^g_9 : Help users recognize, diagnose, and recover from errors.	DP^g_9 : Solutions in error messages.	-
10	-	DP^g_{10} : Help and documentation.	-

Figure 4.3. van Welie et al.'s (1999) model of usability consists of four layers: components from the ISO definition of usability, usage indicators which are operational measures of usability, the means for improving the usage indicators, and designer's knowledge of usability.

The recommendations in Heuristics 5, 8, and 10 were judged as incomplete. These heuristics lacked of either an explicit FR^g or an explicit DP^g , as discussed below.

- Heuristic 5 contained a FR^g_5 “prevent errors” without suggesting a concrete DP^g for satisfying this FR^g .
- Heuristic 8 contained a FR^g_8 “aesthetic design” without suggesting a concrete DP^g for satisfying this FR^g . Moreover, it contained a constraint C_8 “minimalist design”. C_8 was an input constraint, because it was deemed as a requirement that all proposed design solutions should satisfy.
- Heuristic 10 contained a DP^g_{10} “help and documentation”, but it did not identify a corresponding FR^g . In addition, one could argue that the scope of this heuristic was of a different category to the others. Help and documentation is not directly

related to the “look and feel” of a software system, and it can be perceived as a different aspect of a user interface.

Hence, heuristics 5, 8, and 10 were not included in the subsequent analysis.

To identify the interactions, a two-way comparison scheme was used to construct a design equation. This scheme was inspired by Su et al. (2003), and it is as follows:

Step I: FR_i^g is used as a criterion to evaluate the relative contribution of DP^g s to FR_i^g . A design matrix $[A]$ is constructed by inserting an “X” whenever a significant relationship has been identified.

Step II: DP_i^g is used as a criterion to evaluate the relative effect of FR^g s. A design matrix $[B]$ is constructed by inserting an “X” whenever a significant relationship has been identified.

Step III: The final design matrix $[C]$ is constructed by placing “X”s at places where there are “X”s in both $[A]$ with $[B]$. The purpose of the two-way comparison is to eliminate any bias that may result from a one-way measurement.

Using the above scheme, a design equation, which represents the interactions between the usability heuristics, was formulated (see Figure 4.4). Each corresponding pair of FR^g s and DP^g s, as suggested by Nielsen (1994b), was deemed to have a dependency. Other dependencies in the design matrix were determined based on the author’s subjective interpretation and references from the HCI literature (Apple, 2002; IBM, 2003; Nielsen, 1993; Norman, 1988; Theo, 1997), see Table 4.3. Figure 4.4 shows that the usability heuristics were highly coupled, and the coupled FR^g s and DP^g s can be grouped into two clusters (as marked by the dashed lines).

	DP_1^g : System feedback within reasonable time.	DP_2^g : User's language.	DP_3^g : Emergency exits.	DP_4^g : Platform conventions.	DP_6^g : Visible objects, actions, and options.	DP_7^g : Tailorable accelerators.	DP_9^g : Solutions in error messages.
FR_1^g : Keep users informed about system status.	X_{11}	X_{12}		X_{14}	X_{16}		
FR_2^g : Match system to real world.	X_{21}	X_{22}		X_{24}	X_{26}		
FR_3^g : Provide user control and freedom.	X_{31}	X_{32}	X_{33}	X_{34}	X_{36}		
FR_4^g : Maintain consistency.		X_{42}		X_{44}	X_{46}		
FR_6^g : Facilitate recognition rather than recall.		X_{62}		X_{64}	X_{66}	X_{67}	X_{69}
FR_7^g : Provide flexibility and efficiency of use.			X_{73}			X_{77}	
FR_9^g : Help users recognize, diagnose, and recover from errors.	X_{91}	X_{92}	X_{93}	X_{94}	X_{96}		X_{99}

Figure 4.4. Nielsen's (1994b) set of usability heuristics appears to be a coupled system. The dashed boxes indicate clusters.

Table 4.3 A brief explanation for each of the dependencies.

Dependency	Reasoning
X_{11}	As suggested by Nielsen (1994b).
X_{12}	Speaking user's language could help to keep user informed of the system status.
X_{14}	Providing a consistent design could help to keep user informed of the system status.
X_{16}	Making objects, actions, and options visible could help to keep user informed of the system status.
X_{21}	How system feedback is designed could affect the match between the system and the real world.
X_{22}	As suggested by Nielsen (1994b).
X_{24}	Nielsen (1993) stated that a system should not force naming conventions or restrictions on objects named by the user. Platform conventions can be in conflict with the real world.
X_{26}	How objects, actions, and options are presented could affect the match between the system and the real world.
X_{31}	IBM (2003) recommended that, to place user in control, the current state of the system should be obvious.
X_{32}	Speaking user's language could help to keep user feeling in control.
X_{33}	As suggested by Nielsen (1994b).
X_{34}	Providing a consistent design could help to keep user feeling in control.
X_{36}	Apple (2002) recommended that to make users feel that they are in control of the computer, the impact of any user actions should be immediately visible.
X_{42}	Nielsen stated (1993) that a system should not force naming conventions or restrictions on objects named by the user. Platform conventions can be in conflict with the real world.
X_{44}	As suggested by Nielsen (1994b).
X_{46}	According to Theo (1997), consistency in presentation means that users should see information and objects in the same logical, visual, or physical way throughout a system.
X_{62}	Speaking user's language could help to reduce user's memory load.
X_{64}	Providing a consistent design could help to reduce user's memory load (Norman, 1988).
X_{66}	As suggested by Nielsen (1994b).
X_{67}	According to Theo (1997), providing "undo"s, "redo"s, and interface shortcuts helps to reduce user's memory load.
X_{69}	Indicating error recovery solutions in error messages could help to reduce user's memory load.
X_{73}	Providing emergency exits could help to improve flexibility and efficiency of use.
X_{77}	As what Nielsen (1994b) suggested.
X_{91}	Providing system feedback could help user to recognize, diagnose, and recover from errors.
X_{92}	Speaking user's language could help user to recognize, diagnose, and recover from errors.
X_{93}	Providing emergency exits could help user to recover from errors.
X_{94}	Providing a consistent design could help user to recognize, diagnose, and recover from errors.
X_{96}	Making objects, actions, and options visible could help user to recognize, diagnose, and recover from errors.
X_{99}	As suggested by Nielsen (1994b).

While the form of a design matrix provides a qualitative metric for characterizing the degree of coupling, a quantitative estimate for the degree of coupling can be calculated using the following formula (Guenov, 2002):

$$\text{Complexity measure} = \sum N_j \ln N_j \quad (4.1)$$

where N_j is the number of dependences or “X”s in the j^{th} column. At a same level of abstraction, a system that has a larger value of Guenov’s complexity measure tends to be relatively more complex than one that has a smaller value. There are 29 dependencies in Figure 4.4, and Guenov’s complexity measure for the design matrix is 43.9.

Due to the way Nielsen (1994b) presented the usability heuristics, one may wrongly assume that each of the heuristics contains a single pair of design goal and solution, and that each heuristic is independent from the others. This analysis shows that some of the FR^g s, such as FR^g_3 , FR^g_6 , and FR^g_9 are actually dependent on most of the DP^g s and some of the DP^g s, such as DP^g_2 and DP^g_4 , affect most of the FR^g s. For instance, FR^g_1 “keep users informed about system status” is dependent on DP^g_1 “system feedback within reasonable time”, but it also dependent on DP^g_2 “user’s language”, DP^g_4 “platform conventions”, and DP^g_6 “visible objects, actions, and options”. Another example is FR^g_9 “help users recognize, diagnose, and recover from errors”, which is dependent on DP^g_9 “solutions in error messages”, DP^g_1 “system feedback within reasonable time”, DP^g_2 “user’s language”, DP^g_3 “emergency exit”, and DP^g_4 “platform conventions”.

Consequently, a designer or analyst may need to determine all of the related DP^g s in order to predict the satisfaction of a FR^g . This type of interactions between usability guidelines results in a complex problem for designers (Alm, 2003).

In some sense, the results of this analysis are not surprising; user interface design problems are known to be complex (Thimbleby, 2004). However, Nielsen’s usability heuristics appear to be formulated in a less than optimum manner. Furthermore, some of the heuristics seem to address several different abstraction levels, and this makes them appear as more coupled than they actually are. For instance, FR^g_4 “maintain

consistency” has relevance to two different aspects of a user interface: the presentation aspect and the behavior and interaction techniques aspect. Presentation aspect refers to the visual design such as feedback and cues -- the “look” of a user interface. Behavior and interaction techniques refer to the behavior of the objects and how a user is supposed to control the objects -- the “feel” of a user interface.

4.2.2 Reengineering the Usability Heuristics

After the interactions in the usability heuristics were identified, an attempt was made to reengineer the heuristics, so as to reduce the degree of coupling. This reengineering process consisted of the following three steps.

It should be noted that, in the AD literature, a common approach for decoupling a design is to select a new set of DPs that would maintain the independence of the FRs; FRs are supposed to be independent by definition (Suh, 1990). However, in the context of usability heuristics, some of the DPs appeared to have little room for change, and some of the FRs seemed to be poorly formulated. Hence, there is a possibility of reengineering the heuristics to reduce couplings by decomposing and integrating some of the FR^gs and DP^gs.

Step I

The design equation in Figure 4.4 shows that DP^g_2 and DP^g_4 affect the most number of FR^gs. It was noticed that “user’s language” and “platform conventions” have an inherent potential to be in conflict with each other. For instance, command names that follow platform conventions may not be familiar and understandable to the user. Since it was difficult to resolve this conflict, the two DP^gs were integrated. In addition, a closer examination revealed that “user’s language” and “platform conventions” can apply to two different aspects of a user interface: the visual presentation aspect and the behavior

and interaction techniques aspect. Hence, to reduce coupling, these two DP^g s and their corresponding FR^g s were decomposed. Therefore, DP^g_2 , DP^g_4 , and their corresponding FR s were reformulated as follows:

$FR^g_{11.1}$ = Match system presentation to the real world and maintain presentational consistency. (If conflict arises, strive for the former)

$DP^g_{11.1}$ = User-familiar and platform-conventional objects and words.

$FR^g_{11.2}$ = Match system behavior and interaction techniques to the real world and maintain consistency in system behavior and interaction techniques. (If conflict arises, strive for the former)

$DP^g_{11.2}$ = User-familiar and platform-conventional behaviors and interaction techniques.

Subsequently, the dependencies in the new set of FR^g s and DP^g s were identified as follows:

$$\left\{ \begin{matrix} FR^g_1 \\ FR^g_3 \\ FR^g_6 \\ FR^g_7 \\ FR^g_9 \\ FR^g_{11.1} \\ FR^g_{11.2} \end{matrix} \right\} = \begin{bmatrix} X & & & & & & \\ & X & & & & & \\ & & X & & & & \\ & & & X & & & \\ & & & & X & & \\ & X & & & & X & \\ & & X & & & & X \\ X & & & & & & \\ & & & & & & X \end{bmatrix} \left\{ \begin{matrix} DP^g_1 \\ DP^g_3 \\ DP^g_6 \\ DP^g_7 \\ DP^g_9 \\ DP^g_{11.1} \\ DP^g_{11.2} \end{matrix} \right\} \quad (4.2)$$

FR^g s and DP^g s, which had interactions with FR^g_2 , FR^g_4 , DP^g_2 , and DP^g_4 were re-evaluated to determine their interactions with the re-formulated FR^g s and DP^g s. $FR^g_{11.1}$ was deemed as dependent on DP^g_1 and DP^g_6 , but $FR^g_{11.2}$ was not. This was because DP^g_1 and DP^g_6 were considered as less relevant to behavior and interaction techniques.

To rearrange the design matrix to identify coupled blocks, Steward's algorithm was applied to Equation 4.2:

$$\left\{ \begin{matrix} FR_{11.2}^g \\ FR_1^g \\ FR_3^g \\ FR_6^g \\ FR_7^g \\ FR_9^g \\ FR_{11.1}^g \end{matrix} \right\} = \begin{bmatrix} X & & & & & & \\ & X & & X & & & \\ & X & X & X & X & & \\ & X & & & X & X & X \\ & & & X & & X & \\ & X & X & X & X & & X \\ & X & & X & & & X \end{bmatrix} \left\{ \begin{matrix} DP_{11.2}^g \\ DP_1^g \\ DP_3^g \\ DP_6^g \\ DP_7^g \\ DP_9^g \\ DP_{11.1}^g \end{matrix} \right\} \quad (4.3)$$

There were 25 dependencies in the design matrix, and Guenov's complexity measure was 33.4.

Step II

In Equation 4.3, the DP^g s that are related to the most number of FR^g s are DP_1^g , DP_6^g , $DP_{11.1}^g$, and $DP_{11.2}^g$. A close examination of these DP^g s and their corresponding FR^g s revealed a strong interaction between FR_1^g and FR_6^g :

- FR_1^g "Keep users informed about system status" is largely dependent on DP_6^g "visible objects, actions, and options".
- FR_6^g "facilitate recognition rather than recall" is also affected by DP_1^g "system feedback within reasonable time".

Since the distinction between these two FR^g s and their corresponding DP^g s at the visual presentation level was not obvious, it might be able to integrate them without losing the original intentions of the heuristics. Hence, two new FR^g s and DP^g s were formulated:

FR_{12}^g = Maintain system transparency.

DP_{12}^g = Visible objects, actions, options, and system status.

Subsequently, the behavior and interaction techniques component of FR_6^g was integrated with FR_7^g :

FR_{13}^g = Provide flexibility, efficiency of use, and reduce user memory load on interaction techniques.

DP_{13}^g = Tailorable accelerators (same as original DP_7^g).

The dependencies in the new set of FR^g s and DP^g s were identified as follows:

$$\left\{ \begin{matrix} FR_{11.2}^g \\ FR_3^g \\ FR_9^g \\ FR_{11.1}^g \\ FR_{12}^g \\ FR_{13}^g \end{matrix} \right\} = \begin{bmatrix} X & & & & & \\ X & X & & X & X & \\ X & X & X & X & X & X \\ & & & X & X & \\ & & & X & X & X \\ X & X & X & & & X \end{bmatrix} \left\{ \begin{matrix} DP_{11.2}^g \\ DP_3^g \\ DP_9^g \\ DP_{11.1}^g \\ DP_{12}^g \\ DP_{13}^g \end{matrix} \right\} \quad (4.4)$$

There were 20 dependencies in the design matrix, and Guenov's complexity measure was reduced to 24.6.

Step III

Equation 4.5 shows that FR_9^g is dependent on all of the DP^g s. This was believed to be caused by the loose definition of FR_9^g – the scope of FR_9^g overlapped with the scope of several other FR^g s. “Recognition of errors” was perceived as part of FR_{12}^g “system transparency”, and “helping users to recover from errors” relies on almost all of the other FR^g s. Therefore, FR_9^g was reformulated as

FR_{14}^g = Provide guidance for error recovery.

DP_{14}^g = Solutions in error messages (same as the original DP_9^g).

Subsequently, the dependencies in the new set of FR^g s and DP^g s were identified as follows:

$$\left\{ \begin{matrix} FR_{11.2}^g \\ FR_3^g \\ FR_{14}^g \\ FR_{11.1}^g \\ FR_{12}^g \\ FR_{13}^g \end{matrix} \right\} = \begin{bmatrix} X & & & & & \\ X & X & & X & X & \\ & & X & X & X & \\ & & & X & X & \\ & & X & X & X & \\ X & X & X & & & X \end{bmatrix} \left\{ \begin{matrix} DP_{11.2}^g \\ DP_3^g \\ DP_{14}^g \\ DP_{11.1}^g \\ DP_{12}^g \\ DP_{13}^g \end{matrix} \right\} \quad (4.5)$$

To rearrange the design matrix to identify coupled blocks, Steward's algorithm was applied to Equation 4.5:

$$\begin{Bmatrix} FR_{11.2}^g \\ FR_{14}^g \\ FR_{11.1}^g \\ FR_{12}^g \\ FR_3^g \\ FR_{13}^g \end{Bmatrix} = \begin{bmatrix} X & & & & & \\ & X & X & X & & \\ & & X & X & & \\ & X & X & X & & \\ X & & X & X & X & \\ X & X & & & X & X \end{bmatrix} \begin{Bmatrix} DP_{11.2}^g \\ DP_{14}^g \\ DP_{11.1}^g \\ DP_{12}^g \\ DP_3^g \\ DP_{13}^g \end{Bmatrix} \quad (4.6)$$

Equation 4.6 shows a reengineered system that is less coupled than the original. The diagonalized cluster contains 17 dependencies, as compared to 24 in Figure 4.4, 25 in Equation 4.3, and 20 in Equation 4.4. Guenov's (2002) complexity measure has been reduced from 43.9 to 19.1.

The final set of reengineered FR^g s and DP^g s is listed in Table 4.4. Coupling remains between the following FR^g s:

- 1) FR_{14}^g "provide guidance for error recovery".
- 2) $FR_{11.1}^g$ "match system presentation to the real world and maintain presentational consistency".
- 3) FR_{12}^g "maintain system transparency".

Among the DP^g s, $DP_{11.1}^g$ and DP_{12}^g are related to the most number of FR^g s.

This study shows that Nielsen's (1994b) usability heuristics constitute a coupled system. The recommended design goals, solutions, and constraints are not well defined, and there are many interactions. It is not a one-to-one mapping that one would assume according Nielsen's presentation of the heuristics. This study also demonstrates how the concepts of AD can be used as a mechanism for representing the interactions between usability guidelines and for reengineering the guidelines.

Table 4.4 Reengineered set of FR^gs and DP^gs.

FR ^g	DP ^g
<i>FR_{11.2}^g</i> : Match system behavior and interaction techniques to the real world and maintain consistency in system behavior and interaction techniques. (If conflict arises, strive for the former)	<i>DP_{11.2}^g</i> : User-familiar and platform-conventional behaviors and interaction techniques.
<i>FR₁₄^g</i> = Provide guidance for error recovery.	<i>DP₁₄^g</i> : Solutions in error messages.
<i>FR_{11.1}^g</i> : Match system presentation to the real world and maintain presentational consistency. (If conflict arises, strive for the former)	<i>DP_{11.1}^g</i> : User-familiar and platform-conventional objects and words.
<i>FR₁₂^g</i> : Maintain system transparency.	<i>DP₁₂^g</i> : Visible objects, actions, options, and system status.
<i>FR₃^g</i> : Provide user control and freedom.	<i>DP₃^g</i> : Emergency exit.
<i>FR₁₃^g</i> : Provide flexibility, efficiency of use, and reduce user memory load on interaction techniques.	<i>DP₁₃^g</i> : Tailorable accelerators.

4.3 Research Limitations

The main limitations of this study are as follows:

- 1) The dependencies in the design matrices were determined based on subjective interpretation of the usability heuristics and understanding of user interface design. References were made to the HCI literature where possible. Therefore, the dependencies were *not* validated empirically and the final results are not deterministic. It is possible to ask usability experts for advice, but chances are they will produce great diversity in results.
- 2) Throughout the analysis and reengineering process, the author aimed at retaining a close match between the intentions of the original set of usability heuristics and the FR^gs and DP^gs that were formulated. However, the heuristics are high-level principles; they are general, and to some extent, ambiguous. This makes the formulation and reformulation rather arbitrary; there are many ways to reformulating the FR^gs and the DP^gs. So far, the author has not yet found a

definite pattern for reformulating the FR^gs and DP^gs, which may or may not emerge over time.

- 3) There is a need to extend this research by testing the actual usefulness and the usability of the reengineered heuristics with practicing user interface designers and usability specialists. However, it should be noted that validating usability guidelines has always been difficult; even testing just a few interacting variables, at a few levels of implementation, requires an experimental design of challenging complexity (Smith, 1986).

4.4 Chapter Summary

Psychological theories and practical experience provide user interface designers with many usability guidelines. However, while these guidelines aim to improve usability, these guidelines themselves have been reported to be difficult to use.

The role of usability guidelines in software design can be conceptualized using the Axiomatic Design (AD) framework. The recommendations in the guidelines can be distinguished as generic design goals, generic design solutions, and generic design constraints. Designers are supposed to select and tailor these recommendations according to their design context.

Since coupling between design goals and design solutions is a source of complexity for designers, there is a need to understand and minimize interactions between usability guidelines. Nielsen's (1994b) usability heuristics was used as a case study to explore how AD's concepts could be adapted as a tool for this purpose. First, the recommendations in the heuristics were distinguished as design goals, design solutions, and design constraints. Next, the interactions between the goals and solutions were represented by a design equation.

The results show that the usability heuristics constitute a highly coupled system. A reengineering process, which used design equations as a mechanism for identifying the locus of coupling, was performed. By decomposing, integrating, and reformulating some of the design goals and design solutions, interactions between the heuristics were reduced. The usability of the modified heuristics for practicing designers needs to be determined.

CHAPTER 5. CONCLUSION AND RECOMMENDATION

5.1 Axiomatic Design for Analyzing Human-Machine Interaction

This research explores how Suh's (1990) Axiomatic Design (AD) can be adapted and developed into a method for analyzing user tasks in human-machine interaction. A framework, which consists of a goal domain, a functional domain, a physical domain, and an action domain, was constructed for modeling human-machine interaction.

Based on this framework and AD's design equation notation, a method for identifying coupling between the goals that a user wants to achieve and the actions that are designed into a user interface was developed. Coupling is a generic shaping factor of complexity, and coupling between user goals and user actions results in complexity for the user. This type of complexity widens the "gulf of execution".

The form of a design matrix provides a powerful visual summation of the coupling characteristic of a design. Several case studies were conducted to illustrate the application and to understand the generalizability of the proposed method.

The proposed method enables a design engineer to analyze the structure of a set of user tasks and identify any potential usability problems. In addition, the proposed framework makes it easy for a design engineer to identify the locus of complexity: coupling between the goal and functional domains is due to poor functional specification, coupling between the functional and physical domains is due to poor engineering design, and coupling between the physical and action domains is due to poor user interface design. Several case studies were used to illustrate each of these types of coupling.

Since the proposed method is systematic and explicit, it has the potential to facilitate rational discussions among members of a design team. Furthermore, by providing a general and rational criterion, the proposed method enables an early evaluation that is less dependent on the experience of individual design engineers.

In conclusion, a method for analyzing the complexity of human-machine interaction based on the principles of Axiomatic Design has been developed. This method is conceived as an additional tool that design engineers can use for identifying structural usability problems in the early stages of design.

5.2 Analyzing Interactions in Usability Guidelines

In the latter part of this research, a study was conducted to explore how AD can be adapted and developed into a method for analyzing coupling in software usability guidelines. The proposed method facilitates the identification and representation of interactions between guidelines by 1) distinguishing between design goals and design solutions and 2) by making use of the design equation notation. By providing an explicit notation, the design equations support discussions among members of a design team, and the design matrix provide a mechanism for reengineering a coupled set of guidelines.

Nielsen's (1994b) usability heuristics were used as a case study in the development of the proposed method. The heuristics were identified as highly coupled, and they were reformulated to eliminate some of the interactions.

5.3 Limitations and Recommendations

The limitations of this research and a list of corresponding recommendations for future work are stated as follows.

- To improve the generalizability of the proposed method, further case studies will be required. Future studies should investigate the effects of 1) increasing the size of the design problems, 2) varying the domain of the design problems, and 2) varying the type of user interfaces.
- Due to the use of binary values in the design matrix, the current notation identifies significant dependencies among various design attributes, but it does not differentiate between positive and negative interaction effects. It is reasonable to expect systems where, due to the design of the user interface, interactions between the functional and physical domains can be cancelled off by counter-acting interactions between the physical and action domains. For example, in process control theory, there are several techniques for decoupling a process control system (Shinskey, 1983). Further studies are required to propose an appropriate method for expressing this type of decoupling design.
- Coupling is a familiar concept in engineering. Based on the definition of complexity, the results of a previous experiment, and the case studies that have been presented in this thesis, it is argued that the detrimental effects of coupling in user control tasks are rather obvious. However, it is important to carry out further empirical studies to investigate how users perceive and deal with coupling in different environments.
- A redundant user-task design may have the potential of providing greater flexibility -- users can use different methods to achieve a single set of goals. However, redundancy also means that users have to make a selection decision. Hence, there will be a need to carry out further studies to investigate the usability effects of redundancy in interaction design.
- The usability of a method is important. Previous studies have proposed many usability methods, but usually only the easier methods are commonly used in industry. The proposed method is straightforward and coupling is a familiar

concept for engineers. Furthermore, it is based on the concepts of AD, which has been applied in a variety of fields. Nevertheless, it will be necessary to evaluate the learnability and the acceptability of the proposed method with practicing design engineers.

- Usability has a broad scope, and task procedures affect only certain aspects of usability (Kieras, 2004). There are many existing usability methods in the literature, and it will be necessary to investigate how the proposed method can complement other methods and fit into a system/product design and development process.
- The proposed method for analyzing usability guidelines is at a preliminary stage. Few studies have reported on the procedures that designers follow and the differences between novice and experienced designers in using guidelines. It will be important to investigate these issues so that the implications of coupling in guidelines can be better understood.
- In the usability heuristics case study, the formulation of the FR^gs and DP^gs, the identification of their interdependencies, and the reformulation of the FR^gs and DP^gs were largely based on subjective interpretations. Since these interpretations are expected to vary according to designers' subjective understanding and the domain of a design problem, it may be worthwhile to investigate these variations.

Usability is a broad concept. The proposed method enables a designer to study the task-procedural or structural aspects of usability: how user goals and user actions (or design goals and design solutions in the case of usability guidelines) are interrelated and whether there are any interactions that will result in complexity. There are aspects of usability that are not covered by the proposed method, such as the semantic of command names, the visual design of displays, etc. These aspects have to be studied through other means.

REFERENCES

- Ahmed, S., Wallance, K. M., and Blessing, L. T. M. (2003). Understanding the differences between how novice and experienced designers approach design tasks. *Research in Engineering Design*, 14(1), 1-11.
- Alexander, C. (1963). The determination of components for an Indian village. In *Proceedings of Conference on Design Methods* (pp. 83-114). Pergamon Press.
- Alexander, C. (1964). *Notes on the synthesis of form*. Cambridge, MA: Harvard University Press.
- Alexander, C., Sara, I., and Silverstein, M. (1977). *A pattern language: Towns, buildings, construction*. New York, NY: Oxford University Press.
- Alm, I. (2003). Designing interactive interfaces: Theoretical consideration of the complexity of standards and guidelines, and the difference between evolving and formalised systems. *Interacting with Computers*, 15, 109-119.
- Annett, J. (2004). Hierarchical task analysis. In D. Diaper, and N. Stanton (Eds.), *The Handbook of Task Analysis for Human-Computer Interaction* (pp. 67-82). Mahwah, NJ: Lawrence Erlbaum Associates.
- Annett, J., Duncan, K. D., Stammers, R. B., and Gray, M. J. (1971). *Task analysis*. Department of Employment Training Information Paper No. 6. London, UK: Her Majesty's Stationary Office.
- Apple (2002). Inside Mac OS X: Aqua human interface guidelines. Retrieved 26 May 2002, from <http://developer.apple.com/techpubs/macosx/Essentials/AquaHIGuidelines/index.html>
- Axiomatic Design Solutions (2004). *Acclaro Designer*. Retrieved 26 June 2004, from <http://www.axiomaticdesign.com/products/designer.asp>
- Baber, C., and Stanton, N.A. (1994). Task analysis for error identification. *Ergonomics*, 37(11), 1923-1942.

- Baber, C., and Stanton, N. A. (2004). Task analysis for error identification. In D. Diaper, and N. A. Stanton (Eds.), *The Handbook of Task Analysis for Human-Computer Interaction*. Mahwah, NJ: Lawrence Erlbaum Associates.
- Bastien, J. M. C., and Scapin, D. L. (1993). *Ergonomic criteria for the evaluation of human-computer interfaces (Report No. 156)*. Rocquencourt, France: Institut National de Recherche en Informatique et en Automatique.
- Beyer, H., and Holtzblatt, K. (1998). *Contextual design: Defining customer-centered systems*. San Francisco, CA: Morgan Kaufmann Publishers.
- Caldenfors, D. (1998). *Top-down reasoning in design synthesis and evaluation*. Licentiate Thesis. Linköping, Sweden: Graduate School of Human-Machine Interaction, Linköpings Universitet.
- Card, S. K., Moran, T. P., and Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *CACM*, 23(7), 396-410.
- Carter, J. (1999). Incorporating standards and guidelines in an approach that balances usability concerns for developers and end users. *Interacting with Computers*, 12, 179-206
- Clausing, D. (1994). *Total quality development: A step-by-step guide to world-class concurrent engineering*. New York, NY: ASME Press.
- Cognitive Engineering Laboratory at the University of Toronto (2004). *Research platform – DURESS II*. Retrieved 3 Jan 2004, from <http://www.mie.utoronto.ca/labs/cel/research/platform/>
- Cross, N. (Ed.) (1984). *Developments in design methodology*. Chichester, UK: John Wiley & Sons.
- Cross, N. (1994). *Engineering design methods: Strategies for product design (2nd ed.)*. Chichester, UK: John Wiley & Sons.
- de Souza, F., and Bevan, N. (1990). The use of guidelines in menu interface design: Evaluation of a draft standard. In *Proceedings of the IFIP TC13 Third International Conference on Human-Computer Interaction* (pp. 435-440).
- Dix, A. J. (2003) Upside down As and algorithms - computational formalisms and theory In J. Carroll (Ed.), *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science* (pp. 381-429). New York, NY: Morgan Kaufman.

- Dix, A. J., Finlay, G., Abowd, and R. Beale (1998). *Human-computer interaction (2nd ed.)*. Singapore: Prentice Hall.
- Döner, D. (1996). *The logic of failure: Why things go wrong and what we can do to make them right*. New York: Henry Holt. (Original work published 1989)
- El-Haik, B., and Tate, D. (Eds.) (2002). *Proceedings of the Second International Conference on Axiomatic Design - ICAD2002*. Cambridge, MA: Institute for Axiomatic Design.
- Endsley, M. R., Bolte, B., and Jones, D. G. (2002). *Designing for situation awareness: An approach to user-centered design*. London, UK: Taylor & Francis.
- Frey, D. D., and Hirschi, N. W. (2002). The effects of coupling on parameter design. In B. El-Hail, and D. Tate (Eds.), *Proceedings of the Second International Conference on Axiomatic Design - ICAD2002*. Cambridge, MA: Institute for Axiomatic Design.
- Gallagher, C. C., and Knight, W. A. (1986). *Group technology production methods in manufacture*. Chichester, UK: Halsted Press.
- Goel, V., and Pirolli, P. (1992). The structure of design problem spaces. *Cognitive Science*, 16, 395-429.
- Gould, J. D., Stephen J. B., and Clayton L. (1991). Making usable, useful, productivity: Enhancing computer applications. *Communications of the ACM* (January 1991), 72-86.
- Grammenos, D., Akoumianakis, D., and Stephanidis, C. (2000). Integrated support for working with guidelines: The Sherlock guideline management system. *Interacting with Computers*, 12, 281-311.
- Grant, S., and Mayes, T. (1991). Cognitive Task Analysis? In G. R. S. Weir, and J. L. Alty (Eds.), *Human-Computer Interaction and Complex Systems* (pp. 147-168). CA: Academic Press.
- Gray, W. D., John, B. E., and Atwood, M. E. (1993). Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world performance. *Human-Computer Interaction*, 8(3), 237-309.

- Guenov, M. D. (2002). Complexity and cost effectiveness measures for systems design. In G. Frizelle, and H. Richards (Eds.), *Proceedings of 2nd International Conference of the Manufacturing Complexity Network* (pp. 455-466). Cambridge, UK: Institute of Manufacturing.
- Hajdukiewicz, J. R., and Vicente, K. J. (2004). A theoretical note on the relationship between work domain analysis and task analysis. *Theoretical Issues in Ergonomics Science*, 5(6), 527-538.
- Halford, G., Wilson, W., Philips, S. (1997). Abstraction: Nature, costs and benefits. *International Journal of Educational Research*, 21-35.
- Han, S., Yun, M., Kim, K., and Kwahk, J. (2000). Evaluation of product usability: Development and validation of usability dimensions and design elements based on empirical models. *International Journal of Industrial Ergonomics*, 26, 477-488.
- Harrison, M. D., and Thimbleby, H. W. (Eds.) (1990). *Formal methods in human-computer interaction*. New York, NY: Cambridge University Press.
- Helander, M. G. (2005). The discipline of human factors engineering and ergonomics. In I. Noy, and W. Karwowski (Eds.), *Handbook of Forensic Human Factors* (Chapter 1). Boca Raton, FL: CRC Press.
- Helander, M. G., and Jiao, J. (2002). Coupling in design of human computer interaction. Implications for usability. In B. El-Haik, and D. Tate (Eds.), *Proceedings of the Second International Conference on Axiomatic Design - ICAD2002*. Cambridge, MA: Institute for Axiomatic Design.
- Helander, M. G., and Lin, L. (2000). Anthropometric design of workstations. In D. Tate (Ed.), *Proceedings of the First International Conference on Axiomatic Design – ICAD2000* (pp. 130-138). Cambridge, MA: Institute for Axiomatic Design.
- Helander, M. G., and Lin, L. (2002). Axiomatic design in ergonomics and an extension of the information axiom. *Journal of Engineering Design*, 13(4), 321-339.
- Helander, M. G., and Lo, S. (2003). Developing a method for formal top-down analysis of usability. In H. M. Khalid, T. W. Lim, and N. K. Lee (Eds.), *Proceedings of SEAMEC 2003*. Kuching, Malaysia: UNIMAS.
- Henninger, S. (2001). An organizational learning method for applying usability guidelines and patterns. In *8th IFIP Working Conference on Engineering for Human-Computer Interaction*. Toronto, Canada.

- Hirschi, N. W., and Frey, D. D. (2002). Cognition and complexity: An experiment on the effect of coupling in parameter design. *Research in Engineering Design*, 13, 123-131.
- Hoare, C. A. R. (1978). Communicating sequential processes. *Communications of the ACM*, 21(8), 666-677.
- Howie, D. E., and Vicente, K. J. (1998). Measures of operator performance in complex, dynamic microworlds: Advancing the state of the art. *Ergonomics*, 41, 485-500.
- IBM (2003). *Design basics*. Retrieved 27 May 2003, from http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/6
- ISO 9241-11 (1998). *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability*. Geneva, Switzerland: International Organization for Standardization.
- Jensen, S. (2003). *The simplicity shift: Innovative design tactics in a corporate world*. Singapore: Cambridge University Press.
- John, B. E. (2003). Information processing and skilled behavior. In J. M. Carroll (Ed.), *HCI Models, Theories, and Frameworks: Towards a Multidisciplinary Science*. Boston, MA: Morgan Kaufmann Publishers.
- Jones, C. B. (1980). *Software development: A rigorous approach*. Upper Saddle River, NJ: Prentice Hall PRT.
- Jones, J. C. (1992). *Design methods: Seeds of human futures (2nd ed.)*. New York, NY: Wiley.
- Kang, H. G., and Seong, P. H. (1998). An information theory-based approach for quantitative evaluation of user interface complexity. *IEEE Transactions on Nuclear Science*, 3165-3174.
- Karwowski, W. (1991). Complexity, fuzziness and ergonomics incompatibility issues in the control of dynamic work environments. *Ergonomics*, 34, 671-686.
- Karwowski, W. (1997). Ancient wisdom and future technology: The old tradition and the new science of human factors/ergonomics. In *Proceedings of the Human Factors and Ergonomics Society 41st Annual Meeting* (pp. 875-977). Santa Monica, CA: Human Factors and Ergonomics Society.

- Karwowski, W. (2003). Achieving compatibility in human-computer interface design and evaluation. In J. A. Jacko, and A. Sears (Eds.), *The Human-Computer Interaction Handbook* (pp. 1226-1238). Mahwah, NJ: Lawrence Erlbaum Associates.
- Kieras, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In M. G. Helander (Ed.), *Handbook of Human-Computer Interaction* (pp. 135-157). Amsterdam, The Netherlands: North-Holland.
- Kieras, D. E. (1997). Task analysis and the design of functionality. In A. B. Tucker (Ed.), *The Computer Science and Engineering Handbook* (pp. 1401-1423). CRC Press.
- Kieras, D. E. (2004). GOMS models for task analysis. In D. Diaper, and N. A. Stanton (Eds.), *The Handbook of Task Analysis for Human-Computer Interaction* (pp. 83-116). Mahwah, NJ: Lawrence Erlbaum Associates.
- Kieras, D. E., and Meyer, D. E. (2000). The role of cognitive task analysis in the application of predictive models of human performance. In J. M. C. Schraagen, S. E. Chipman, and V. L. Shalin (Eds.), *Cognitive Task Analysis*. Mahwah, NJ: Lawrence Erlbaum.
- Kieras, D. E., and Polson, P.G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies*, 22, 365-394.
- Kieras, D. E., and Polson, P. G. (1999). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies*, 51(2), 405-434.
- King, J. R. (1980). Machine-component grouping in production flow analysis: An approach using a rank order clustering algorithm. *International Journal of Production Research*, 18(2), 213-232.
- Lee, D., and Yoon, W. C. (2004). Coupling structural and functional models for interaction design. *Interacting with Computers*, 16, 144-161.
- Mandel, T. (1998). *The elements of user interface design*. New York, NY: John Wiley & Sons.
- Mayhew, D. J. (1992). *Principles and guidelines in software user interface design*. Englewood Cliffs, NJ: Prentice Hall.
- Mayhew, D. J. (1999). *The usability engineering lifecycle: A practitioner's handbook for user interface design*. San Francisco, CA: Morgan Kaufmann Publishers.

- McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on Software Engineering*, SE-2, 308-320.
- Microsoft (2003). *Design principles and methodology*. Retrieved 1st June 2003, from <http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnwue/html/ch02b.asp>
- Miller, C. (2000). The human factor in complexity. In J. Weyrauch, and T. Samad (Eds.), *Automation, Control and Complexity: An Integrated Approach* (pp. 35-57). Singapore: John Wiley & Sons.
- Miller, C. A., and Vicente, K. J. (2001). Comparison of display requirements generated via hierarchical task and abstraction-decomposition space analysis techniques. *International Journal of Cognitive Ergonomics*, 5(3), 335-355.
- MIT and UIUC DSM Research Teams (2003). *The Design Structure Matrix Web Site*. Retrieved 26 April 2004, from <http://www.dsmweb.org/>
- Mosier, J. N., and Smith, S. L. (1986). Application of guidelines for designing user interface software. *Behaviour and Information Technology*, 5, 39-46.
- National Academy of Sciences (2002). *Approaches to improve engineering design*. Retrieved 2 June 2003, from <http://books.nap.edu/books/NI000469/html/R1.html>
- Nielsen, J. (1993). *Usability engineering*. New York, NY: Morgan Kaufmann.
- Nielsen, J. (1994a). Enhancing the explanatory power of usability heuristics. In *Proceedings of the Association of Computerized Machinery CHI'94 Conference* (pp. 152-158). New York, NY: ACM Press.
- Nielsen, J. (1994b). Heuristic evaluation. In J. Nielsen, and R. L. Mack (Eds.), *Usability Inspection Methods*. New York, NY: John Wiley & Sons.
- Nielsen, J. (2004). *Heuristics for user interface design*. Useit.com. Retrieved 26 February 2003, from http://www.useit.com/papers/heuristic/heuristic_list.html
- Norman, D. A. (1988). *The psychology of everyday things*. New York, NY: BasicBooks.
- Ossher, H. L. (1987). A mechanism for specifying the structure of large, layered systems. In B. Shriver, and P. Wegner (Eds.), *Research Directions in Object-Oriented Programming*. Cambridge, MA: MIT Press.

- Payne, S. J., and Green, T.R.G. (1989). The structure of command languages: An experiment on task-action grammar. *International Journal of Man-machine Studies*, 30, 213-234.
- Perrow, C. (1984). *Normal accidents: Living with high-risk technologies*. New York, NY: Basic Books.
- Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T. (1994). *Human-computer interaction*. Singapore: Addison-Wesley.
- Pribeanu, C., and Vanderdonckt, J. (2002). Exploring design heuristics for user interface derivation from task and domain models. In *Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces CADUI'2002* (pp. 103-110). Dordrecht, Netherlands: Kluwer Academics Publishers.
- Quesada, J., Kintsch, W., Gomez, E. (2005). Complex problem-solving: A field in search of a definition? *Theoretical Issues in Ergonomics Science*, 6(1), 5-33.
- Quill, L., Kancler, D. E., Revels, A. R, and Batchelor, C. (2001). Application of information visualization at various stages of system development. In *Proceedings of the Human Factors and Ergonomics Society 45th Annual Meeting* (pp. 1713-1717). Santa Monica, CA: Human Factors Society.
- Rasmussen, J., Pejtersen, A. M., and Goodstein, L. P. (1994). *Cognitive systems engineering*. New York, NY: Wiley.
- Rauterberg, M. (1992). A method of a quantitative measurement of cognitive complexity. In G. C. van der Veer, M. J. Tauber, S. Bagnara, and A. Antalovits (Eds.), *Human-Computer Interaction: Tasks and Organisation* (pp. 295-307). Roma, Italy: CUD.
- Rauterberg, M. (1996). How to measure cognitive complexity in human-computer interaction. In R. Trappl (Ed.), *Cybernetics and Systems '96*, 2, 815-820. Vienna: Austrian Society for Cybernetic Studies.
- Reed, P., Holdaway, K., Isensee, S., Buie, E., Fox, J., Williams, J., and Lund, A. (1999). User interface guidelines and standards: Progress issues, and prospects. *Interacting with Computers*, 12, 119-142.
- Sadun, L. (2001). *Applied linear algebra: The decoupling principle*. New Jersey, NJ: Prentice Hall.

- Schiele, F., and Green, T. R. G. (1990). Formalisms in HCI: The case of task-action grammar. In M. Harrison, and H. Thimbleby (Eds.), *Formal Methods in Human-Computer Interaction*. New York, NY: Cambridge University Press.
- Shackel, B. (1991). Usability – context, framework, design and evaluation. In B. Shackel, and S. Richardson (Eds.), *Human Factors for Informatics Usability* (pp. 21-38). Cambridge: Cambridge University Press.
- Shinskey, F. G. (1983). *Controlling multivariable processes*. An independent learning module from the Instrument Society of America. Foxboro, MA: Instrument Society of America.
- Shneiderman, B. (1987). *Designing the user interface: Strategies for effective human-computer interaction*. Reading, MA: Addison-Wesley.
- Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: MIT Press.
- Slade, S. (1994). *Goal-based decision making: An interpersonal model*. Hillsdale, N.J.: Lawrence Erlbaum Associates.
- Smith, S. L. (1986). Standards versus guidelines for designing user interface software. *Behaviour and Information Technology*, 5(1), 74-61.
- Smith, S. L., and Mosier, J. N. (1986). *Guidelines for designing user interface software*. Report ESD-TR-86-278. Bedford, MA: The Mitre Crop.
- Spivey, J. M. (1988). *Understanding Z: A specification language and its formal semantics*. New York, NY: Cambridge University Press.
- Stanton, N. A., and Young, M. S. (1998). Is utility in the mind of the beholder? A study of ergonomics methods. *Applied Ergonomics*, 29(1), 41-54.
- Stanton, N. A., and Young, M. S. (1999). *A guide to methodology in ergonomics: Designing for human use*. New York, NY: Taylor & Francis.
- Stanton, N. A., and Young, M. S. (2003). Giving ergonomics away? The application of ergonomics methods by novices. *Applied Ergonomics*, 34, 470-490.
- Steward, D. V. (1981). *Systems analysis and management: Structure, strategy, and design*. New York, NY: Petrocelli Books.

- Strong, G. W. (1994). *New directions in human-computer interaction education, research and practice (report sponsored by NSF and ARPA)*. Retrieved 7 July 2003, from <http://www.sei.cmu.edu/community/hci/directions/>
- Su, C.-Y., Chen, S.-J., and Lin, L.. (2003). A structured approach to measuring functional dependency and sequencing of coupled tasks in engineering design. *Computers and Industrial Engineering*, 45(1), 195-214.
- Suh, N. P. (1990). *The principles of design*. New York, NY: Oxford University Press.
- Suh, N. P. (2001). *Axiomatic design: Advances and applications*. New York, NY: Oxford University Press.
- Suh, N. P., (2005). *Complexity: Theory and applications*. New York, NY: Oxford University Press.
- Tate, D. (Ed.) (2000). *Proceedings of the First International Conference on Axiomatic Design – ICAD2000*. Cambridge, MA: Institute for Axiomatic Design.
- Tetzlaff, L., and Schwartz, D. R. (1990). The use of guidelines in interface design. In S. P. Robertson, G. M. Olson, J. S. Olson, J. S. (Eds.), *Proceedings of the ACM CHI'91* (pp. 329-333). New York, NY: ACM Press.
- Theo, M. (1997). *The elements of user interface design*. New York, NY: Wiley Computer Pub.
- Thimbleby, H. W. (1990). *User interface design*. Singapore: Addison-Wesley.
- Thimbleby, H. W. (2004). User interface design with matrix algebra. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 11(2), 181-236.
- Timpf, S. (1999). Abstraction, levels of detail, and hierarchies in map series. In C. Freksa, and D. M. Mark (Eds.), *International Conference Cosit '99* (pp. 125-140). Stade, Germany: Springer-Verlag Telos.
- van Welie, M. van der Veer, G. C., and Eliëns, A. (1998). An ontology for task world models. In *5th International Eurographics Workshop on Design Specification and Verification of Interactive Systems DSV-IS98* (pp. 57-70). Abingdon, UK.
- van Welie, M. van der Veer, G. C., and Eliëns, A. (1999). Breaking down usability. In *Proceedings of Interact 99* (pp. 613-620). Edinburgh, Scotland.

- van Welie, M. van der Veer, G. C., and Eliëns, A (2000). Patterns as tools for user interface design. In *International Workshop on Tools for Working with Guidelines* (pp. 313-324). Biarritz, France.
- Vanderdonckt, J. (1995). SIERRA: An interactive system for ergonomic realization of applications. *SIGCHI Bulletin*, 27(2), 50-51.
- Vanderdonckt, J. (1999). Development milestones towards a tool for working with guidelines. *Interacting with Computers*, 12, 81-118.
- Vicente, K. J. (1999). *Cognitive work analysis: Toward safe, productive, and healthy computer-based work*. Mahwah, NJ: Lawrence Erlbaum Associates.
- Weir, G. R. S. (1991). Living with complex interactive systems. In G. R. S. Weir, and J. L. Alty (Eds.), *Human-Computer Interaction and Complex Systems* (pp. 1-21). New York, NY: Academic Press.
- Weng, G., Bhalla, U. S., and Lyengar, R. (1999). Complexity in biological signaling systems. *Science*, 284, 92 – 96.
- Whitefield, A., Wilson, F., and Dowell, J. (1991). A framework for human factors evaluation. *Behaviour and Information Technology*, 10, 65-79.
- Wickens, C. D., and Hollands J. G. (2000). *Engineering psychology and human performance*. New York, N.Y.: HarperCollins Publishers.
- Wilensky, R. (1983). *Planning and understanding: A computational approach to human reasoning*. Reading, Massachusetts: Addison-Wesley.
- Wing, J. M. (1990). A specifier's introduction to formal methods. *IEEE Computer*, 23(9), 8-24.
- Woods, D. D. (1988). Coping with complexity: The psychology of human behaviour in complex systems. In L. P. Goodstein, H. B. Andersen, and S. E. Olsen (Eds.), *Tasks, Errors and Mental Models: A Festschrift to Celebrate the 60th Birthday of Professor Jens Rasmussen* (pp. 129-147). New York, NY: Taylor & Francis.
- Wu, J. (1999). Hierarchy and scaling: Extrapolating information along a scaling ladder. *Canadian Journal of Remote Sensing*, 25(4), 367-380.
- Yates, F. E. (1978). Complexity and the limits to knowledge. *American Journal of Physiology*, 4, R201-R204.

APPENDIX 1. AXIOMATIC DESIGN

A1.1 Fundamentals of Axiomatic Design

Suh (1990) introduced a design framework and methodology that is known as Axiomatic Design (AD). AD is a formal method that can be used to represent a variety of design problems, such as mechanism design, software design, and organizational design (National Academy of Sciences, 2002).

AD conceptualizes design activity as “a continuous interplay between *what we want to achieve* and *how we want to achieve it*” (Suh, 1990, p. 25). There are five fundamental concepts in AD:

- 1) the concept of design domains,
- 2) the concept of design hierarchy and abstraction levels,
- 3) the concept of zigzagging between design domains,
- 4) the Independence Axiom, and
- 5) the Information Axiom.

A basic introduction to these concepts is provided below.

The AD framework consists of four design domains:

- 1) the customer domain,
- 2) the functional domain,
- 3) the physical domain, and
- 4) the process domain.

For each pair of adjacent design domains in Figure A1.1, the one on the left represents “*what we want to achieve*”, while the one on the right represents the design solution of “*how we propose to achieve it*”.

Figure A1.1. Axiomatic Design's framework consists of four design domains (adapted from Suh, 2001)

Design begins by identifying the customer needs (CNs) and specifying these needs as functional requirements (FRs). FRs refer to a set of mutually independent functional needs that characterize a design problem. The designer conceives a physical embodiment and identifies the design parameters (DPs) that satisfy the FRs. The DPs are mapped onto a set of process variables (PVs) for implementation. PVs could be, for example, the key parameters of the manufacturing processes that are used to produce the DPs.

Design descriptions vary in term of abstraction level. At each level of description, there is a set of FRs. Before these FRs can be decomposed into more detailed FRs, the designer needs to select a set of DPs to satisfy these FRs. *Zigzagging* refers to this process of alternating between pairs of design domains during design decomposition (see Figure A1.2).

Figure A1.2. The concept of zigzagging states that higher level FRs should not be decomposed into lower level FRs before the higher level DPs have been conceptualized (adapted from Suh, 2001).

The zigzagging concept is based on the fact that decisions that are made at a higher level of abstraction constraint the decisions that are made at a lower level of abstraction. For example, if a team of engineers are to design a system that “transports passengers from country A to country B” (this is the top-level FR), the engineers will need to decide which type of transportation system it should be (this is the top-level DP) before they can specify what are the lower-level FRs. The lower-level FRs of an “air transportation system” and the lower-level FRs of a “sea transportation system” would be very different. This decomposition process should proceed layer by layer until the leaf elements are concrete and detailed enough for implementation.

Based on his experience and observation, Suh (1990) proposed two design axioms that are known as the Independence Axiom and the Information Axiom. The Independence Axiom states that a good design solution is one that maintains the mutual independence of the FRs of the design problem. This is achieved by selecting a correct set of DPs. The Information Axiom states that, among the design solutions that satisfy the Independence

Axiom, the best solution is the one that has the lowest information content. The information content of a design is calculated according to the following formula:

$$I = \log \frac{1}{p} \quad (\text{A1.1})$$

Most of the existing applications of the Information Axiom are related to manufacturing tolerance. To understand how this axiom can be applied to other aspects of design, the definition of information content needs to be further developed (National Academy of Sciences, 2002). The method that has been developed in this research does not involve the use of the Information Axiom. However, the author conceives that the Information Axiom may be useful for advancing the method in future work.

The mappings between design domains are expressed mathematically in the form of matrix equations (Suh, 1990). At a given level of a design hierarchy, the FRs constitute a FR vector in the functional domain and the DPs constitute a DP vector in the physical domain. The relationship between these two vectors can be written as

$$\{FR\} = [A]\{DP\} \quad (\text{A1.2})$$

where $[A]$ is called the *design matrix*. Design matrix $[A]$ is of the following form:

$$[A] = \begin{bmatrix} A_{11} & A_{12} & \cdot & A_{1n} \\ A_{21} & A_{22} & \cdot & A_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ A_{m1} & A_{m2} & \cdot & A_{mn} \end{bmatrix} \quad (\text{A1.3})$$

Each element A_{ij} of a design matrix relates a component of the **FR** vector to a component of the **DP** vector. Typically, a design matrix is expressed qualitatively as shown in Equation A1.4. A ‘I’ or an ‘X’ indicates that the column’s DP affects the

row's FR, while an 'O' or a blank indicates that it does not. In Equation A1.4, DP_1 affects all three FRs, while DP_2 affects only FR_2 , and DP_3 affects only FR_3 .

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{bmatrix} X & & \\ X & X & \\ X & & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix} \quad (\text{A1.4})$$

Designs that satisfy the Independence Axiom have either a diagonal design matrix or a triangular design matrix, and they are respectively known as *uncoupled designs* or *decoupled designs*. Designs that have neither a diagonal nor a triangular design matrix do not satisfy the Independence Axiom, and they are known as *coupled designs*. Therefore, the shape of a design matrix can be used as a qualitative metric for characterizing the degree of coupling in a design.

A1.2 Advance Issues

Since the publication of Suh (1990), researchers from different fields have reported on both successful applications of AD and difficulties in applying AD. This chapter presents a discussion on several issues in Suh's (1990; 2001) Axiomatic Design (AD). This discussion does not relate directly to the final products of the research; rather, it is a by-product of the research. It was part of the process in developing a better understanding of AD for the purpose of this research. The following two issues will be discussed in this chapter:

- 1) The type of hierarchical structures in the AD framework.
- 2) Imaginary coupling as a result of failing to distinguish abstraction levels.

A1.2.1 Hierarchical Structures in Axiomatic Design

Abstraction levels are typically represented in the form of a hierarchy, which is defined as an ordered structure (Timpf, 1999). Different types of hierarchical structures have been used frequently to model complex systems (Vicente, 1999). Cognitive systems engineering typically uses a combination of two hierarchical structures to representation work domains: a decomposition hierarchy and an abstraction hierarchy (Rasmussen et al., 1994) (see Figure A1.3).

Figure A1.3. Cognitive systems engineering typically uses a combination of a whole-part hierarchy and a means-ends hierarchy to represent work domains (from Rasmussen et al., 1994).

The decomposition hierarchy specifically refers to a structure that has a *part-whole* relationship between levels, and an abstraction hierarchy specifically refers to a structure that has a *means-ends* relationship between levels (Vicente, 1999). The means-ends hierarchy displays a strong triadic characteristic; when one studies a phenomenon at a particular abstraction level, the mechanistic understanding comes from the next lower level, whereas the significance of that phenomenon can only be revealed at the next higher level (Wu, 1999). Therefore, each three adjacent levels in the abstraction hierarchy can be described as having a “*why-what-how*” relationship (see Figure A1.4). This characteristic of the means-ends hierarchy has been reported to support *goal-directed problem solving* (Vicente, 1999).

Figure A1.4. The triadic characteristic of a means-ends hierarchy.

In AD, between each pair of adjacent design domains, there exists a “*what-how*” relationship (Suh, 1990). Since this is a means-ends hierarchy, based on the above discussion, one can also say that between each triplet of interlinked design domains there is a “*why-what-how*” relationship (see Figure A1.5).

Figure A1.5. Each triplet of adjacent design domains has a “why-what-how” form of means-ends relationship.

Suh (1990) stated that two design hierarchies are formed by the zigzagging mapping process between the functional and physical domains, and according to his decomposition of a lathe system, both the functional and physical hierarchies are part-whole hierarchies (see Figures A1.6 and A1.7).

Figure A1.6. The functional hierarchy of a lathe system (from Suh, 1990).

Figure A1.7. The physical hierarchy of a lathe system (from Suh, 1990).

However, a review of other AD examples shows that the functional hierarchy is usually a means-ends hierarchy, whereas the nature of the physical hierarchy is inconsistent (El-Haik and Tate, 2002; Suh, 1990; Suh, 2001; Tate, 2000). For example, Table A1.1 shows the decomposition of a Newcomen steam engine, and both the functional and physical hierarchies are means-ends hierarchies (Suh, 2001). On the other hand, Table A1.2 shows the decomposition of a refrigerator, and the functional hierarchy is a means-ends hierarchy while the physical hierarchy is a means-ends hierarchy (Suh, 2001).

Table A1.1 Decomposition of a Newcomen Steam Engine (Suh, 2001).

Functional Domain	Physical Domain
FR_1 = Extend the piston	DP_1 = Pressure of the steam
FR_2 = Contract the piston by creating a vacuum in the cylinder	DP_2 = Vacuum in the cylinder/piston by condensation of the steam
$FR_{1,1}$ = Generate the steam	$DP_{1,1}$ = Boiler
$FR_{1,2}$ = Inject the steam	$DP_{1,2}$ = Valve
$FR_{1,3}$ = Expand the steam and move the piston outward	$DP_{1,3}$ = Steam
$FR_{2,1}$ = Condense the steam	$DP_{2,1}$ = Cold water spray
$FR_{2,2}$ = Move the piston inward	$DP_{2,2}$ = Pressure difference caused by condensation
$FR_{2,3}$ = Discharge the condensate	$DP_{2,3}$ = Discharge valve

Table A1.2 Decomposition of a refrigerator (Suh, 2001).

Functional Domain	Physical Domain
FR_1 = Freeze food for long-term preservation	DP_1 = The freezer section
FR_2 = Maintain food at cold temperature for short-term preservation	DP_2 = The chiller (i.e., refrigerator) section
$FR_{1,1}$ = Control the temperature of the freezer section in the range of $-18^\circ\text{C} \pm 2^\circ\text{C}$	$DP_{1,1}$ = Sensor/compressor system that turns the compressor on (off) when the air temperature is higher (lower) than the set temperature in the freezer section
$FR_{1,2}$ = Maintain a uniform temperature throughout the freezer section at the preset temperature	$DP_{1,2}$ = Air circulation system that blows air into the freezer section and circulates it uniformly throughout the freezer section at all times
$FR_{1,3}$ = Control humidity of the freezer section to relative humidity of 50%	$DP_{1,3}$ = Condenser that condenses the moisture in the returned air when its dew point is exceeded
$FR_{2,1}$ = Control the temperature of the chiller section in the range of 2°C to 3°C	$DP_{2,1}$ = Sensor/compressor system that turns the compressor on (off) when the air temperature is higher (lower) than the set temperature in the chiller section
$FR_{2,2}$ = Maintain a uniform temperature throughout the chiller section within 0.5°C of the preset temperature	$DP_{2,2}$ = Air circulation system that blows air into the chiller section and circulates it uniformly throughout the chiller section at all times

In conclusion, AD framework utilizes two hierarchies, as illustrated in Figure A1.8. A horizontal means-ends hierarchy can be used to represent the multi-facet nature of

design problems. There are also vertical hierarchies within the design domains, which support problem formulation and solution conceptualization. These hierarchies support goal-based decision-making (Slade, 1994), but their structures may not strictly follow a particular type.

Figure A1.8. The Axiomatic Design framework consists of a horizontal hierarchy and a vertical hierarchy.

A1.2.2 Distinguishing Abstraction Levels

Failing to distinguish abstraction levels in analysis increases the perceived complexity of a design problem. A microscope workstation design problem can be used to illustrate this phenomenon (Helander and Lo, 2003):

Let the top-level FRs and DPs of a microscope workstation be

FR_1 = Adjustability of workstation.

DP_1 = Adjustability mechanisms.

This design can be decomposed as follows:

$FR_{1.1}$ = Footrest adjustability.

$FR_{1.2}$ = Work table height adjustability.

$FR_{1.3}$ = Microscope table adjustability.

$DP_{1.1}$ = Adjustable legs of footrest.

$DP_{1.2}$ = Adjustable legs of worktable.

$DP_{1.3}$ = Adjustable legs of microscope table.

If this design is analyzed at the first level, then the design equation will show an uncoupled design:

$$\{FR_1\} = [X]\{DP_1\} \quad (A1.5)$$

If this design is analyzed at the second abstraction level, then the design equation will also show an uncoupled design:

$$\begin{Bmatrix} FR_{1.1} \\ FR_{1.2} \\ FR_{1.3} \end{Bmatrix} = \begin{bmatrix} X & & \\ & X & \\ & & X \end{bmatrix} \begin{Bmatrix} DP_{1.1} \\ DP_{1.2} \\ DP_{1.3} \end{Bmatrix} \quad (A1.6)$$

However, if the first and second abstraction levels are not distinguished, the design equation will shown a coupled design:

$$\begin{Bmatrix} FR_1 \\ FR_{1.1} \\ FR_{1.2} \\ FR_{1.3} \end{Bmatrix} = \begin{bmatrix} X & X & X & X \\ X & X & & \\ X & & X & \\ X & & & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_{1.1} \\ DP_{1.2} \\ DP_{1.3} \end{Bmatrix} \quad (A1.7)$$

Therefore, it is important to be consistent in terms of abstraction level in analysis; otherwise, the design may appear to be more complex than it really is.

APPENDIX 2. MATRIX REORDERING ALGORITHMS

This chapter reproduces two algorithms that can be used to reorder a matrix. The first algorithm was presented by Steward (1981). The second algorithm was presented by King (1979). A comparison between these two algorithms was presented in Section 3.2.1.

A2.1 Steward's Algorithm

Suh (1990, p. 383) presented an algorithm for reordering a full design matrix into, if possible, a triangular matrix. This algorithm appears to be based on Steward's (1981) algorithm, which was used for reordering a Design Structure Matrix (DSM). Suh's version is reproduced below. Equation A2.1 shows an exemplary design matrix to aid explanation.

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \\ FR_4 \end{Bmatrix} = \begin{bmatrix} A_{11} & A_{12} & 0 & A_{14} \\ 0 & A_{22} & A_{23} & 0 \\ 0 & 0 & A_{33} & 0 \\ 0 & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \\ DP_4 \end{Bmatrix} \quad (\text{A2.1})$$

- 1) Find the row which contains one non-zero element. Rearrange the order of {FRs} and {DPs} by putting the row and the column that contains the non-zero element first (i.e., if the i^{th} row contains one non-zero element at j^{th} column, then put i^{th} component of {FR} and j^{th} component of {DP} first), see Equation A2.2.

$$\begin{Bmatrix} FR_3 \\ FR_1 \\ FR_2 \\ FR_4 \end{Bmatrix} = \begin{bmatrix} A_{33} & 0 & 0 & 0 \\ 0 & A_{11} & A_{12} & A_{14} \\ A_{23} & 0 & A_{22} & 0 \\ A_{43} & 0 & A_{42} & A_{44} \end{bmatrix} \begin{Bmatrix} DP_3 \\ DP_1 \\ DP_2 \\ DP_4 \end{Bmatrix} \quad (A2.2)$$

- 2) Excluding the first row and column, find the row which contains one non-zero element. Rearrange the components of {FRs} and {DPs} by putting the row and the column that contains the non-zero element at the row and column second (see Equation A2.3).

$$\begin{Bmatrix} FR_3 \\ FR_2 \\ FR_1 \\ FR_4 \end{Bmatrix} = \begin{bmatrix} A_{33} & 0 & 0 & 0 \\ A_{23} & A_{22} & 0 & 0 \\ 0 & A_{12} & A_{11} & A_{14} \\ A_{43} & A_{42} & 0 & A_{44} \end{bmatrix} \begin{Bmatrix} DP_3 \\ DP_2 \\ DP_1 \\ DP_4 \end{Bmatrix} \quad (A2.3)$$

- 3) Repeat the procedure until there are no more sub-matrices to analyze (see Equation A2.4).

$$\begin{Bmatrix} FR_3 \\ FR_2 \\ FR_4 \\ FR_1 \end{Bmatrix} = \begin{bmatrix} A_{33} & 0 & 0 & 0 \\ A_{23} & A_{22} & 0 & 0 \\ A_{43} & A_{42} & A_{44} & 0 \\ 0 & A_{12} & A_{14} & A_{11} \end{bmatrix} \begin{Bmatrix} DP_3 \\ DP_2 \\ DP_4 \\ DP_1 \end{Bmatrix} \quad (A2.4)$$

This algorithm can be carried out on a computer. Axiomatic Design Solutions (2004) provide an Acclaro Designer® software specifically for AD, while the MIT and UIUC DSM Research Teams (2003) provide a Microsoft Excel™ Macros for reordering DSMs. The software and the Macros appear to be based on similar algorithms. This research used the Macros, because it is publicly available.

A2.2 King's Algorithm

King (1980) presented a rank-order cluster algorithm to generate diagonalized groupings of matrix entries for production flow analysis. The algorithm is reproduced as follows:

- 1) For $\forall_j$, calculate the total weight of column w_j :

$$w_j = \sum_{\forall_i} 2^i M_{ij} \quad (\text{A2.5})$$

- 2) If w_j is in ascending order, go to Step 3. Otherwise, rearrange the columns to make w_j fall in an ascending order.
- 3) For $\forall_j$, calculate the total weight for row w_i :

$$w_i = \sum_{\forall_j} 2^j M_{ij} \quad (\text{A2.6})$$

- 4) If w_i is in ascending order, stop. Otherwise, rearrange the rows to make w_i fall in an ascending order and go to step 1.

The following figures illustrate the operation of the algorithm. Applying the algorithm to the matrix in Figure A2.1 clusters the non-zero entries into two groups at the diagonal in Figure A2.2.

		COMPONENTS					
		1	2	3	4	5	6
OPERATIONS	1	0	1	0	1	1	0
	2	1	0	1	0	0	1
	3	0	1	0	1	1	0
	4	0	0	1	0	0	1

Figure A2.1. A simple production flow analysis matrix (Gallagher and Knight, 1986).

		COMPONENTS					
		3	6	1	4	2	5
OPERATIONS	4	1	1	0	0	0	0
	2	1	1	1	0	0	0
	3	0	0	0	1	1	1
	1	0	0	0	1	1	1

Figure A2.2. The reorganized production flow analysis matrix (Gallagher et al., 1986).

APPENDIX 3. LIST OF VARIABLES IN DURESS II

Variable	Description
T0	Inlet water temperature
T1	Temperature of reservoir 1
T2	Temperature of reservoir 2
MO1	Mass output flow rate for reservoir 1
MO2	Mass output flow rate for reservoir 2
MI1	Mass input flow rate for reservoir 1
MI2	Mass input flow rate for reservoir 2
V1	Volume of reservoir 1
V2	Volume of reservoir 2
E1	Total energy stored in reservoir 1
E2	Total energy stored reservoir 2
EI1	Energy input flow rate for reservoir 1
EI2	Energy input flow rate for reservoir 2
EO1	Energy output flow rate for reservoir 1
EO2	Energy output flow rate for reservoir 2
FH1	Flow from heater HTR1
FH2	Flow from heater HTR2
FA1	Flow rate from valve VA1
FB1	Flow rate from valve VB1
FA2	Flow rate from valve VA2
FB2	Flow rate from valve VB2
FPA	Flow rate from pump PA
FPB	Flow rate from pump PB
FVA	Flow rate from valve VA
FVB	Flow rate from valve VB
HTR1	Setting for heater of reservoir 1
HTR2	Setting for heater of reservoir 2
PA	Setting of pump in feedwater stream A
PB	Setting of pump in feedwater stream B
VA	Setting of initial valve in feedwater stream A
VB	Setting of initial valve in feedwater stream B
VA1	Setting of valve 1 in feedwater stream A
VB1	Setting of valve 1 in feedwater stream B
VA2	Setting of valve 1 in feedwater stream A
VB2	Setting of valve 2 in feedwater stream B
VO1	Setting of output valve 1 in reservoir 1
VO2	Setting of output valve 2 in reservoir 2

(Adapted from Vicente, 1999)

APPENDIX 4. LIST OF ACRONYMS

Acronym	Full Name	Definition
Cs	Constraints.	Bounds on acceptable solutions.
CNs	Customer needs.	The needs of customers.
FRs	Functional requirements.	A set functional needs that characterizes a design problem.
FR ^g _s	Functional requirements that are recommended by usability guidelines.	A set of generic usability design goals that can be extracted from a set of usability guidelines.
FR ^u _s	Functional requirements that are related to the user goals.	A subset of the functional requirements of a human-machine system that is used to satisfy the user goals.
DPs	Design parameters.	A set of variables in the physical domain that is selected by a designer to satisfy the functional requirements.
DP ^g _s	Design parameters that are recommended by usability guidelines.	A set of generic design parameters that can be extracted from a set of usability guidelines and are means for satisfying usability design goals.
DP ^u _s	Design parameters that are related to the user actions.	A subset of the design parameters of a human-machine system that is controlled by the user actions.
PVs	Process variables.	A set of variables in the process domain that implements the design parameters of a system.
UAs	User actions.	A set of control actions that is designed into a user interface for a user to achieve his or her goals.
UGs	User goals.	A set of user goals that constitutes a user's desired state of a human-machine system, and it is deemed as a subset of customer needs.
UT	User task.	A user goal and its related user action(s).