A Risk Dynamics Model of Complex System Development

by JAY JOOTAR

B.E., Electrical Engineering
Chulalongkorn University, 1995

M.B.A., Financial Engineering
Massachusetts Institute of Technology, 1998

SUBMITTED TO THE SLOAN SCHOOL OF MANAGEMENT
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY in MANAGEMENT
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2002

©2002 Massachusetts Institute of Technology. All rights reserved.

Signature of Author..............................................................

Sloan School of Management
April 18, 2002

Certified by...........................................................................

Steven D. Eppinger
Professor of Management

Accepted by...........................................................................

Birger Wernerfelt
Chairman, Doctoral Program, Sloan School of Management
A Risk Dynamics Model of Complex System Development

JAY JOOTAR
jootar@alum.mit.edu

ABSTRACT

The development of complex systems is a challenging endeavor which has captured the attentions of scholars and practitioners alike. Throughout the decades, numerous methods have been proposed to help manage such development efforts more effectively and efficiently. Some of these methods, such as prototyping, concurrent engineering, iterative model for software development, and system-focused development for R&D, are process models which recommend better ways to structure the development process to handle the complexity of the system under development.

This thesis seeks to understand the complex system development from a risk perspective. Continuing from the work done by other researchers, this work combines issues which are traditionally considered separately into one single model. More specifically, the model explicitly captures the dependencies in the system and the structure of an iterative development process and their interactions.

The resulting mathematical problem demonstrates the risk characteristics of a development process. It shows that the optimality calls for a trade-off between the reduction of the probability of risk and the increase in the impact of risk. From its structure, the model also helps us understand how different aspects of system architecture affect the structure and the performance of the development process. In addition, the model also reveals the fundamental problems of process models and proposes a generic risk-based alternative.

To explore the applicability of the model, the thesis also provides a case study in a software development process and a set of heuristics for solving the resulting combinatorial problem.
Thesis Advisor:  Steven D. Eppinger, Sc.D.
Professor of Management
Sloan School of Management

Doctoral Committee:  Steven D. Eppinger, Sc.D.
Professor of Management
Sloan School of Management

Stephen C. Graves, Ph.D.
Professor of Management
Sloan School of Management

Andreas S. Schulz, Ph.D.
Associate Professor of Operations Research
Sloan School of Management
ACKNOWLEDGEMENT

I would like to thank my advisor Steven Eppinger for his guidance, patience, support, and friendship throughout my years in the doctoral program at Sloan. I have learned a lot from him and have a delightful experience working with him. I also would like to thank two members of my committee, Stephen Graves and Andreas Schulz, who provided me with helpful advices and guidance in several aspects of this work.

I would like to thank the Center for Innovation in Product Development (CIPD) at MIT for providing me with fundings throughout my doctoral study and for being the center of a community that assists me in different aspects. Kathleen Wang was very helpful in finding industrial sponsors for my research projects. Daniel Whitney was kind enough to read through my thesis and provided me with many useful insights and comments. Darian Unger helped me edit this thesis. I also benefited from exchanging idea with many researchers who are associated with CIPD either directly or indirectly.

I would like to thank George Kocur’s assistance in providing me with a contact at Verizon. I appreciate SuperPages.com backoffice team for helping me in the case study. Especially, I would to thank Lev Koyfman, Alex Sakharov, Paul Hoffmann, and Gang Lu who spend time with me in answering some important questions pertaining to the case study.

Personally, I feel greatly indebted to my family. Dad, Mom, Jittra, and Don have constantly provided me with love, support, and encouragement. I also would like to thank my friends within and outside of Boston who made these years such a delightful experience.

Spiritually, I would like to thank the late Buddhadasa Bhikkhu for his Dhamma teaching which has kept me sane throughout these years.
# Table of Content

**Chapter 1: Research Motivation** .......................................................... 11
  1.1 Introduction .................................................................................. 11
  1.2 Complex System Development ...................................................... 12
  1.3 Process Models ........................................................................... 16
  1.4 Risk Dynamics ............................................................................ 20
  1.5 Research Focus ........................................................................... 24
  1.6 Thesis Structure ......................................................................... 29

**Chapter 2: Literature Survey** .............................................................. 31
  2.1 Introduction ................................................................................ 31
  2.2 Context, Scope and Results .......................................................... 31
  2.3 Benefits and Limitations ............................................................... 35
  2.4 Problems with Information ........................................................... 36
  2.5 Next Direction ........................................................................... 38

**Chapter 3: Model Formulation** .......................................................... 41
  3.1 Introduction ................................................................................ 41
  3.2 The Premise ............................................................................... 41
  3.3 Risk Dynamics Model ................................................................ 44
  3.4 Probability and Impact Function .................................................. 53
  3.5 Risk Matrix ............................................................................... 59
  3.6 Work-Minimization Problem ......................................................... 60

**Chapter 4: Model Analysis** ................................................................. 67
  4.1 Introduction ............................................................................... 67
  4.2 Risk Efficiency ........................................................................... 67
  4.3 Risk Concentration ..................................................................... 77
  4.4 Product Architecture .................................................................. 91
  4.4 Interactions of Risks .................................................................. 101
  4.6 Process Models .......................................................................... 109
CHAPTER 1: RESEARCH MOTIVATION

1.1 Introduction

Complex systems have been a critical part of human civilizations since the antiquity. The Egyptians built canals and irrigation systems to support the agricultural economy. The Romans built aqueducts and complex road systems which sustain their vast empire. Since the advent of modern science and technology, human beings have built various types of complex systems for a variety of purposes. Airplanes, automobiles, telecommunication systems, computer systems, and software systems are examples of complex systems built to serve the needs of our societies.

Yet the development of a complex system becomes increasingly more challenging. Systems become bigger and more complex. Pressure on time, costs, and performance increases due to more intense competition. Despite significant efforts of numerous intelligent individuals, the development does not always go according to plan. There are many accounts of failures in product development and problems in the form of iterations, rework, overrun costs, delay, etc.

Due to its importance and challenges, scholars and practitioners alike have devoted a tremendous amount of effort to understanding the development of a complex system. A good part of such effort is to find a way to structure the development process in the most effective manner. Process models, such as concurrent engineering, the spiral software model, etc., arose in different contexts to propose how development tasks should be undertaken in the development process.

Mathematical models, as an abstraction tool, can help us understand how to structure a development process more effectively. Such knowledge enables us to apply the same principles to a wider context. The goal of this thesis is to come up with a model that expands our understanding of a development process from a variety of different aspects.

This chapter provides a motivation for the model that is the center of this thesis. It begins with the typical views of a development process and ends with the risk dynamics as an abstract way to see the problems. First, Section 1.2 describes complex system development and the associated issues of risk, iteration, and rework. Section 1.3 reviews process models and describes how process models affect information and decision pattern in a development process.
Section 1.4 then explains how such patterns affect the evolution of risk in the development process, hence the risk dynamics model. From this motivation of risk dynamics model, Section 1.5 explains the research focus of the thesis. Finally, Section 1.6 provides the structure of the thesis and the plans for the remaining chapters.

1.2 Complex System Development

The issues in complex system development can be better understood with a recognition that such issues are related to three interrelated domains: the system to be created (product domain), the organization that creates it (organization domain), and the process by which the creation is carried out (process domain) [Eppinger and Salminen (2001)].

**System (Product)**

Rechtin (1991) defines systems as the collections of different elements which together produce results unachievable by the elements alone. Systems themselves consist of functional and physical domains.

The system consists of different elements working together to produce the desired results. These elements and their relationships constitute the physical domain of the system. The results, or system functions, are the purposes for which we build the system. These results, also called emergent properties, goals, intents, or functions, constitute the functional domain of the system.

The dependencies in a complex system result from the interrelationships between these two domains. They underlie the difficulty in the design and the development of a complex system and represent the major challenges that need to be overcome by the development organization.

**Organization**

A development organization carries out the development of a complex system. Such an organization typically involves many people who in turn communicate with each other to carry out necessary tasks to create the system.

The communication within a development organization is the major driver of the complex system development. Its effectiveness is dictated by many factors. The location of various team members, the allocation of people into teams, the meeting schedule, and the media of communication affect how effective the communication can be and thus dictate the performance of a development organization in carrying out development tasks.
Process

The creation of a complex system is divided up into tasks, subtasks, or work units. Often times, these activities are organized into a formal process. An example for such a process is the one which proceeds from requirements to conceptual development, system-level design, detailed design, testing and refinement, and production ramp-up [Ulrich and Eppinger (2000)]. The structure of activities is the development process has a major impact on complex system development.

Domain Interactions

These three domains in turn have interactions with each other in the following ways.

Product domain and process domain interact. Von Hippel (1990) characterizes development activities as problem-solving activities, where the goals and constraints are given and the task is to come up with a technical system that satisfies them. Tasks are usually associated with the system components to be developed. Therefore, the dependencies among system components translate into the dependencies among tasks, hence the interaction of product and process.

Process domain and organization domain interact. Development tasks are allocated to teams of people to be carried out. Due to the dependencies among development tasks, some kind of coordination is needed to ensure that the information exchange happens at the right time among the right people responsible for interrelated development tasks. In this regard, the task dependencies which underlie the development process are closely related to the communication that happens in the development organization.

Finally, product domain and organization domain interact. The dependencies in the system under development drive the dependencies in the development tasks. As mentioned before, these tasks in turn affect the communication patterns in the organization. Therefore, the dependencies of a complex system are also the driver of communication within the development organization.

Development Risks

The interactions of the three domains pose some major challenges in the development of a complex system. In this light, we can see that a complex system development involves marshalling an organization to undertake interdependent tasks to create a complex entity which delivers intended results.
For such a challenging endeavor, there is likelihood that the goal will not be met. Development risks represent a mismatch between the goal in the development and the actual outcome. These risks are classified into six different types in Browning (1998).

These risk categories can in turn be reclassified into two major sets. The first set classifies risks according to the causes of such risks. These include technology risks, market risks, and business risks. The second set classifies risks according to the effects they have on the system development. These include performance risks, schedule risks, and cost risks.

Cause-based classification of risks

1. Technology risks: This type of risk is related to the uncertainty regarding the functioning of the technology selected for the project.

2. Market risks: This type of risk is related to the uncertainty regarding the market acceptance of the system under development.

3. Business risks: This type of risk is related to the uncertainty regarding the business environment in which the system is planned for. This is a broad category of risks and might be associated with the economic, financial, societal, and other types of factors.

These cause-based risks correspond to different hierarchies of a system. In this view, a system comprises different elements or components working together to produce the results. On the other hand, a system is also part of a business unit which belongs to a larger societal system. Risks correspond to the requirements posed by any of these hierarchies of the system.

Technology risks correspond to the component-level requirements. They are related to the question of whether a particular component delivers the intended functions. For example, the engine of a car is determined by technical factors such as horsepower, etc.

Market risks correspond to the system-level requirements. They are related to the question of whether the product satisfies the requirements of the potential customers. For example, a customer evaluates a car not only on the technical factors, but on a lot of other aspects such as safety, fuel efficiency, comfort, etc.

Finally, business risks correspond to the business unit-level requirements. They are related to the question of whether the system results in a viable business. Such viability is determined not only by the market acceptance, but also by factors such as the profitability, regulation compliance, etc.
Effect-based classification of risks

1. Performance risks: This is associated with the likelihood and the degree to which the performance of the system under development deviates from the goal. The performance in this case can be both technical performance and other types of performances such as the price points, etc.

2. Schedule risks: This is associated with the likelihood and the degree to which the project is finished within the span of time that is planned.

3. Costs risks: This is associated with the likelihood and the degree to which the development costs stay within the limit decided upon in the project plan.

These effect-based risks result from the challenges in meeting the requirements at different levels of the system. They are closely related as trade-offs need to be made when not all the goals can be met. For example, in some occasions the development team needs to sacrifice the system performance in order to deliver the product on time and vice versa.

*Iteration and Rework*

With regard to the development risks, development organizations do not just proceed in their work from the beginning to the end, only to learn that they miss their goals. Along the process, they also evaluate how they perform and make necessary changes. Such changes result in the iteration and rework in the development process.

Iteration has been defined as “the repetition of activities to improve an evolving design” in Eppinger et al. (1997). Iteration in turn implies rework, which need to be undertaken so that the resulting system meets the objectives. Generally, iteration is triggered by the discovery of new information about such objectives, which can come from upstream tasks, couple tasks, or downstream tasks.

Several studies have documented iteration as the critical drivers of the development process [Clark and Wheelwright (1993), Eppinger et al. (1994), Osborne (1993), Wheelwright and Clark (1992), Whitney (1990)]. In particular, Osborne (1993) found that iteration accounted for between 13% and 70% of total development time for semiconductor development activities at Intel.
Process improvement

Due to the challenges posed by development risks, iteration and rework, many approaches have been proposed to improve the development process of a complex system. These approaches include software systems like Computer-aided Design (CAD), conceptual tools like the House of Quality [Hauser and Clausing (1988)], design heuristics, and process models such as the concurrent engineering practice [Nevins and Whitney (1989)] and the spiral model for software development [Boehm (1988)].

1.3 Process Models

Process models serve as platforms to structure the development process. They propose how to execute development processes, what tasks are to be undertaken, and the location of those tasks in the process. These process models often arise in reaction to the established practices in the industry. Such established practices sometimes are ineffective and result in a poor process performance. Process models come in different names, as they arise from different industries and from different stages of development.

Despite their apparent differences, these process models all aim to restructure the tasks in the processes to improve the process performance. Because they focus on the structure of the tasks rather than on the specific systems, they serve as a good starting point for us to see the effects of the process structure on the performance of the development process.

As will be seen, the essence of these process models lies in the way they affect the patterns of information generation and decision making. To see this, I will illustrate examples in the following process models.

1. Concurrent engineering
2. Prototyping method
3. Iterative model for software development
4. System-focused development for R&D
5. Set-based design method

A Risk Dynamics Model of Complex System Development
Concurrent Engineering

Concurrent engineering suggests that we consider design issues simultaneously where they may have been considered sequentially in the past [Nevins and Whitney (1989)]. The practice has become prevalent in engineering organizations because it can significantly reduce the cycle time.

This practice arose in reaction to the practice in which one set of design issues are considered and decided upon before another set of design issues are considered ("throw it over the wall").

The most salient features of the concurrent engineering practice, as the name suggests, lie in the sequencing of the decisions in the engineering tasks. This new task sequence implies concurrent decision making among engineers of different disciplines. In a more typical case, this means that product engineers and process engineers make decisions on their respective systems jointly rather than in sequence as in the "throw it over the wall" approach.

The practice also prescribes an interdisciplinary information exchange among teams. Meetings are held among engineers in different disciplines to ensure that their individual issues are discovered early and communicated to all parties involved. These engineers are responsible for various subsystems of the system under development. Their participation in the design meetings ensure that information regarding their subsystems is generated and distributed early on in the process.

In this view, concurrent engineering essentially recommends a change in the pattern of the decision making and the information generation in the design process. Decisions on a variety of subsystems are made simultaneously, while the information related to such subsystems is also considered jointly by all parties involved in making such decisions. Concurrent engineering results in a process in which decision making and information generation is performed concurrently for all subsystems.

Prototyping Method

Ulrich and Eppinger (2000) defines a prototype as an approximation of the product along one or more dimensions of interest. Prototypes range from concept sketches, fully functional artifacts, to simulations, and can be classified into physical and analytical prototypes. Prototypes are used for both technical and management reasons. They are also used in both hardware and software development. When prototype is formally planned into the process, we might call the resulting process a prototyping process.
Prototype is essentially an information-generation tool. They are used to extract market or technological information necessary for the system development. Prototypes are used to help a team learn more about the customer requirements or are also used to determine the feasibility of a technology. From this point of view, a prototyping process is a process in which information is generated throughout the process with the aid of the prototypes.

The information generated by prototypes drives the decision making in a development process. Prototypes are used to assess certain properties of the system to help the team make decisions which are related to the properties in question. For example, a physical prototype is used to determine the look and feel of a product and help drive the decisions regarding such look and feel. On the other hand, a simulation can be used to assess certain technical performances and help narrow down the technical choices which affect such performances.

Prototyping process is therefore a process in which the information and the decision are driven by the prototypes that are planned along the process. Information is gathered as prototypes are created. Such information in turn drives the decisions on the system design. A good prototyping process is the one in which relevant information is gathered early through the use of prototypes, which in turn drives the important decisions early in the process.

**Iterative Model for Software Development**

There are numerous process models that can be classified as iterative models for software development. Here, I focus on the spiral process model [Boehm (1988)], which is usually thought of as a meta-model that includes other models as special cases.

In Boehm (1988), the spiral model of software development is proposed as an alternative to the waterfall model [Royce (1970)]. In contrast to the waterfall model, in which the development is conducted sequentially from the design phase to the testing phase, with a possible feedback only to the previous phase, the spiral model calls for a systematic plan to handle various types risks in the software development.

In the spiral model, the software development is done in several cycles. Each cycle considers a certain type of risks and consists of four major phases to 1) Determine objectives, alternatives, constraints, 2) Evaluate alternatives, identify, resolve risks, 3) Develop, verify next-level product, and 4) Plan next phases.
The basic difference of the spiral model from the waterfall model lies in the systematic information gathering and decision making, which is similar to a prototyping process. Instead of assuming that the team has all the information early in the process, the spiral model suggests that the team obtains information corresponding to certain types of risks in phase 1) and 2). Such information then allows the team to make decisions and proceed with the development in phase 3) and 4). The cycle is repeated over and over until all risk issues are sufficiently handled.

*System-focused Development for R&D*

Iansiti (1995) prescribes a system-focused approach for the development which is closely related to research. This is an alternative to the traditional approach in which two activities are done in sequence, i.e. research, followed by development. The system-focused approach calls for technology integration activities, which are the activities that ensure that system components that result from research activities can work together to deliver functions and satisfy cost constraints in the development phase.

In this case, the role of technology integration activities is to generate information regarding the system functions from research activities early in the process. While the traditional process waits until research is finished to move on to development, the system-focused approach incorporates activities that help the team assess the consequences on the system of the technology in question early on.

Such information helps the research team make a better decision in their work. By arming the team with information regarding the system properties of the technology under research, the system-focused approach ensures that the team makes the appropriate decision regarding the technology early, thus avoiding the costly changes that arise late in the process.

*Set-based Design Method*

Toyota Motor Company uses a process called “set-based concurrent engineering” in its design process [Ward et al. (1995)]. In contrast to other process models, set-based design method is a philosophy rather than a prescription on how to restructure the development process. In this approach, the engineers explore all the possible sets of design together without making a commitment until absolutely necessary. The process moves slowly but with high certainty.

This process calls for set-based parameters rather than point values in making decisions. Because the decision parameters are sets, the team can narrow the set of possible design alternatives as far
as the information permits without making discrete changes as in the traditional design. The process also calls for extensive use of "lessons learned" books. The practice helps the team effectively glean information from design and make decisions in the most effective way.

From the above discussions of various process models, we can see that they all aim to restructure the pattern of information generation and decision making in the development process. Building on this, the next section will show how such patterns are related to development risks.

1.4 Risk Dynamics

The preceding discussion on process models indicates that the essential feature of a development process lies in its patterns of information generation and decision making. To provide a motivation for a risk perspective of the development process, this section will in turn provide an explanation of how such patterns affect development risks.

It is important to note that the discussions in this section should be considered as a motivation for the model rather than a rigorous logical proof. The detailed formulation, which is much more sophisticated than what is given in this section, will be provided later on in Chapter 3.

Mechanisms: Information and Decision

The development process can be thought of in terms of a series of decisions (D), each specifying certain parameters given the parameters that have been specified by earlier decisions. For example, in the conceptual design stage, we take the inputs in the form of customer requirements and technological possibilities and come up with the system concepts that satisfy the purposes. From such a concept, the following stages focus on elaborating and refining the system model further until we derive at a complete model of the system to be built.

The tremendous complexity of the system often times challenges the cognitive ability of the organization, even a highly effective one. In such case, the information (I) necessary for carrying out development tasks or making decisions is not available at all time to all people involved. The complexity of the system sometimes makes it impossible for the developers to be absolutely certain about the consequences of certain decisions until they actually proceed.

The inadequacy of information is the root cause of risks in complex system development. Without perfect knowledge, engineers or developers need to proceed with their tasks by using preliminary information, recognizing that changes might be needed later. Here, they need go
ahead and make decisions with the information they have at their disposal. (I-D). While proceeding with their tasks, they anticipate that the lacking information will become available later on in the process, either from their own work or from other members of the team. In either case, information results from the decisions that are made early on in the process (D-I). Finally, equipped with the better information, they then proceed and make decisions based on the information that has been improved (I-D) and the process repeats itself.

From this perspective, the development process can be viewed as a succession of activities which in turn lead to information and decision (Figure 1.1). Development activities take as the inputs the decisions made based on the best available information and create development work as the outputs.

On the other hand, the information in the development process is generated from certain information-generation activities, which need to be undertaken to bring about the information. Examples as given in the previous section include prototyping, design meeting, etc. These activities take as the inputs the work that has been done with preliminary information. From such work, they create better information as the outputs into the process.

From one perspective, the development process is an information generation process (I-D-I) in which better information is generated as the process moves on. From another perspective, the development process is a decision updating process in which better decisions are made from the preliminary decisions (D-I-D). In other words, information and decision are shown to be closely related in the development process.

![Figure 1.1 Risk Dynamics](image-url)

*Chapter 1: Research Motivation*
Measures: Probability and Impact of Risk

Risk, as a measure of how things can go wrong or how the development can deviate from the goal, is usually thought of in terms of two basic components [Smith and Reinertsen (1998)]. The first component is called the probability of risk. The probability of risk is a measure of the likelihood that things will go wrong. The second component is called the impact of risk. The impact of risk is a measure of penalty when things go wrong. It can be in the form of the additional costs, the amount of rework, or the performance penalty resulting from such occurrence.

As we make our progress in the development process, these two components of risk evolve according to how the process is managed. (Figure 1.2) They also affect the iteration and rework in the development process. The probability of risk is a measure of how likely an iteration will happen while the impact of risk is a measure of how much rework is needed when such iteration actually occurs.

In general, the impact of risk increases over time. For example, when the measure is related to the amount of work, the impact increases due to the fact that as more development work is done, the amount of rework to be done to accommodate changes when things go wrong is correspondingly higher.

On the other hand, the probability of risk tends to decrease over time. If the process is managed well, the learning that the development team accumulates over time will help them make a better informed decision. Thus the probability that things will go wrong will be reduced.

Figure 1.2 Evolutions of Two Components of Risk

A Risk Dynamics Model of Complex System Development
Risk dynamics in this thesis refer to the evolution of these two components of risk. From the above discussion, we see that such evolution depends on the way the development process is managed. In general, the desirable characteristic of the risk dynamics is such that the probability of risk decreases quickly while the impact of risk increases slowly.

**Risk Dynamics**

The two mechanisms of risk, the information and the decision pattern, have major impacts on the evolution of the risk measures, the probability and the impact of risk, and thus the risk dynamics.

First, it is important to recognize that some decisions are more costly to change than others. This is due to the fact that making a decision implies carrying out certain tasks based on such decision. For example, the overall specifications of a car are more costly to change than, say, the position of some screws, because the former has consequences to more development tasks than the latter.

The tasks thus carried out in turn determine the impact of risks. Tasks that have been carried out based on certain decisions are subject to rework in the future if such decisions are changed. The impact of risks gets accumulated as some decisions are made and the development tasks are executed based on those decisions. From this point of view, the task sequence, which is implied by the decision sequence, determines the evolution of the impact of risk.

Next, all else being equal, better information implies a better decision. The better decision leads to a lower probability that things will go wrong. In other words, the timing of information has a direct effect on the evolution of the probability of risk. Information that comes at the right time help reduces the probability of risk in the right manner. From this framework, the information-generation activities will be referred to as risk-mitigation activities due to their effects on risks.

Less obviously, the sequence of development tasks also determines the evolution of the probability of risk as well. After all, risk-mitigation activities generate information using the output of development activities as the inputs. The task sequence determines the quality of the information generated by the risk-mitigation activities. In this light, both the risk-mitigation activities and the task sequence affect the evolution of probability of risk.

**Putting it all together**

The purpose of the discussion so far is to put forth a framework for the risk dynamics model of a development process. From examples in the process models, the development process is
conceptualized as interplay between the information and the decision in the development process. In the discussion on risk dynamics, the pattern of information generation and the decision making in turn dictate the evolution of the risk measures, the probability of risk and the impact of risk. These risk measures, in turn, affect the iterations and rework, which are the major determinants of the performance of a development process.

In short, here I put forth an argument that a development process can therefore be understood from the point of view of the risk dynamics, i.e. the evolution of the key parameters of risks.

1.5 Research Focus

Risk Dynamics Model

The conceptual model that has been discussed thus far motivates a risk-focused perspective for looking at the development process of a complex system. More specifically, risk dynamics are put forth as way to explain how iteration, rework, and other process metrics are affected by process parameters that characterize process models we see in practice. Figure 1.3 depicts this conceptual framework of risk dynamics.

![Risk Dynamics Framework](image)

Figure 1.3 Risk Dynamics Framework

Given the central importance of risk in a development process, I also hold the view that the management of a development process is nothing more than the management of risk itself. Therefore, a model that describes how risk dynamics take place in the development process will
provide us with a mental tool to understand how we can manage the risk, hence the development process, better.

Proposal

The research objective for this thesis is to model the development process of a complex system using the conceptual framework of risk dynamics. The model should describe how process parameters such as risk-mitigation activity and task sequence, as mentioned in previous discussions, affect the pattern of information generation and decision making in the development process.

The model provides an explanation for how pattern of information generation and decision making affect the evolution of key risk measures, namely the probability of occurrence and the amount at risk. The risk measures, in turn, should also be used to model the iteration, rework, and other phenomena in the development process.

In other words, the major purpose is to provide a mathematical model that links all these elements together in an integrative and coherent fashion to explain how the development process unfolds. To clarify the research objective, the next section will give a general idea of the nature of the model.

Methodology

![Figure 1.4 Model Structure](image)

The model under study will have the following characteristics (see Figure 1.4)

First, it will have two basic elements, the system to be developed, and the process to be structured. The system is decomposed into its constituent elements (to be defined and discussed in Chapter 3). The process structure consists of tasks that are sequenced in certain ways, with risk-mitigation activities that dictate the timing of information generation in the process.

Chapter 1: Research Motivation
The model will have the objective that is directly related to the process performance. In this case, I choose the amount of work as the primary metric for process performance, which can be used to derive the development cost or cycle time (section 7.3).

In addition, the model will result in an optimization problem. Such model makes it possible for us to understand the relationships between the model parameters under the optimal condition. This allows us to make some statements about the 'best' or the 'ideal' process policy that corresponds to the optimal solution.

Questions and Concerns

1. What are the relationships with Bayesian model?

Before we go further, it is also important to discuss what this research is not. The proposed model is not a Bayesian model, which uses Bayes’ theorem to prescribe how to update decision making in an iterative fashion as more information become available. The proposed model is very similar to Bayesian models, but the difference lies in the fact that this process of updating is presumed to be exclusively outside of the model. The procedure for such updates is assumed to be done by the development team and will not be discussed. As will be seen in the formulation, the model only captures the effect of such decisions from the process point of view.

2. What are the relationships with decomposition and clustering?

This model is closely related to the decomposition and the clustering decision in the product development. First of all, the decomposition of the system into constituent elements and their resulting relationships serve as the inputs into the model. The relationships between such parameters and the resulting process performance form the core part of the analysis. This means that the model can be used as a tool to evaluate the decomposition with respect to the resulting process.

Second, this model is related to the clustering decision as well. However, we can see in the literature that clustering can be done to sequence the tasks as well as to divide tasks among teams. Since this model only deals with process, it pertains to the clustering only from the point of view of sequencing the tasks. The division of tasks among teams is not explicitly captured in this model.
**Benefits and Validations**

There are several benefits to be obtained from such a model. The following paragraphs list some of the benefits that can be seen directly from the features of the model discussed so far. They also describe the work done on this thesis that either directly demonstrates the benefits or provides the validations that they can be derived in practice.

1. **The model will help us better understand the interactions of product and process**

The first benefit is the direct result of the structure of the model, whose two primary elements are the system (product domain) and the tasks (process domain). As will be shown in the next chapter, which provides a survey of the modeling literature, this model is unique in that it explicitly captures the interactions between these two domains, while other models capture them in a subtle way. This explicitness allows us to deal with the issues of the interactions between these two domains in a direct way.

To demonstrate how we can derive a better understanding the interactions of these two domains, I devote the whole chapter (Chapter 4) to the analysis of the model. I focus on special cases of the model with some simplifying assumptions. Such models give rise to important structural results that illustrate the relationships between the product and the process domain. These relationships provide some important insights into how such interactions affect the performance of processes. They also serve as an example of how such study can be done.

2. **The model aids in the design and analysis of process models**

Process models that arise in many industries seem to have similar effects on the development process from the point of view of risk. Their prescriptions lead to changes in the information pattern and the decision pattern in the process, which in turn drive the evolution of risk in a more efficient manner.

From this view, the model that captures such risk dynamics should enable us to capture the essence of these process models in the common language, mathematics. This will make it easier for people in different industries to compare and contrast their process models, which otherwise would be difficult due to terms and concepts that are different in different industries. The comparison, if possible, can lead to a cross learning that helps a good practice in one industry to spread to others.
There will likely be a debate on the generality of the model proposed in this thesis. Although this model has been constructed with generality as one overriding goal, the thesis provides only one case study in software development, primarily to validate the applicability of the model. This issue of generality will therefore not be fully addressed in this thesis and will likely be the subject of further debate. In any case, I maintain that the model proposed in this thesis represents at least an attempt to reconcile practices in different industries under the same mathematical language.

3. The model provides us with a decision support tool

The last, definitely not least, and perhaps the most obvious benefit of the model is that it can be used as a decision support tool. The model, with its structure, objectives, and optimization nature can be used in the project planning of a development process. More specifically, the model should answer the questions related to the sequence of execution or the partitioning of the system to be developed in succession, i.e. how should the system be divided up into subsystems that are developed first, second, and so on.

I provide a validation for the model as a decision support tool in two ways. First, I provide a case study to show how the parameters of the model can be obtained. Second, I also provide a chapter on the numerical approach of the model. The chapter provides an analysis of the complexity of the system, which will be shown to be of NP-complete class. Because of this, the chapter also recommends several heuristics that are found to be empirically effective in a number of numerical examples.
1.6 Thesis Structure

The structure of the remaining chapters of this thesis is explained in the following paragraphs.

Chapter 2 provides a discussion of the modeling literature in product development. Here, we will look at various mathematical models that touch upon some of the issues mentioned in the research objective. From such literature, I provide a critique on their strengths and weaknesses.

Chapter 3 provides a complete formulation of the model. It describes the risk dynamics model which prescribes how various types of work is generated in a process. From such a model, the chapter then formulates a work-minimization problem, which will be the subject of study in the chapters that follow.

Chapter 4 is the analytical section of this thesis. From the optimization problem in chapter 3, this chapter looks at its meaning in details. I will provide an analytical study of some important special cases of the optimization problem. This will provide some insights into the management of a development process. The chapter will also answer to the foregoing questions about the relationships between product domain and process domain in product development.

Chapter 5 provides a real application of the model to an industrial example. In this chapter, the focus will be on the application of the model to the ongoing software development process of SuperPages.com at Verizon. This chapter will provide a detailed account of how one can obtain the parameters of the model and how the model helps the decision making in the development process.

Chapter 6 provides a treatment of the numerical aspects of the model. Given the optimization problem, I will illustrate that the problem in general is a complex one, i.e. an NP-complete problem. From such observation, I will propose several heuristics that can aid in solving the model numerically. Numerical examples are also provided to illustrate the effectiveness of such heuristics.

Finally, Chapter 7 provides a conclusion to our study and explores the avenues of future research.

Chapter 1: Research Motivation
CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

There are many models which capture the dynamic aspects of a product development process. These models deal with many different contexts, e.g. concurrent engineering, prototyping, etc. They vary in the scope and the complexity of the model. They also employ different concepts and techniques for arriving at important results.

The purpose of this chapter is to provide the reader with some knowledge about the literature on the mathematical models of product development processes. A good understanding on the scopes, the approaches, and the limitations of the models in the extant literature will help the reader better understand the issues that led to the model proposed in this thesis.

The choice of literature to be reviewed in this chapter needs to be selective yet representative to provide the readers with a background for the remaining chapters. With this in mind, the models that are included are those which give a good representation of the variety of contexts, issues and the modeling approaches.

2.2 Context, Scope, and Results

I divide the models of a product development process into two families, the iterative process models and the DSM-based models.

Iterative Process Models

The first group, to be called iterative process models has modeling approaches that bear similarities to Ha and Porteus (1995), the first model within this family. The paper models progress reviews in concurrent design for manufacturability. The model consists of two development activities, product design and process design. The objective function to be minimized is the total cycle time and is driven by the information transfer between the two activities.
Product design is assumed to generate information to process design and to itself as it progresses. The progress review, which is assumed to result in setup time and inform time, releases this information and has beneficial effects on the cycle time in two ways.

First, the release of information increases the maximum amount of work that can be done on process design, thus increasing the parallel portion of both activities. Second, it frees a portion of product design that is subject to design flaws, thus reducing the amount of rework on product design. In the stationary version of this model, it is shown that the ratio of product design time to process design time and the probability of design flaws together determine the optimal review periods.

Within this group, there are three papers which consider the overlapping problem in design process. Krishnan et al. (1997) models a process with two overlapping tasks. In this model, the upstream task generates information regarding parameter values necessary for the downstream task. Each information transfer releases such information and generates an amount of work for the downstream task. This amount of work, the duration for a downstream iteration, is a function of the change in the value of the exchanged information. The function is called “sensitivity function”.

The change in the value of the exchanged information in turn is a function of the change in “evolution function” defined as the rate at which the upstream activity narrows the width of the parameter interval to zero. These downstream durations in turn determine the cycle time of the process. The transfer of information in each iteration is assumed to happen without additional costs.

Similarly, Loch and Terweisch (1998) models a process with upstream task and downstream task. Engineering changes are assumed to arrive according to a heterogeneous Poisson process from the execution of the upstream task. These changes are released to downstream in each communication batch and have an impact on the downstream task in the form of rework. The amount of rework per change is a function of downstream time called “impact function” which increases as downstream task progresses. Each communication batch costs a certain amount of time.

The objective of the model is to minimize the cycle time, which in turn is derived from the amount of overlapping, the pre-communication cost, the communication cost and downstream rework. They found that if evolution is fast, i.e. if most changes come at the beginning, the optimal communication frequency increases over time, and vice versa.
In addition, higher uncertainty and higher dependence decrease the optimal amount of overlap, but slower evolution may increase or decrease it. They also found that the optimal solution is either extensively pre-communicate (frontload) and then overlap fully, or to not pre-communicate and then proceed sequentially.

Roemer et al. (2000) generalizes the overlapping problems to the case of multiple tasks. This model, however, still assumes a unidirectional information flow from a task only to its consecutive task. Here, evolution and sensitivity is combined into a function of the overlap of two consecutive tasks called an "extended design function". This function is the expected design time increase at a particular stage due to overlapping with the former stage.

The paper suggests one way to model this function using the probability function. This represents the chance that a unit of downstream task needs rework and is a function of how long that particular unit of work is executed before receiving the accurate information upon the completion of upstream task. The extended design function is therefore the integral of this probability function over the overlapping duration. The paper gives an algorithm to minimize the total cycle time and the total costs from this formulation of the problem.

Thomke and Bell (2001) considers the frequency, timing, and fidelity of testing or prototyping activities. In this case, the testing activity generates information about the customer requirements and/or technical feasibility. This information is captured as the number of problems to be corrected. The fidelity of a test is the degree to which the test reveals the problems accumulated so far. The cost of a test is a function of the test fidelity.

The cost of rework is the product of the number of revealed problems and the unit cost of rework, which is a function of time at which the test occurs. The structural result on the optimal test frequency confirms the results of previous papers. More importantly, the paper also provides insights about how to select test with different levels of fidelity in different circumstances.
DSM-based Models

The second group, DSM-based models, all derives modeling approaches from the Design Structure Matrix, a system engineering tool widely used to understand the dependencies in the product development.

Smith and Eppinger (1997a) uses an adapted version of DSM, the work transformation matrix (WTM), to characterize the evolution of a set of highly dependent design tasks in an engineering design process. The design tasks are done in parallel and in stages.

The amount of work of a task in a stage is the weighted sum of the amount of work of all tasks in the previous stage. The weight coefficients represent the strength of dependencies between the task and all other tasks. The underlying assumption is that more rework for a particular task is incurred when more work is done on the information-generating tasks.

The paper uses eigenvalues and eigenvectors of WTM to describe the mode of evolution of the design process. The important idea in this paper is the dominant design modes, mathematically the weighted sums of design tasks, which represent the functional requirements that take most time and resource for the development team to satisfy.

Smith and Eppinger (1997b) assumes a sequential task execution and uses Markov chain to model the development cycle time. The process is assumed to happen in stages, each corresponding to a set of active development tasks. Each stage adds exactly one development task to the previous stage. Within each stage, the execution of one task provides information that might necessitate the rework of another task in that stage. The process is characterized as a Markov chain with rework probabilities represented in the DSM. The paper provides algorithm for finding optimal ordering of task and presents some insights derived from such model.

Carrascosa (1999) generalizes these models to the case of arbitrary execution, i.e. not sequential, nor parallel. The paper characterizes a development process using Markov chain and the concept of the probability of change and the impact of change. This results in a mathematical model that can be used to predict the process performance and derive useful insights about development process.

A Risk Dynamics Model of Complex System Development
2.3 Benefits and Limitations

The models described in the above paragraphs give us many insights on the relationships between various aspects of information and process performances. These models help us understand a variety of issues in product development process such as rework, information, dependencies, etc.

More particularly, they provide us with an understanding of how two important process parameters, the information timing and the task sequence affect the performance of the development process. Some of these models help us understand the information timing in terms of how often should information exchange take place between tasks. This issue is closely related to those regarding the appropriate number of iteration and how big each iteration should be. In terms of the task sequence, some models help us understand how to sequence the task. More generally, they help us understand issues such as overlapping and also the relative importance of tasks.

These models, however, tend to have strengths in either one of these issues. Due to its structure, iterative process models help us understand the issues of information timing very well. However, they do not provide us with much insight concerning the task sequence issues. One of the reasons is that the number of tasks in these models is limited. Most consider only one or two tasks. In cases when multiple tasks are considered, such as Roemer et al. (2000), the sequence of task is pre-specified and the only remaining decisions to be determined are the amount of overlapping between two consecutive tasks.

On the other hand, DSM-based models help us understand the issues of task sequence. If this is not done explicitly, the models provide an answer to the priority of tasks under development [Smith and Eppinger (1997a)]. However, since these models permit multiple tasks and general directionality of information flow, the resulting models tend to be very complex, and often times intractable. This means that these models are more suitable as a performance evaluation model to evaluate the consequences of candidate policies as opposed to an optimization model where an optimal policy can be found from a given set of model parameters. When it comes to the information timing, DSM usually has no prescription, for the reason to be explained later. In the case they do [Smith and Eppinger (1998)], the model is limited only to two iterations due to the intractability issues mentioned before.
2.4 Problems with Information

In the previous section, the models of the two families are shown to have weaknesses in different areas. From the point of view of process parameters, some do not handle the task sequence very well, while others do not handle the information timing issues. In fact, the reasons for these limitations are very simple. The process parameters that are not handled are simply assumed to be out of the scope of the model.

To understand how and why this is the case, first we need to recognize that these models all conform to the information-processing paradigm, under which development tasks are considered information-processing activities. This is manifested in the assumptions that the amount of work for a particular task depends on the information that goes into it. The way each of them models how information generates work determines the characteristics, the scopes, and the limitations of the model.

**Directionality and Timing**

In iterative product development models, the issue of task sequence has been taken out of the models because they pre-specify the task sequence by limiting the directionality of information flow.

This can be seen in two ways. First, the information is explicitly assumed to flow in one direction, hence the definition of upstream task and downstream task. Second, note that a task in these models is usually a set of many subtasks done in sequence, e.g. product design or process design tasks. In operationalizing these models, model parameters are functions of time, or the amount of overlap, rather than task. Such specifications make it impossible the analysis of the case when we alter the sequence within the task, hence the implicit unidirectionality of information flow and the prespecified sequence.

In these models, because we cannot prioritize between upstream/downstream or within the task, task sequence has been essentially decided and cannot be studied fully. The symptom can be seen in that there is a limit to the number of tasks in these models. In the case of multiple tasks, they are pre-sequenced and the remaining decision parameters are only the amount of overlap.
In the same manner, DSM-based models could not provide a prescription regarding the information timing because such decisions are exogenous to the model. The structure of information transfer has been pre-determined due to the fact that these models implicitly assume the instantaneous generation of rework after information transfer. Since they only model the work generation process, this has the effect of specifying the timing of the information transfer at the time of work generation.

In some models, e.g. Smith and Eppinger (1997b), information is assumed to flow sequentially from one task to another. In some others, e.g. Smith and Eppinger (1997a), each task generates information to all other tasks right at the end of each iteration.

Often times, the DSM-based models are not analytically tractable. Browning (1998) generalizes the structure of information transfer by modeling the process using the band of activities. Smith and Eppinger (1998) models the alternative between the concurrent engineering in which all tasks are taken care at the same time and the sequential process in which they are considered in two successive iteration. Carrascosa (1999) incorporates the concept proposed in Krishnan et al. (1997) into DSM-based model. In these cases, though the information exchange allows some flexibility, the resulting model is intractable and cannot be analyzed by the analytical method.

Information Model

From our discussions, it is possible for us to postulate the fundamental issues faced by these models. This has to do with the how these models handle information. First, it is important to notice that under the information-processing paradigm, information is generated from tasks while tasks also make use of information. Thus, a good model of information under this paradigm has to specify the relationships between information and task in both directions.

Though these models all conform to information-processing paradigm, the information in these models is handled only in a subtle way, as the means to describe how work, the process measures of interest, is generated. The other side of the picture, which is about how tasks generate information, is not handled explicitly. This is the primary reason why the models are limited in the directionality of information and the timing of information exchange.

In some cases, information is completely hidden behind the scene. This is seen in many DSM-based models, [Smith and Eppinger (1997a), Smith and Eppinger (1997b), Smith and Eppinger (1998), Browning (1998)]. Because there is no conceptual basis for a stock of information, it is assumed to lead directly to amount of work.

Chapter 2: Literature Survey
The benefit of this is that these models gain the generality in the directionality of the information. This allows them to treat the issues related to task sequence (or more generally, task prioritization) in a comprehensive fashion. But at the same time they cannot incorporate the information timing issues under the model.

On the other hand, Ha and Porteus (1995), Loch and Terweisch (1998), and Thomke and Bell (2001) model information in the form of the number of problems or changes, which represent only the indirect measure of information. Because information is not modeled explicitly, these models need to assume the unidirectionality of information. These models have a complete control over the timing of the information, but they lose the directionality.

Krishnan et al. (1997) models information in the form of the design parameters that change over time. This is the most direct way a modeling literature comes to grapple with the conceptual basis of information. The paper derives many insights from the model. But from the point of view of modeling, the paper still needs to assume the unidirectionality of information. This is because the information is modeled only partially. The design parameters represent the information that drives the development tasks. But the model does not specify how tasks in turn generate such information. The evolution of design parameters is assumed to be a function of the elapsed time.

In Roemer et al. (2000), this approach is extended to handle multiple tasks. The model has a complete control of timing, but still assumes unidirectionality. Carrascosa (1999) also extends the concept in Krishnan et al. (1997) to DSM-based model. The model, like others in its family, has a general directionality but the timing of information exchange is still outside of the model.

2.5 Next Directions

We have some clues from looking at the models mentioned above that a better understanding of information can lead us to a more general model which captures a complete set of issues. To do that, we need to look deeply into the system and investigate how the tasks generate information and how they use information. This is an important basis for modeling the risk dynamics in a development process. In fact, this will be the first issue to be discussed in the next chapter on the formulation of the model.
CHAPTER 3: MODEL FORMULATION

3.1. Introduction

This chapter explains the formulation of the model central to the thesis. Before I proceed with the formulation, section 3.2 discusses the premise underlying the model. This premise represents an important conceptual development which enables the formulation of the model and warrants a notice early on in this chapter.

The formulation of the models then follows. Section 3.3 formulates the risk dynamics model, which is essentially a theory on the evolution of risks and on how various types of work are generated in a process. The model answers the question of how the process policy such as task sequence and information timing determines the amount of work in the process.

Section 3.4 discusses in more details important aspects of the risk dynamics model, that of the probability function and the impact function. In Section 3.5, the risk matrix, a representation of the model parameters is discussed. Finally, Section 3.6 formulates a work-minimization problem from the risk dynamics model. The problem is to be used in the remaining of the thesis.

3.2 The Premise

Before we go on, I find it important to state the premise of the model to be formulated in this chapter right at the beginning. This premise is important in several aspects.

1. This premise is the assumption which is not explicitly used in the extant modeling literature. Here, its use in this model represents a conceptual development which I believe is important in the improvement of the modeling approach.

2. In addition to being a unique assumption, it is the only ‘real’ premise in a sense of a synthetic statement which cannot be derived from a logical deduction or a mere structural assumption, which allows for easy formulation and analysis.

3. This premise is important to several results contained in the thesis. Although a lot of other results are still valid without this assumption, the premise enables us to cover a variety of issues that cannot be handled without it.
4. In addition, it is also the feature of the model which most readers might find the most difficult to understand. The premise is relatively subtle conceptually. It is also the assumption which makes the model complicated. Without understanding its importance from both conceptual and modeling point of view, the assumption will likely be thought of as redundant.

Due to its significance, it is important to discuss this premise before we delve into the details of the model. This is so that the reader keeps this in mind and recognizes it while reading the detailed formulation. It will also help the reader understand the relationships between this premise and the results of the model to be discussed in the next chapter.

Instead of explaining it in a lengthy mathematical equation, I state it in plain words, the premise.

The relevant information for the system development comes from its “functions”

Some discussions are needed to explain this important statement.

First, functions in this case have a more general meaning. They include any emergent properties of the system, be they the features of the system, as in software features, the ‘illity’ families such manufacturability, reliability, or things like costs. They are the properties of the system that we care about.

Relevant information is the information that is inherently lacking in the development. In this meaning, I exclude information which is lacking because of organizational issues. For example, information which resides in subteams and not readily available to others is not included in here. The information that is inherently lacking holds back the development in a sense that, without it, the development needs to proceed with uncertainty.

Now, I come to grapple with the validity of this statement. First, this statement is in fact implied by the prescriptions of a lot of process models in the product development. For example, we find that in system-focused development, the role of the technology integration activities is to assess the ‘system functions’ and the ‘costs’ of the system and feed this information to the research team.
In concurrent engineering, we re-sequence the task in order that the important information such as the manufacturability, reliability, and other issues are fed into the product team early in the process. This is in contrast to the 'throw-it-over-the-wall' approach where such properties are assessed later in the process design phase.

In prototyping, the information that we derive from creating the prototypes is related to certain functional properties of the system such as the look and feel, the mechanical properties like torque, energy, etc., or the weights. In other words, these process models all prescribe to feed relevant information related to certain properties of the system early on in the process.

Conceptually, functions can also be seen as the relevant source of information in the development process that helps reduce the uncertainty. Taking the point of view of the risk mechanisms alluded to in Chapter 1, we can see that some information is used to update our assumptions and make necessary changes in order to ensure that the system meets the goal.

Ignoring the organizational issues mentioned above, the only information that can be used to make changes can only come from the functions of the system. Since we assume that we already made the best assumptions before we proceed with the development, such assumptions will be updated only when we find out about whether certain system functions, which are the goals in the development, work or do not work as intended.

The knowledge about the functions of the system provides us with a clue to understand whether the previous assumptions are right or wrong and allows us to make changes correspondingly. Other type of knowledge, though useful, does not lead directly to changes in the assumptions. Therefore, functions are the only source of relevant information, again without organization issues, which allows us to make changes in our assumptions before we proceed.

With this premise stated, we are now ready to proceed to the formulation of the model.
3.3 Risk Dynamics Model

*System model*

The first element to be discussed of the model is the system to be developed. This is shown in Figure 3.1. System is divided into modules, with a mapping of functions to these modules. There also remains uncertainty in some aspects of the system in the form of interfaces, which link the modules together.

Note here that the counterpart of our system model in other literature is the tasks to be undertaken. Here, however, we define the system using three basic conceptual constructs.

![Figure 3.1 System and its various elements](image)

**Figure 3.1 System and its various elements**

**Modules** – the physical elements of the system that needs to be designed and built. This corresponds to system components we can assign engineers to design and develop.

The important aspect related to modules in the model has to do with its resource consumption. We need resources to design and develop modules. If things change, as will be explained in details in sections that follow, we also need to rework on those modules.

More formally, a system consists of \( M \) modules, indexed from 1 to \( M \). Let \( \Sigma \) be the set of all modules, each represented by its index.
**Functions** – the functional behaviors or other emerging properties of the system we intend to create. Assume that there are \( N \) functions for the system in question. Let \( \Phi \) be the set of all functions, each represented by its index.

Functions are closely related to modules. Functions are delivered through a set of modules of the system. A function is defined in this model by the subset of modules that are needed to derive such function. Mathematically, function \( n \) is defined by \( F_n \subseteq \Sigma \), the set of modules needed for that function. A module can belong to several functions and therefore two functions can have some shared modules. This relationship between functions and modules is called a functional mapping and is a very important aspect of the model.

As mentioned earlier, the important aspects of functions have to do with its information. Functions are related to the goal of the system. As we build the system, we are able to determine the degree to which the system satisfies its functions. This is the source of information that allows us to determine the course of actions to ensure that we meet the development goals. The information is modeled in the form of the probability of change corresponding to a function. This will be explained in more details later on in this section.

**Interfaces** – the assumptions in the design. Taking a view of a development task as a problem solving activity, interfaces represent the specifications of such problem. This is closely related to the design parameters in Krishnan et al. (1997). Here, I extend the meaning to cover any aspect of design, whether it can be represented as a quantitative parameter or not. Therefore, interface can be design parameters, assumptions, inputs, etc.

The important aspect of interface is its change. When an interface changes, it affects modules which takes that particular interface as the problem specifications. We need to rework on modules that are related to it to make sure that such modules conform to the new value of the interface.

To formulate this, I assume that the system has \( Q \) interfaces. In other words, I assume that we can somehow cluster the problem specifications, assumptions, into \( Q \) groups. Associated with each interface is the set of modules affected by a change of the interface.
Process structure

The uncertainty calls for an iterative mode of development in which knowledge is gained from building parts of the system. Knowledge gained is then used to reduce the uncertain aspects of the system.

![Diagram](image)

**Figure 3.2 Iterative process**

The nature of the process requires a partitioning of the development of the system modules into iterations as shown in Figure 3.2. In successive iterations, we grow the system by developing more and more modules, while making necessary changes along the way as we learn more about the system. The number of iterations, \( J \), is also one of the decision variables in this model.

There are three stages within an iteration. In the **work stage**, we first develop the modules allocated to that iteration and integrate these modules with the rest of the system that has been developed so far.

Then we test the resulting system against its intended functions during the **risk-mitigation stage**. Examples of activities that can be modeled as risk-mitigation activities include prototyping activity, communication between subteams, technology integration activity, etc.

The information revealed in this phase might necessitate rework of the modules that have been developed up to that point in the **rework stage** before we proceed to the next iteration.

Process policy

As mentioned, the iterative process calls for the allocation of these system modules to iterations.  

We call this allocation the **process policy** and denote the modules allocated to various iterations as

\[
(S^1, S^2, ..., S^J),
\]
where \( S^i \subseteq \Sigma \) represents the set of modules allocated to iteration \( j \). Note that, throughout this thesis, the superscript is reserved for the iteration index, while the subscript will refer to different things. For notational convenience, define

\[
\hat{S}^j = \bigcup_{i=1}^{j} S^i
\]

as the set of modules developed up to iteration \( j \). The process policy has two properties

1. Exclusivity, \( S^i \cap S^j = \emptyset, i \neq j \).

2. Exhaustiveness, \( \bigcup_{j=1}^{J} S^j = \Sigma \)

*Function buildup*

Given a set of modules we can find the corresponding set of functions whose constituent modules belong to that set. Define such mapping as

\[
\mathcal{A}(\hat{S}) = \{ n | F_n \subseteq \hat{S} \}.
\]

Note that \( \mathcal{A}(\hat{S}) \subseteq \Phi \). For a particular process policy \( (S^1, S^2, \ldots, S^j) \), I define an incremental function set for an iteration as the set of functions that emerges in each iteration, i.e. becomes available for scrutiny in the risk-mitigation stage.

\[
\Delta^j = \mathcal{A}(\hat{S}^j) \setminus \mathcal{A}(\hat{S}^{j-1})
\]

Define the incremental function vector as \( (\Delta^1, \Delta^2, \ldots, \Delta^j) \), the set of functions that become available for scrutiny in successive iterations. Each element of this vector represents the set of functions that are added in current iteration in incremental to all previous iterations. These are the functions that just become available as we grow the system.

*Rework buildup*

I assume that in the work stage and the rework stage, we spend development effort in the form of the amount of work to develop the module. In the work stage, we develop modules dedicated to that iteration the first time. In the rework stage, we rework on the modules the second time, the third time, etc.
The amount of work in work stage is assumed to be independent of how we allocate the modules to various iterations. Rework, on the other hand, represents the additional work needed to make the modules work and depends on the module allocations. Rework in a particular iteration is assumed to happen to only modules that have been through work stage up until and including that iteration. In other words, we only perform rework on a module if it has been worked on.

Figure 3.3 shows an example of how this happen. Here, in the first iteration, the first three modules go through work stage. After changes happen in the risk-mitigating activities, these modules are subject to a certain amount of rework.

<table>
<thead>
<tr>
<th>Risk-Work</th>
<th>mitigation</th>
<th>Rework</th>
</tr>
</thead>
<tbody>
<tr>
<td>S^1</td>
<td>O</td>
<td>R^1</td>
</tr>
<tr>
<td>S^2</td>
<td>O</td>
<td>R^4</td>
</tr>
<tr>
<td>S^3</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

**Figure 3.3 Work and rework on modules**

In the same fashion, the next three modules go through work stage in iteration 2. In the rework stage, we might need to rework these six modules that have been through work stage. In the last iteration, the next three modules go through work stage. In this iteration, all nine modules are subject to certain amount of rework.
Risk-mitigation work

The role of the risk-mitigation task is to generate information from the parts of the system that have been developed so far. We can obtain the information in several ways. For example, we can have users use the system and assess the degree to which it meets their needs. If we are interested more in the technical issue, we can test the technical aspects of the system. Risk-mitigation task includes such activities as prototyping, technology integration, communication, etc.

We use such information to update our assumptions, thus causing interface changes. The role of risk-mitigation task is to glean information from the system, resolve risks using such information, and make necessary changes. In other words, risk-mitigation task takes as an input the partial system up to that time, and creates as an output the changes in the interface.

I assume that risk-mitigation tasks necessitate a certain amount of work. In particular, let $L^j$ represents the risk-mitigation work, the amount of work required for risk-mitigation tasks, for iteration $j$. The risk-mitigation work can be formulated in many different ways. For example, if we perform different kinds of risk-mitigation activities in different iterations, the risk-mitigation work could be a function of the iteration index.

$$L^j = l(j)$$

Since risk-mitigation activities are performed on modules, the amount of work required can be a function of such modules, hence

$$L^j = l^j(S^j, S^{j-1}, \ldots, S^1)$$

On the other hand, we can also argue that it is a function of the system functions.

$$L^j = l^j(\Delta^j, \Delta^{j-1}, \ldots, \Delta^1)$$

In this thesis, however, the risk-mitigation work will not be the focus of the attention. I simplify the formulation by assuming that the risk-mitigation work is simply a fixed number

$$L^j = L$$

In doing so, I implicitly assume that the risk-mitigation tasks are the same and do not depend on the process policy, except for the number of iterations.

Chapter 3: Model Formulation
**Interface change: the trigger of rework**

Now we come to the question of what triggers rework. From the definition of system model, we can see that the basis for the development of a module is the interfaces, which represent inputs, specifications, or design assumptions needed to solve the design problems related to that module. Examples of interfaces are design parameters in a mechanical system; database design, network protocol, and user interface specifications in a software system.

We use the preliminary value of the interfaces in the development of modules. The uncertainty in the development arises from the uncertainty in the interfaces of the system. From this, it is clear that the rework in a given module needs to come from the "change" in the interface.

In this model, I assume that these changes take place after the risk-mitigating stage when we learn about the system we develop so far. Based on such knowledge, we then revise our assumptions, thereby changing the interfaces.

In this context, Interface change corresponds with individual iteration in the process. In a given iteration, each interface either changes or does not change. Represent the interface change as a binary random variable. Since there are Q interfaces, define $\overline{I}^j$, a Q-binary random vector, as the interface change associated with iteration $j$, as defined below.

$$\overline{I}^j = (\overline{I}_1^j, \overline{I}_2^j, ..., \overline{I}_Q^j)$$

$$\overline{I}_q^j = \begin{cases} 1 & \text{interface q changes in iteration j} \\ 0 & \text{interface q does not change in iteration j} \end{cases}$$

Rework on a module is a direct result of changes in the interfaces underlying the development of that module. I assume that we can reasonably predict the degree of dependency of module on these interfaces. In other words, I assume that we can approximate the amount of rework that is needed when changes in the interfaces take place. Denote this dependency as $g_m$, which tells us how much rework is needed given a particular profile of interface change.

$$R_m^j = g_m(I^j)$$

where $g_m : \{0, 1\}^Q \rightarrow \mathbb{R}$ is defined for each module $m$ and representing the mapping between a particular realization of the interface change to the amount of rework for that module.
Function: the driver of interface change

Finally, we now define how functions generate the relevant information in the process. To see this, let us go back to revisit how the problem has been formulated so far. I have assumed that the interface specifications represent the best possible state of knowledge at the particular point in time. With the intention to create a system with functions that satisfy the requirements, we do our best in coming up with such specifications. In other words, the functions are the goals we try to achieve; and the interface specifications are done with that goal in mind.

So, what is it that makes such specifications obsolete and necessitates changes in the interfaces? From our foregoing discussion, it is obvious that the answer can be none other than the knowledge about the goals we set out to achieve itself. We have set the specifications using the best knowledge to satisfy the goals. We only change the interfaces when we find out that our goals are compromised and that the only way to rectify the situations is to change the interfaces.

Functions represent the goals in the development process. A partial set of functions represent the subgoals in the development. We make assessment of the goals of the system based on such subgoals. When we only have the design, we can only make a guess about whether or not the system will function as we intend. But as we grow our system by performing real development, we have something concrete to test and verify. New functions emerge and become available for scrutiny. As we add more modules to the system, the new functional behaviors of the system can then be assessed with respect to the goals.

With such view, I argue that the change in interfaces comes about from the system functions available for testing at the time. In other words, the interface change for a given iteration depends on the set of functions that emerge in iteration 1 up to the iteration in question. More specifically, the interface change of iteration $j$, $I_j$ is a function of all the incremental function sets from iteration 1 through iteration $j$. Since $I_j$ is characterized by its probability, I state that

$$Pr[I_j = I_e] = f^j(N, N^{-1}, ..., N^j; I_e)$$

In other words, for all possible $I_e \in \{0, 1\}^Q$, $Pr[I_j = I_e]$, is a function of the system functions in the current iteration, and possibly, the foregoing iterations.
The dependence of the probability of interface change on the functions in the foregoing iterations has to do with the quality of the risk-mitigation task. If the risk-mitigation task is perfect, we can glean all the information from whatever functions happen to be available in that iteration. On the other hand, if such activity is not perfect, then functions which become available in the foregoing iterations can come back into the pictures and generate information several iterations later.

From the point of view of risk management, the former case is evidently preferable. To simplify the model, I assume that the risk-mitigation task is perfect, i.e. the probability of change depends only on the functions in that particular iteration

$$\Pr(\hat{I}^j = I_o) = f^j(\Delta^j; I_o)$$

Here, I assume that the probability of change is also a function of $j$, the iteration index. This happens, for example, if the risk-mitigation tasks are different in iteration $j_1$ from iteration $j_2$. To simplify our formulation, I assume that the probability function is not dependent on which iteration the change takes place, thus dropping the superscript $j$.

$$\Pr(\hat{I}^j = I_o) = f(\Delta^j; I_o)$$

In addition to that, I also assume that each interface ($q=1, \ldots, Q$) is probabilistically independent, i.e.

$$\Pr(\hat{I}^j = I_o) = \prod_{q=1}^{Q} \Pr(\hat{I}_q^j = I_{oq})$$

In other words, from the modeling viewpoint, we only need to concern with

$$\Pr(\hat{I}_q^j = 1)$$

for each $q$. This, in turn is defined as

$$\Pr(\hat{I}_q^j = 1) = f_q(\Delta_j)$$

and is called the probability function. The probability function relates the incremental function set to the probability of change of each interface.
3.4 Probability and Impact Functions

*Probability Function*

In the previous section, the probability of change for interface \( q \) in iteration \( j \) is assumed to depend on the incremental function set of iteration \( j \).

\[
\Pr[\bar{T}_q = 1] = f_q(\Lambda_j)
\]

In this section, I will probe a little deeper into the meaning of this equation.

From the formulation, interface \( q \) has some uncertainty which persists throughout the process. Such uncertainty has to do with the value of the interface \( q \) which will allow us to create the system to meet requirements of all functions in \( \Phi \).

The uncertainty reduces as we grow the system. This is due to the fact that, as we add modules to the system, more functions become available for scrutiny. Upon investigating such functions, we can assess how well interface \( q \) has been set. With such knowledge, we are in a position to adjust the value of interface \( q \) if we need to.

![Diagram of impacts of functions on change of an interface]

*Figure 3.4 Impacts of functions on the change of an interface*

The probability of change for interface \( q \) in a given iteration depends on the set of functions available for scrutiny in that particular iteration. This implies that interface \( q \) has implications on the working of those functions, i.e. that our specifications of interface \( q \) have direct impacts on whether or not those functions can be achieved.
We can visualize a Venn diagram depicting the events that the interface needs to be changed in a particular iteration. Such events correspond to the sample points in the circles. Each circle represents the change events in which interface \( q \) thus specified does not accommodate the function corresponding to the circle and therefore needs to be changed.

![Venn diagram](image)

**Figure 3.5 Venn diagram depicting events that cause interface change from two functions**

The area of the circle represents the probability that an interface change is needed to accommodate a particular function. The bigger the circle, the more dependent that function is on the interface.

When there is more than one function, there are instances in which we need to change the interface to accommodate more than one function at one time. In this picture, this corresponds to the intersection of the two circles.

In this circumstance, iteration policies play a key role in determining the probability of change.
For example, in the above picture, if each of these functions is present alone in an iteration, say iteration \( j_1 \) and \( j_2 \), the probability of change in those iterations is simply the area of each circle, thus \( P^{j_1} = A_1 \), \( P^{j_2} = A_2 \).

However, if both are present in the same iteration, say iteration \( j \), the probability of change is the area of the union of these two circles, thus, \( P^j = A_1 + A_2 - A_{12} \), which explains the equation

\[
\Pr[I_j^j = 1] = f_q(A_j)
\]

and implies that \( f_q \) can be any mapping from \( 2^\emptyset \) to \([0, 1] \).

To simplify our formulation, I make the following assumptions:

1. Corresponding with each interface \( q \) and each function \( n \), there is a probability parameter \( p_{nq} \), representing the probability of change for interface \( q \), given that only function \( n \) is present in the iteration in question. This corresponds to the area of the circle corresponding to each function and interface.

2. The probability of change for interface \( q \) in an iteration is a real function of \( p_{nq} \) for function \( n \) present in the iteration in question.

In making these two assumptions, I have parameterized the probability function as a function of probability parameters of the functions. I assume that the structure of problem is such that we can determine the effective probability of change, which is a result of several functions, from the probability of change corresponding to individual functions.

For this thesis, I define two types of probability functions for further investigation: additive functions and multiplicative functions.
Additive probability function (A)

In this case, the probability of change is simply the sum of the probability parameters.

\[
\Pr[I_q' = 1] = \sum_{n \in N} p_{nq}
\]

This corresponds to the case when functions are independent as far as the change events are concerned. In other words, a particular change event corresponds only to one individual function only.

![Venn diagram for additive probability function](image)

Figure 3.6 Venn diagram for additive probability function

Multiplicative probability function (M)

In this case, we define the probability of change as follows

\[
\Pr[I_q' = 1] = 1 - \prod_{n \in N} (1 - p_{nq})
\]

In this case, there is some overlapping among functions. The overlapping is in such a way that we can calculate the area of the unions from the probability parameters of individual functions.

![Venn diagram for multiplicative probability function](image)

Figure 3.7 Venn diagram for multiplicative probability function
**Impact Function**

In the previous section, I postulate that the amount of rework on a particular module in a particular iteration is a function of the interface change vector of the iteration.

\[ R_n^i = f_n^a (v^i) \]

As mentioned before, interfaces are those necessary inputs to the development of a module. When these interfaces change, we need to redevelop the module or perform rework on it. The exact amount of rework to be done depends on the set of interfaces that "change" in that iteration.

The dependency of a module on interfaces varies. Some interfaces are more crucial to the module and necessitate more rework when they are changed. Therefore, the impact when such interface changes is higher. In addition to that, when more than one of these interfaces change at the same time, the amount of rework to be done depends not only on the impacts from individual interfaces, but also on the relationships among such interfaces.

![Impact of interfaces on the rework of a module](image)

*Figure 3.8 Impact of interfaces on the rework of a module*
In the same spirit as the probability function, I make the following simplifying assumption to parameterize the impact function.

1. Corresponding with each interface \( q \) and each module \( m \), there is an impact parameter \( d_{qm} \), representing the amount of work needed on module \( m \), given that only interface \( q \) changes in the iteration in question.

2. The amount of rework on module \( q \) when a set of interfaces change is a real function of the impact parameters of those interfaces.

To show this, suppose we know that interface \( q_1, q_2, \) and \( q_3 \) change, the amount of rework on module \( m \) is a real function of the impact parameters of those interfaces, i.e. \( R_m = g_m(d_{mq_1}, d_{mq_2}, d_{mq_3}) \). In the next section, when we define the expected rework as the objective function, we will find that we need to obtain the amount of rework for all possible combinations of the interfaces.

In this thesis, we will only deal with the case when the impact from multiple interfaces is independent, i.e. the real function is simply the sum of the interface parameters. We can say that the impact function is of additive type. In this case, we can write the rework on a module for an iteration as

\[
\tilde{R}_m = \sum_{q=1}^{Q} d_{mq} \cdot \tilde{I}_q
\]
3.5 Risk Matrix

From the simplifying assumptions, we now have a parameterized version of the risk dynamics model. For ease of representation, I will write down the parameters of the simplified model in the following super-matrix called the risk matrix, which comprises three constituent matrices $P$ (probability parameters), $D$ (impact parameters, $D$ for delta), and $H$ (functional mapping).

Matrix $P$, on top-left corner, consists of the probability parameters for each function-interface pair. Note that the order of subscripts is such that function precedes interface. In other words, $p_{nq}$ is the probability parameter for function $n$ and interface $q$. This is to signify that function $n$ affects interface $q$, $n \rightarrow q$, i.e. create change. Note that $p_{1q} + \ldots + p_{Nq}$, the total probability of change for a particular interface, might or might not be equal to the unity.

Matrix $D$, on bottom-right corner, consists of the impact parameters for each interface-module pair. In the same spirit as matrix $P$, the order of the subscripts is such that interface precedes module. $d_{qm}$ is the impact parameter for interface $q$ and module $m$. The change in interface $q$ creates rework on module $m$.

Matrix $H$, on bottom-left corner, represents the functional mapping matrix between functions and modules. Corresponding to each function is each column of $H$, which is a zero-one vector where each element corresponds to each module of the system. If an element is zero, the module does not belong to that function. On the contrary, if an element is one, the module belongs to that function.
3.6 Work-minimization Problem

This chapter so far has explained the risk dynamics model, which is a theory of how work and rework is generated in the iterative development process. From the model, we can specify the amount of rework on the module level for a given process policy. In this section, I formulate a work-minimization problem which is the subject of investigation in the following chapters.

Objective function

To study the effect of the process policy, we need to choose a particular process measure as the objective function. For the purpose of illustrating features of the model, I simply select the total sum of risk-mitigation work and expected rework across iterations as the objective function.

This selection means that we ignore the allocation of the work to different resources (development teams, engineers, etc.) which can give rise to differences in the amount of work to be done. In other words, I assume that the model parameters such as impact parameters do not depend on how we allocate the work to various development resources.

In addition, I look at the sum of work which corresponds to development efforts rather than cycle time as our objective. This is done to simplify the analysis and enable me to illustrate various results without having a model that is too complicated. However, I will show in Section 7.3 that it is easy to extend the model so we can use the cycle time as the objective function.

Finally, the selection of the expected rework implies a risk-neutrality assumption in the management of a development process. This, again, is done to simplify our analysis. The model can be used to analyze the case when objective reflects the sensitivity to variance as well.
Simplifying assumptions

In addition to specifying the objective, I also remind the reader the following simplifying assumptions to be used in the problem.

Risk-mitigation work is fixed, \( L \) per iteration, regardless of the process policy.

Interface changes are independent.

Unless otherwise stated, I assume that the probability function is of additive form. I will only use the multiplicative form in section 4.5.

Impact function is of additive form.

Iteration work

From the selection of the objective function and the foregoing simplifications, I am now ready to formulate an optimization program for further investigation. Note that the total objective function can be written as the sum of the expected work for each iteration. I call each of these the iteration work.

\[
W(S^1, S^2, ..., S^j) = w^1 + w^2 + ... + w^j
\]

Iteration work in turn consists of the risk-mitigation work and the expected rework. The risk-mitigation work is assumed to be deterministic and of a fixed value, \( L \) per iteration. The expected rework for an iteration is the sum of the expected rework for all the modules that have been developed up to and including that iteration.

As will be seen shortly, iteration work is a function of two things: \( \hat{S} \), the set of module(s) allocated to the foregoing iteration(s) and \( S \), the set of module(s) allocated to that iteration. Therefore, without making reference to the iteration index, we can write iteration work for the given \( \hat{S} \) and \( S \) as

\[
w(\hat{S}, S) = L + \sum_{m \in \hat{S} \setminus S} E[\bar{R}_m(\hat{S}, S)]
\]
The expected rework for each module is the sum of the expected rework from each interface, which, in turn, is the product of the probability of change (of that interface, in that iteration) and the impact parameter (of the interface and the module in question).

\[ E[\mathbb{R}_m(\hat{s}, S)] = \sum_{q=1}^{Q} P_q(\hat{s}, S) \cdot d_{qm} \]

Now, the probability of change depends on the functions that are present in that iteration. In fact, it is the function of the probability parameters of all such functions present in that iteration.

\[ P_q(\hat{s}, S) = f(\{P_{mq}\}_{m \in \Delta(\hat{s}, S)}) \]

More specifically, we can write the probability of change in the case of additive probability function as

\[ P_q(\hat{s}, S) = \sum_{n=\Delta(\hat{s}, S)} P_{mq} \]

and in the case of multiplicative probability function as

\[ P_q(\hat{s}, S) = 1 - \prod_{m \in \Delta(\hat{s}, S)} (1 - P_{mq}) \]

The set of functions present in the iteration can be written as

\[ \Delta(\hat{s}, S) = \mathcal{Z}(\hat{s} \cup S) \setminus \mathcal{Z}(\hat{s}) \]

Whereas,

\[ \mathcal{Z}(\hat{s}) = \{ F_n \subseteq \hat{s} \} \]

So, in summary, we can write iteration work as

\[ w(\hat{s}, S) = \lambda + \sum_{n \in \Delta(\hat{s}, S)} d_{qm} \cdot f(\{P_{mq}\}_{m \in \Delta(\hat{s}, S) \cup \mathcal{Z}(\hat{s})}) \]
Dynamic programming formulation

There are several ways we can go from here. From the way I define the iteration work, it is easy for us to formulate the problem using the dynamic programming methodology. To write a dynamic programming formulation, I first define $\nu(\hat{s})$ as the minimum remaining expected work after we allocate $\hat{s}$ to the foregoing iterations.

$\nu(\hat{s})$ can then be written in a recursive formula as

$$\nu(\hat{s}) = \min_{S \subseteq \hat{s}} [w(\hat{s}, S) + \nu(\hat{s} \cup S)]$$

In other words, we say that $\nu(\hat{s})$ is, in fact, the minimum, among all possible allocations of modules to current iteration, $\forall S \subseteq \hat{s}$, of the sum of the iteration work of the current iteration, $w(\hat{s}, S)$, and the minimum remaining expected work after such allocation, $\nu(\hat{s} \cup S)$.

The problem is then solved when we find $\nu(\emptyset)$ (corresponds to the iteration 1, when nothing has been allocated), given that $\nu(\emptyset) = 0$, (when all modules have been allocated, the remaining expected work is zero, the boundary condition).

This formulation, however, is not easy due to the complicated nature of the term $w(\hat{s}, s)$. In the next chapter, we will see some special cases of which the special structure allows us to solve the problem using the dynamic programming in a relatively easy manner.

Fixed-iteration formulation

The other way to formulate the problem is to look at a problem when we specify the number of iterations. Using the same derivation derived above, we can also write this in a fixed-iteration formulation as follows.

$$W^J = \min_{S^1, S^2, \ldots, S^J} W(S^1, S^2, \ldots, S^J)$$

The objective for a particular allocation is given by

$$W^J(S^1, S^2, \ldots, S^J) = \sum_{j=1}^{J} w(\hat{s}^{j-1}, S^j)$$

Chapter 3: Model Formulation
To find the optimal solution to the problem, therefore, we look among the optimal solution to the fixed-iteration problem for all the number of iterations.

**Integer-programming formulation**

When we specify that the probability function is additive, the case which we will mostly talk about in the next chapter, this can be written as:

$$W^j(S_1^j, S_2^j, ..., S_J^j) = \sum_{j=1}^{J} \left( L + \sum_{m \in S_i^j} \frac{L}{q+1} \sum_{n \in S_i^{j-1}} \sum_{q=1}^{Q} p_{mq} \cdot d_{mq} \right)$$

$$= \sum_{j=1}^{J} \left( L + \sum_{q=1}^{Q} \sum_{m \in S_i^{j}} \sum_{n \in S_i^{j-1}} p_{mq} \cdot d_{mq} \right)$$

This, in turn can be written in the integer-programming formulation with quadratic objective. To formulate the problem in this way, first we need to define the decision variables for both functions and modules, as opposed to the previous cases where only the modules are decision parameters. We define these decision parameters in the form of zero-one vectors as follows.

$$y^j = (y_1^j, ..., y_n^j)$$

$$y_n^j = \begin{cases} 1; & \text{function } n \text{ is in iteration } j \\ 0; & \text{otherwise} \end{cases}$$

$$z^j = (z_1^j, ..., z_M^j)$$

$$z_m^j = \begin{cases} 1; & \text{module } m \text{ is in iteration } j \\ 0; & \text{otherwise} \end{cases}$$

The function vectors, $y^j$, are defined as row vectors, whereas the module vectors, $z^j$, are defined as column vectors for representation purpose as we will see shortly.

The comprehensive and exclusive properties of the iteration policy (i.e. that we allocate all the modules and functions and that we allocate them to only one iteration) can be written in the equation form as:

$$\sum_{j=1}^{J} y_n^j = 1 \quad \forall n$$

$$\sum_{j=1}^{J} z_m^j = 1 \quad \forall m$$
In addition, the relationships between the functions and modules can be written as,

\[ y_n^j \leq \sum_{m=1}^{l} z_m^j \quad \forall m \in F_n \]

which means that a function can be allocated to iteration \( j \) if all its constituent modules are allocated at or before iteration \( j \).

The objective, in turn, can be written as

\[
W^j \left( S^1, S^2, \ldots, S^j \right) = \sum_{j=1}^{J} \left( L + \sum_{q=1}^{Q} \sum_{m=1}^{M} \sum_{n=1}^{N} p_{mn} \cdot d_{qm} \right) \\
= L \cdot J + \sum_{j=1}^{J} \sum_{q=1}^{Q} \sum_{m=1}^{M} \sum_{n=1}^{N} \left( p_{mn} \cdot y_n^j \right) \cdot \left( d_{qm} \cdot z_m^j \right) \\
= L \cdot J + \sum_{j=1}^{J} \sum_{l=1}^{L} \sum_{j=1}^{J} y_n^j \cdot G \cdot z_l^j
\]

where \( G = P' \cdot D' \), and \( P', D' \) are defined as in section 4.

From this, we can rewrite the problem in the integer-programming form where \( J \) is the number of iterations as

\[
W^j' = L \cdot J + \min_{y^j, \ldots, y^j} \sum_{j=1}^{J} \sum_{l=1}^{L} y_n^j \cdot G \cdot z_l^j
\]

s.t. \( y_n^j \leq \sum_{m=1}^{l} z_m^j \quad \forall m \in F_n \) for \( \forall j, n \)

\( \sum_{j=1}^{J} y_n^j = 1 \quad \forall n \)

\( \sum_{j=1}^{J} z_m^j = 1 \quad \forall m \)

\( y_n^j, z_m^j \in \{0,1\} \quad \forall n, m, j \)

From the formulation in this section, we are now ready to begin our study of the interactions between the product and the process. The next chapter will discuss these issues by analyzing the work-minimization problem posed in this section.

Chapter 3: Model Formulation 65
CHAPTER 4 MODEL ANALYSIS

4.1 Introduction

The work-minimization problem, derived from the risk dynamics model in the previous chapter, serves as an abstraction for the process structuring problem mentioned earlier in this thesis. The optimal solution of the problem corresponds to the ideal process structure when the objective is the total amount of work. The characteristics of the optimality, therefore, can be used as guidance on the most effective way to structure the development process.

This chapter provides the insights into the management of a development process through the analysis of the work-minimization problem. More specifically, I will focus on the analysis of the optimal solutions for some special cases of the problem. Such analysis provides us with the characteristics of the process policy at the optimality. These characteristics in turn will help us gain several important insights into the management of a development process.

Section 4.2 develops the concept of risk efficiency through an analysis of the single-interface system. In Section 4.3, and Section 4.4, the analysis of two special classes of the single-interface system provides insights into the effects of the concentration of risks and the product architecture on a development process. Section 4.5 focuses on the multiple-interface system and provides insights into interactions of risk issues. Finally, Section 4.6 develops a tool for the design and analysis of process models and continues earlier discussions on this topic.

4.2 Risk Efficiency

Continuing from the discussions in Chapter 1, this section investigates how the evolution of risks affects the process performance. More specifically, I will show through the analysis of a single-interface system that the optimal process policy has the characteristics of risk efficiency, with respect to the probability and the impact of risks. The risk efficiency in turn is shown to directly result from the two process decisions, the task sequence and the information exchange.
Note that the focus on a single-interface system might not be as restricted as it seems. The interface in this model, as explained in the previous chapter, refers to those aspects of design that are subject to change. Therefore, a single-interface system can very well have multiple design interfaces, which is what we would expect for a realistic system. The only requirement is that only one of the interfaces is subject to change and modeled explicitly in the model. More generally, the system can even have multiple uncertain interfaces, as long as we can lump them all together into a single interface from the modeling perspective.

**Problem formulation**

The system to be studied has one interface, $N$ functions, and $M$ modules. The probability parameter for function $n$ is denoted $p_n$. The impact parameter of module $m$ is denoted $d_m$. The functional mapping parameter $h_{nm}$ is 1 if module $m$ belongs to $F_n$, the set of modules which defines function $n$, and 0 otherwise. The risk-mitigation work is a fixed parameter $L$. From the previous chapter, the system parameters can be represented in a risk matrix as in Figure 4.1.

![Figure 4.1 Risk matrix of a single-interface system](image)

I will use the fixed-iteration formulation of the work-minimization problem, written as follows.

Let $W^j(S^1, S^2, ..., S^l)$ be the expected total work for a process policy defined by $(S^1, S^2, ..., S^l)$

$$
W^j = \min_{(S^1, S^2, ..., S^l)} W^j(S^1, S^2, ..., S^l)
$$

$$
= \min_{(S^1, S^2, ..., S^l)} \sum_{j=1}^l \left[ L + \left( \sum_{n \in S^j} p_n \right) \left( \sum_{m=1}^M d_m \right) \right]
$$

$$
W^* = \min_j [W^j]
$$

The model captures the process in which the interface is updated as we grow the system. Beginning with a preliminary value of the interface at the beginning of iteration $j$, we build up a set of modules of the system, $S^j$. In that iteration, we incur the risk-mitigation work of $L$ and the expected rework, which is a product of the probability and the impact of interface change.
The probability of interface change is the sum of the probability parameters of all the functions that emerge in that iteration. The impact of interface change is the sum of the impact parameters of all the modules that have been developed up to that iteration. After this iteration, we get an updated value of the interface and proceed to repeat the process until we build the whole system.

**Risk evolution**

To get a better picture, define the following “iteration parameters” for each iteration.

\[ P^j = P(\mathcal{A}^j) = P(\mathcal{A}^j \cup (\mathcal{S}^{j-1} \cup \mathcal{S}^{j-1})) = \sum_{n \in \mathcal{S}^{j-1}} p_n \]  

“Iteration probability”

\[ D^j = D(S^j) = \sum_{m \in S^j} d_m \]  

“Iteration impact”

\[ R^j = P^j \cdot \sum_{i=1}^{j} D^i \]  

“Expected iteration rework” or “Iteration rework”

The objective can therefore be rewritten as

\[ \sum_{j=1}^{i} (L + R^j) = \sum_{j=1}^{i} \left( L + P^j \cdot \left( \sum_{k=1}^{j} D^k \right) \right) \]

In a sense, the iteration probability “takes out” the uncertainty from the process. The iteration probability generates rework only for modules which have been developed up to that iteration and not to any modules after that iteration. On the other hand, the iteration impact “adds to” the impact in the process. The iteration impact contributes to rework from its iteration up until the last iteration in the process.

From this perspective, the iteration probability and the iteration impact can therefore be thought of as the first difference (negative, for probability) of the probability of risk and the impact of risk mentioned in Chapter 1. (For clarity, the term “remaining probability” and “cumulative impact” will be used for the probability and the impact of risk to distinguish them from the iteration probability and the iteration impact.)

Since the iteration probability and the iteration impact depend on the process policy, the evolution of risks, which traces how the remaining probability and the cumulative impact change over time, also depends on the process policy. Figure 4.2 shows the evolution of the remaining probability and the cumulative impact of a particular process policy.

*Chapter 4: Model Analysis*
Two-iteration problem

To better understand the characteristics of the optimal policy, let us now investigate the case when the number of iterations is either one or two. In this case, the problem is reduced to the comparison between the optimal policy for one iteration and that for two iterations and is written as follows. ($P$ and $D$ are the sum of all probability and impact parameters, respectively)

\[
W = \min\left(W^1, W^2\right)
\]

\[
W^1 = L + P \cdot D
\]

\[
W^2 = 2L + \min_{S', S''} \left\{ P^1 \cdot D^1 + P^2 \cdot \left( D^1 + D^2 \right) \right\}
\]

The two-iteration case is preferable to the one-iteration case if and only if

\[
L \leq P \cdot D - \min_{S', S''} \left\{ P^1 \cdot D^1 + P^2 \cdot \left( D^1 + D^2 \right) \right\}
\]

The term on the right-hand side is the difference between the expected rework in the one-iteration case and the minimum expected rework in the two-iteration case. It represents the maximum expected rework that can be avoided from an additional iteration and will be called the maximum avoided rework.

From this, the decision on the number of iterations is done simply by comparing the benefits (maximum avoided rework) with the costs ($L$, risk-mitigation work) for an additional iteration.
Figure 4.3 shows the decision problem for two-iteration case. From this, we can see that the decision variables are the set of modules and the set of functions allocated to these two iterations. Because there are two iterations and functions are related to modules through the functional mapping, the essential decision parameter is only the set of modules allocated to iteration 1, $S^i$. Other decision parameters can all be derived from $S^i$ as follows.

$$S^2 = \Sigma \setminus S^i$$
$$\Delta^i = \mathcal{S}(S^i)$$
$$\Delta^2 = \mathcal{S}(S^i \cup S^2) \setminus \mathcal{S}(S^i) = \Phi \setminus \mathcal{S}(S^i)$$

Thus, the iteration probability and the iteration impact for both iterations can be derived from $S^i$.

$$p^i = \mathbf{P}(\Delta^i) = \mathbf{P}(\mathcal{S}(S^i))$$
$$D^i = \mathbf{D}(S^i)$$
$$p^2 = \mathbf{P}(\Delta^2) = \mathbf{P}(\Phi \setminus \Delta^2) = \mathbf{P}(\Phi) - \mathbf{P}(\Delta^i) = P - p^i$$
$$D^2 = \mathbf{D}(S^2) = \mathbf{D}(\Sigma \setminus S^i) = \mathbf{D}(\Sigma) - \mathbf{D}(S^i) = D - D^i$$

More importantly, the iteration probability and the iteration impact of iteration 2 can simply be written in the form of the iteration probability and the iteration impact of iteration 1. This directly results from the additivity of the probability function and the impact function (Section 3.4). As will be seen, this property will be very useful in the analysis that follows.
**Basic trade-off**

Now take a look at the expected rework of the two-iteration problem, which is a function of $S^j$.

\[
R(S^j) = R^1 + R^2
= P^1 \cdot D^1 + P^2 \cdot (D^j + D^2)
= P^1 \cdot D^1 + (P - P^1) \cdot D
\]

Note that $D^j$ appears in the expressions for both $R^1$ and $R^2$. If the interface changes in iteration 1, the amount of rework is $D^j$, which corresponds to modules allocated to iteration 1. If it happens in iteration 2, the amount of rework includes both $D^j$ and $D^i$. In other words, the impact is cumulative and $D^j$ represents the amount of impact that is added to the process in iteration 1, permanently until the end. Everything else being equal, we therefore would like to minimize $D^j$ to result in lower $R$.

On the other hand, $P^1$ contributes to rework only in $R^1$ while its effect is taken out in $R^2$. The probability of interface change in iteration 1 is $P^1$ while that in iteration 2 is $P - P^1$. In other words, $P^1$ represents the amount of probability that is taken out of the process in the form of the probability of change. Everything being equal, therefore, we would like to maximize $P^1$ to result in lower $R$.

To better see this, the expression of rework is rewritten as follows.

\[
R = P^1 \cdot D^1 + (P - P^1) \cdot D
= P \cdot D - P^1 \cdot (D - D^1)
\]

In this expression, $P(D-D^j)$ represents the avoided rework mentioned before. To minimize $R$, we simply need to maximize $P(D-D^j)$. Since both terms are functions of $S^j$, the optimal policy is obtained by choosing $S^j$ in such a way that the objective is minimized.

From this expression, the essence of the problem has to do with the basic trade-off between $P^1$ and $D^1$. More specifically, to maximize $P(D-D^j)$, what we need to do is to choose $S^j$ to maximize $P^1$ while minimizing $D^1$. Because $P^1$ represents the probability that is taken out of the process while $D^1$ represents the impact that is added to the process, we can state that the optimal process policy is found by prioritizing the modules (finding $S^j$, which is built first, thus having a higher priority) to take the most probability out while adding the least impact to the process.
Risk-efficient frontier

To better illustrate the issue, I will also show how to solve this problem using a geometrical method. From the previous discussions, a process policy is defined by $S^j$, the modules allocated to iteration 1. Corresponding to a given $S^j$, we can calculate $D^j$ and $P^j$, the iteration impact and the iteration probability for iteration 1. Therefore, we can plot a point $(D^j, P^j)$ on a chart for any given $S^j$ as shown in Figure 4.4.

The objective function, $R$, can be written as a function of $D^j$ and $P^j$. Geometrically, the objective function is equal to the sum of area of the two rectangles corresponding to $R^j$ and $R^2$ as shown below. The area of the shaded region represents the avoided rework. The problem, therefore, is to find a point on $D^j$-$P^j$ chart which minimizes the sum of the area of the two rectangles or, equivalently, maximizes the area of the shaded region.

![Figure 4.4 Rework and avoided rework](image)

For a given set of system parameters, we can plot $(D^j, P^j)$ for all possible process policies as in Figure 4.5. Corresponding to each point on this plot is a rectangle whose area corresponds to the avoided rework. The problem is then reduced to finding the point among such points which minimizes the rectangular area.
Now, let us take a look at the problem from a different perspective. Because $R$ is a function of $P'$ and $D'$, we can also find all the pairs of $(D', P')$ which gives rise to the same value of $R$. Corresponding to the set of all such pairs is a curve with the same value of $R$ and the same area of the rectangle as shown in Figure 4.6.

Because all the points on this curve yield the same expected rework, $R$, the curve will be called an iso-rework curve. From the formula of the expected rework, this curve is simply a hyperbola with the origin at $(D, 0)$. The further away the curve is from its origin, the bigger the rectangular area and the lower the value of $R$, the expected rework.
Geometrically, therefore, the two-iteration problem can be solved by finding the outer-most iso-rework curve which touches one of the points that correspond to the feasible process policies. This outer-most iso-rework curve corresponds to the smallest $R$ that can be obtained among all the feasible process policies. The optimal process policy in turn corresponds to the point which touches this curve.

Due to the shape of the iso-rework curve, this point can also be found as the intersection between the outer-most iso-rework curve and the outer envelope of the region defined by all the feasible points. This outer envelope is called the **risk-efficient frontier** because the points on this frontier are risk efficient. They all yield the highest value of $P^i$ for a given $D^i$.

![Figure 4.7 Optimal policy at the intersection of iso-rework curve and risk-efficient frontier](image)

**Portfolio-optimization analogy**

In fact, this type of analysis parallels those found in the portfolio optimization problem in finance. In that context, we are given a set of financial securities with the expectation and the covariance of returns. Define a portfolio by the weight parameters which represent the proportions of the securities in that portfolio.

The weight parameters combine the expectation and the covariance of returns for individual securities to the expectation and the variance of returns for the portfolio. We are given a utility function, which is a function of the expectation and the variance of returns for the portfolio. The problem is to find the set of weight parameters that maximizes the utility function.
The solution is obtained by finding the intersection between the outer-most iso-utility curve and the risk-efficient frontier, in this case with respect to the expectation and the variance of returns.

The same approach is used here to solve the two-iteration work-minimizing problem. The iso-utility curve and the iso-rework curve represent the indifference curve which yields a particular value of the objective function. The risk-efficient frontiers in both cases represent the points with the best trade-off between two parameters that contribute to the objective function: the expectation and the covariance; and the probability and the impact.

The solution is obtained by finding the intersection point between the outer-most indifference curve and the risk-efficient frontier. In the portfolio optimization, this point corresponds to the portfolio of securities which yields the highest utility. In the work-minimization problem, the point corresponds to the "portfolio" of modules which yield the lowest rework.

**Risk efficiency**

Risk efficiency denotes a special characteristic of the evolution of risks. A process is risk efficient if the probability of risk (remaining probability) reduces quickly while the impact of risk (cumulative impact) increases slowly. In this model, the risk evolution depends on the process policy, which specifies how to prioritize the tasks to be undertaken and the amount of information exchange in the process.

The optimal process policy of the work-minimization problem for a single-interface system is shown to have the characteristic of risk efficiency. First, the optimal policy sequences the tasks to take out high probability while adding low impact to the process. Second, the optimal policy determines the optimal number of iterations through the trade-off between the benefits and the costs of information exchange in terms of the avoided rework and the risk-mitigation work.

In this regard, the concept of risk efficiency can provide a guideline for the management of a development process. The optimal process policy can be obtained by sequencing the tasks in the right order and having enough information exchange to ensure that risks in the process evolve in the most efficient manner.
4.3 Risk Concentration

This section expands the analysis of a work-minimization problem to the case when no constraints on the number of iterations exist. To derive insightful results, the focus will be on the single-interface system with a special functional mapping called a modular functional mapping.

The analytical results show that we can extend the concept of risk efficiency from the level of iterations, as shown in the previous section, to the level of modules. The structure of the optimal solution is shown to be dictated by the ratio of probability to impact parameter for individual modules. In addition, this leads to a finding related to the effects of risk concentration, an architectural characteristic of a system, on the total expected work, a process performance.

Modular functional mapping

This system under study in this section has a functional mapping (Section 3.3) which will be called a modular functional mapping. For such a functional mapping, there is a one-to-one correspondence between functions and modules. Each module implements a specific function. No two functions share a common module.

The name modular functional mapping is motivated from the definition of a modular architecture as that which maps a functional behavior to a specific physical element [Ulrich & Eppinger (2000)]. One can think of a Swiss-army knife as one of example of a system which has modules that correspond to independent functions. This is in contrast with an integral architecture where a module can involve several functions.

In the literature, a modular architecture enables the modules to be developed independently relative to an integral architecture. This is due to the underlying assumption in the literature that the design specifications that give rise to such a modular architecture remain unchanged throughout the process. This implies that each module is completely independent of each other.

In our model, however, the interfaces, which represent specifications, assumptions, or inputs, are subject to change throughout the process. This means that, even though modules are independent as far as their functions are concerned, they depend on each other through their common dependencies on the interface. The planning and the execution of the development process for the system, therefore, have to be done with this consideration in mind.
Figure 4.9 A single-interface system with a modular mapping

Figure 4.9 shows a picture of a single-interface system with a modular functional mapping. Note the dependencies of functions and modules on the interface. Also observe that, in this case, since no two functions share a common module, there is no hierarchy of functions or modules. All modules and functions can thus be considered to be on the same level.

**Problem formulation**

The system to be studied has the following properties. There are $N$ functions and $N$ modules. The modules and functions are labeled in such a way that function $n$ corresponds to modules $n$. In this case, the functional mapping matrix, $H$, becomes an identity matrix.

The risk-mitigation work is $L$ per iteration. Functions have probability parameters given by $p_1$, $p_2$, ..., $p_N$. At this point, I also make a simplifying assumption that all the modules have the same impact parameter of $d$. Later in the section, modules will be allowed to have varying impact parameter $d_n$.

From this specification, the model parameters consist of $L$, $d$, $p_1$, ..., $p_N$. The risk matrix can be represented as in Figure 4.10.

![Risk matrix table]

**Figure 4.10 Risk matrix of a single-interface system with a modular functional mapping**
Because of the modular functional mapping, the allocation of a module to a particular iteration automatically puts the corresponding function in the incremental function set (Section 3.3, "Function buildup") of that iteration. The probability parameter of this function in turn contributes to the iteration probability as a term in a summation.

In this case, we can think of the probability parameter as being associated directly with the module that belongs to the function. Both the iteration probability and the iteration impact can then be written in terms of the module parameters as shown in the fixed-iteration formulation below. This simplifies the formulation and makes the problem much easier to analyze.

\[
W^{*} = L \cdot J + \min_{(s^{1},s^{2},...s^{l})} \sum_{i=1}^{l} \left( \sum_{n \in S^{i}} p_{n} \right) \left( \sum_{m \in S^{i}} d_{m} \right)
\]

\[
= L \cdot J + \min_{(s^{1},s^{2},...s^{l})} \sum_{i=1}^{l} p^{i} \cdot \left( \sum_{j=1}^{l} D^{j} \right)
\]

where \(p^{i}\) and \(D^{j}\) are defined as

\[
p^{i} = \sum_{n \in S^{i}} p_{n}
\]

\[
D^{j} = \sum_{m \in S^{j}} d_{m}
\]

**Optimal sequence**

The previous section shows that the optimal policy for a two-iteration problem has the characteristics of risk efficiency. The candidate solution is shown to correspond with the risk-efficient frontier, which in turn consists of points that have the highest iteration probability \(p^{i}\) for a given iteration impact \(D^{j}\). More generally, this frontier represents the process policies which, at any given point in the process, take out the most probability for a given cumulative impact.

When the probability and the impact parameters are both the parameters of a module, one could expect that such risk efficiency might be reflected at the level of individual modules. In other words, to obtain the risk efficiency in the development process, the ratio of the probability and the impact parameter of a module should play a key role in determining the allocation of that module to one of the iterations in the process.
In the case when the impact parameters are all identical, the probability-impact ratio of the modules do in fact determine the module sequence in the optimal solution. Theorem 4.1 shows this result.

**Theorem 4.1:** In the optimal policy, let \( p_{\text{min}}^j \) and \( p_{\text{max}}^j \) be the minimum and the maximum probability parameters, respectively, among those of modules allocated to iteration \( j \). Then,

\[
\frac{p_{\text{min}}^j}{d} \geq \frac{p_{\text{max}}^{j+1}}{d}
\]

**Proof:**

Proof by contradiction. First, suppose that the optimal number of iterations is \( J \). Recognize that the objective function can be written as

\[
J \cdot L + \sum_{j=1}^{J} \left( \sum_{n \in S^j} p_n \right) \left( \sum_{n \in S^j} d_n \right)
\]

\[
= J \cdot L + \left( \sum_{n \in S^1} p_n \right) \left( \sum_{n \in S^1} d_n \right) + \ldots + \left( \sum_{n \in S^{j+1}} p_n \right) \left( \sum_{n \in S^{j+1}} d_n \right) + \ldots + \left( \sum_{n \in S^J} p_n \right) \left( \sum_{n \in S^J} d_n \right)
\]

Now, assume that \( p_{\text{min}}^j < p_{\text{max}}^{j+1} \) in an optimal solution for some \( j \).

Then, create another solution by swapping the two modules corresponding to these two probability parameters. All the other terms except the terms corresponding to iteration \( j \) and iteration \( j+1 \) remain the same. The iteration impacts for these two iterations also remain the same since the impact parameters are all identical. Only the iteration probabilities of these two iterations change by the swapping.

Next, subtract the new objective from the original one to obtain \( \left( p_{\text{max}}^{j+1} - p_{\text{min}}^j \right) \cdot D^{j+1} \), a positive number.

This means that the new objective is lower than the old objective, which contradicts the hypothesis that the original solution is optimal. Therefore, \( p_{\text{min}}^j \geq p_{\text{max}}^{j+1} \) and hence the result.

Theorem 4.1 shows that, in the optimal policy, a module with a higher probability-impact ratio is allocated to an iteration no later than those of modules with lower ratio. In other words, at the optimality, there modules satisfy a precedent relationship (with respect to iterations) based on the probability-impact ratio, which in turn can be thought of as a measure of risk efficiency at the level of module.
Risk efficiency at the module level

In the previous section, the analysis for a general functional mapping shows us that the optimal policy calls for the risk efficiency at the level of the iteration. The optimal solution is shown to lie on the risk-efficient frontier. This frontier corresponds to the best trade-off of risk in a particular iteration with the most reduction in the probability for a given increase in the impact.

The analysis in this section shows that the concept of risk efficiency can be extended to the level of individual modules for a modular functional mapping. In this case, the optimal policy calls for a sequence of modules based on the ratio of the probability and the impact parameters. Modules with high probability-impact ratio have higher risk efficiency. To obtain the risk efficiency at the process level, such modules should be taken care of early on as shown in Theorem 4.1.

This also corresponds to the rule of thumb in risk management which recommends that we should take care of the tough issues, which are most likely to cause changes, early on in the process. By doing so, we force changes to happen early on; thereby avoiding the costly consequences if such changes happen later. In this model, modules with higher probability-impact ratio represent such tough issues. Such modules should be allocated to earlier iterations so that we obtain a high reduction in the probability for a given increase in the impact.

Solution algorithm

From a computational standpoint, Theorem 4.1 also helps us reduce the complexity of the work-minimization problem and leads to a relatively simple algorithm. Instead of solving the allocation problem with a huge solution space, we can solve this problem using a much simpler procedure by exploiting the optimal sequence.

First, we sort the modules by the probability-impact ratio in a descending order. After the sorting, we get the module sequence $n_1, n_2, \ldots, n_N$ with decreasing probability-impact ratio. Because the optimal solution satisfies the precedence constraints implied by this sequence, we only need to determine how to split this sequence into iterations to minimize the objective.

Figure 4.11 shows the iteration policy that results from a specific set of splitting points. In this picture, there are two splitting points, one after module $n_i$, the other after module $n_j$. This corresponds to the process policy in which we allocate modules $n_1$ through $n_i$ to iteration 1, modules $n_{i+1}$ through $n_j$ to iteration 2, and modules $n_{j+1}$ through $n_N$ to iteration 3.
This sequence splitting problem, in turn, could be translated easily into a shortest-path problem. To see this, observe that the objective is the sum of the risk-mitigation work and the expected rework of individual iterations. The sum of the risk-mitigation work and the expected rework for an iteration with modules $n_{a+1}$ through $n_b$, $a, b \in \{0, 1, \ldots, N\}$, $a < b$, can be written as

$$W_{a,b} = L + \left( \sum_{n_{a+1}}^{b} p_{n_i} \right) \cdot b \cdot d$$

Notice that this is a function of $a$ and $b$, the two splitting points that mark the iteration in question. This means that we can create an arc between $a$ and $b$ to represent an iteration which includes modules $n_{a+1}$ through $n_b$. The length of the arc is $W_{a,b}$, the sum of the risk-mitigation work and the expected rework for such iteration.

For a particular set of $J-1$ splitting point (marking $J$ iterations) that is located after module $a_1, a_2, \ldots, a_{J-1}$ in the sequence, the objective can be written as

$$W(a_1, a_2, \ldots, a_{J-1}) = W_{0,a_1} + W_{a_1,a_2} + \ldots + W_{a_{J-1},N}$$

This is simply the sum of the length of arcs along a path which starts from the first dummy node (node 0) and ends on the last node (node $N$) in the sequence. The problem is then translated into that of finding the shortest path among all possible paths from node 0 to node $N$.

Let us recap our formulation of this problem again. We show that the problem can be cast in terms of a shortest-path problem. The network in question involves $N+1$ nodes from 0 to $N$ where node 1, … $N$ represents the splitting point that is after module $n_1$ to $n_N$. Node 0 is a dummy node that is located before module 1.
The length of the arc between node \( a \) and \( b \), \( W_{a,b} \), is defined as

\[
W_{a,b} = L + \left( \sum_{i=0}^{b} p_{ni} \right) \cdot b \cdot d
\]

The problem of choosing splitting points with the minimum amount of work is mapped to the problem of finding the shortest path from node 0 to node \( N \).

![Figure 4.12 Equivalent network of work-minimization problem](image)

The sum of the lengths of the arcs along the path represents the objective we are trying to optimize, the sum of the risk-mitigation work and the expected rework. The shortest path corresponds to the optimal process policy whereas the length of the shortest path from node 0 to node \( N \) is equal to the optimal objective.

Note that according to this formulation, the network is acyclic with positive lengths. An arc exists between any given pair of nodes where \( a < b \). The optimal sequence helps reduce the original problem to a shortest-path problem which is relatively much easier to solve.

**Numerical examples**

Due to the optimal sequence based on the probability-impact ratio, the iteration parameters of the optimal policy for this type of system also exhibit some very interesting characteristics. To provide a motivation for this finding, we will now look at a numerical example for a work-minimization problem of a single-interface system with a modular functional mapping.

The system for this example has 100 functions and 100 modules and \( d = 1 \). \( L = 1, \ d = 1 \). The probability parameters are randomly generated and sum to one. The shortest-path algorithm is used to solve the work-minimization problem. Figure 4.13 shows the optimal results for this problem. The chart depicts, at optimality, the evolution of the iteration probability, the iteration impact, the iteration rework, and the ratio of iteration probability and impact.
The chart shows a very interesting evolution pattern of the iteration parameters. In this particular example, the iteration probability decreases in iterations. The iteration impact, on the other hand, increases as the process unfolds. From this, the ratio of the iteration probability and the iteration impact also decreases in time.

These trends show that the risk efficiency characteristics are very strong. In this case, not only does the probability-impact ratio, which serves as a metric for risk efficiency, is decreasing, but the iteration probability itself is also decreasing while the iteration impact is increasing. In other words, the risk efficiency here is extended to the "absolute" value of the risk measures themselves, not just their ratio.

Before discussing the proof, let us now turn our attention towards the risk-efficient frontier, which will give us a different perspective also provide an intuition to explain these trends.

**Risk-efficient frontier**

Figure 4.14 shows the risk-efficient frontier for a modular single-interface system. Here, the subtracted (taken out from the process) probability and the cumulative impact, generalize $P^j$ and $D^j$ of the two-iteration problem. Each point on the frontier corresponds to $\hat{S}^j$, the cumulative set of modules at iteration $j$. Since there is no constraint on the number of iterations, a set of points on the frontier defines a process policy. The area of each rectangle corresponds to the expected rework for each iteration. The optimal policy is obtained by finding the set of points along the frontier which maximize the shaded area which represent the avoided rework.
Figure 4.14 Risk-efficient frontier of a single-interface system with a modular functional mapping

The unique characteristic of the frontier for a modular functional mapping is the concavity. This is due to the properties of the optimal sequence which put modules with higher probability-impact ratio early on in the process. This makes the slope of the frontier higher early on and decreasing as we move on in the process, thus the concavity of the risk-efficient frontier. We can see that this is responsible for the trends in the iteration parameters as discussed earlier.

**Iteration parameters**

The following theorems provide a formal proof for the trends of iteration parameters.

**Theorem 4.2:** In the optimal partitioning, \( \frac{P_i}{D_i} \geq \frac{P_i^{i+1}}{D_i^{i+1}} \)

**Proof:**

Proof by contradiction using the swapping argument. First, assume that in an optimal policy, \( P_i/D_i < P_i^{i+1}/D_i^{i+1} \).

Next, swap the modules in these two iterations. Subtract the objective of the resulting partitioning to that of the original one to obtain \( P_i^{i+1} \cdot D_i - P_i \cdot D_i^{i+1} \), a positive quantity.

This contradicts our assumption that the original partitioning is optimal. Therefore, in an optimal solution, \( P_i/D_i \geq P_i^{i+1}/D_i^{i+1} \).
**Theorem 4.3**: In the optimal policy, the following conditions are true.

1. \( D^j - D^{j-1} \leq d \)
2. \( p^j - p^{j-1} \geq -p_{\text{max}}^{j-1} \)

**Proof:**

(1) We first prove that in an optimal partitioning,

\[
d/p_m \leq (d + D^{j-1})/p^j \quad \text{for} \ m \in S^j
\]

This can be proved by contradiction. First, assume that the opposite of the above condition is true in an optimal partitioning. Then, we consider the difference in objective between the supposedly optimal partitioning and the one in which \( m \) is moved from iteration \( j \) to iteration \( j+1 \), which, due to the opposite conditioned, is positive, thus contradicting the optimality assumption.

Therefore, the condition has to be true for an optimal partitioning. Then, we prove that

\[
D^j/p^j \leq d/p_{\text{max}}^{j-1} \leq (d + D^{j-1})/p^j
\]

The first inequality is due to the fact that the iteration impact and iteration probability are simply the sum of the individual parameters of all modules/functions in the iteration. The second inequality comes from the condition we just proved above. Hence, we obtain the result \( D^j - D^{j-1} \leq d \).

(2) The second condition can be proved in the same fashion. We first prove that

\[
(p^j + p_m)/D^{j-1} \geq p_m/d \quad \text{for} \ m \in S^{j-1}
\]

This can be proved using a similar line of argument. Then, we get that

\[
(p^j + p_{\text{max}})/D^{j-1} \geq [p/d]_{\text{max}} \geq p^{j-1}/D^{j-1}
\]

Hence, the property \( p^j - p^{j-1} \geq -p_{\text{max}}^{j-1} \).
General impact parameters

So far, the focus of this section is on the system which has identical impact parameters. In such cases, the optimal sequence of the modules is solely determined by the probability-impact parameters. In other words, the risk efficiency argument was shown to be applied at the module level.

Lemmas 4.1, 4.2 and Theorem 4.4 establish similar results in a more general setting where impact parameters of modules are allowed to be different.

**Lemma 4.1:** Consider the optimal solution for a general case when impact parameters are not necessarily identical. Let \( x \) be any module in iteration \( j \), then, the following condition is satisfied:

\[
\frac{p_x}{d_x} \geq \frac{P_j}{d_x + D_j}
\]

**Proof:** Prove by contradiction. Assume that in an optimal solution, this is not true. We find that the difference in objective function between the case when module \( x \) is put in iteration \( j+1 \) and the original case can be written as 

\[-P_j d_x + p_x (d_x + D_{j+1})\]. If the condition is untrue, this term becomes negative. This contradicts our assumption about the optimality. Therefore, the condition given above has to be true for the optimal solution. \(\square\)

**Lemma 4.2:** Consider the optimal solution for a general case when impact parameters are not necessarily identical. For a particular iteration \( j+1 \), let \( w \) be any module in iteration \( j+1 \). Then, the following condition is satisfied:

\[
\frac{P_j + P_w}{D_{j+1}} \geq \frac{P_w}{d_w}
\]

**Proof:** Prove by contradiction. Assume that in an optimal solution, this is not true. We find that the difference in objective function between the case when module \( x \) is put in iteration \( j+1 \) and the original case can be written as 

\[(P_j + P_w)d_w - P_w D_{j+1}\]. If the condition is untrue, this term becomes negative. This contradicts our assumption about the optimality. Therefore, the condition given above has to be true for the optimal solution. \(\square\)
From Lemma 2.1 and 2.2, we can then state the result for the more general settings as follows.

**Theorem 4.4:** In the optimal policy for additive probability function, let $x$ be any module with the lowest probability/impact ratio in iteration $j$ and $w$ be any module with the highest probability/impact ratio in iteration $j+1$. Then, the following conditions are true.

1. \[ \frac{p_x}{d_x} - \frac{p_w}{d_w} \geq \frac{p_x + p_w}{D_{j+1}} \]

2. \[ \frac{d_w}{p_w} - \frac{d_x}{p_x} \geq \frac{d_x + d_w}{D_{j+1}} \]

**Proof:** Using Lemma 2.1 and Lemma 2.2, we can rearrange the term to get both conditions.

This shows that the difference between the probability-impact ratios of modules in successive iterations is shown to be bounded by a real number. The fact that this real number can differ from zero means that we cannot make a strong statement about whether the optimal sequence exists like the case of identical impact parameters.

In other words, Theorem 4.4 shows that, in the case when impact parameters are not identical, the risk efficiency argument still applies, albeit to a lesser extent. The relationships between probability-impact ratio of modules in successive iterations show that the probability-impact ratio of a module still play some roles in determining the way we allocate modules to iterations in the optimal solution.
Risk concentration

The concentration of risks in a system is perceived as a particularly important architectural decision in the management of a development process. Smith and Reinertsen (1998) argues that the concentration of risks in certain parts of a system is beneficial for a number of reasons. Risk concentration reduces the multiplicative effects of spreading risks in various modules. It also helps reduce the communication efforts as a small number of talented people can be assigned to the risky modules. In addition, the monitoring can be done easier when risks are concentrated.

To understand this issue from a modeling viewpoint, let us consider two modular single-interface systems DIS and CON, which represent the systems with distributed and concentrated risks, respectively. Both systems have $N$ functions, $N$ modules, risk-mitigation work $L$, and identical impact parameters, $d$. Assume that the sum of the probability parameters are the same for both systems, i.e. $Np = p_1 + \ldots + p_N = P$ to ensure that these two systems have equal amount of uncertainty.

![Figure 4.15 Risk matrix](image)

System DIS has probability parameters that are identical across the functions, i.e. it has distributed risks. System CON, on the other hand, has varying probability parameters, i.e. its risks are concentrated on the modules with high probability parameters. From the given parameters, we can find the optimal policies and the optimal objectives for both systems.

The optimal objectives in this case are the minimum total work that can be achieved for these two systems. They represent the best process performance that can be obtained, given the risk profiles of two systems. Since these systems have the same amount of uncertainty, the comparison of their objectives and optimal solutions will help us understand the effects of risk concentration on the process management.

Chapter 4: Model Analysis
Theorem 4.5 states that the optimal objective of system CON is always lower than that of system DIS. In other words, a system with concentrated risks will always yield a better process performance compared to a system with distributed risks.

**Theorem 4.5:** The optimal expected total work of system DIS is equal to or higher than the optimal expected total work of system CON.

**Proof:**

To find the optimal policy, we simply need to sort the function from highest probability parameters to lowest parameters and then determine how to split the sequence of modules into iterations.

Let \( n_1, n_2, \ldots, n_b \) be the index of modules of system CON after such sequencing. Note that since all probability parameters are identical, the sequence of system DIS does not matter.

First, we state without proof that \( P_{DIS}^b \equiv \sum_{j=1}^{b} p_{n_i}^{DIS} \leq \sum_{j=1}^{b} p_{n_i}^{CON} \equiv P_{CON}^b \), which directly results from such sorting.

Now, for a given splitting policy, represented by splitting point \( (n_1, n_2, \ldots, n_{j-1}) \), we can write the expected rework of both systems as

\[
R_{DIS} = P_{DIS}(n_1) \cdot n_1 d + (P_{DIS}(n_2) - P_{DIS}(n_1)) \cdot n_2 d + \ldots + (P - P_{DIS}(n_{j-1})) \cdot N d \\
= -P_{DIS}(n_1) \cdot (n_2 - n_1) d + P_{DIS}(n_2) \cdot (n_3 - n_2) d - \ldots - P_{DIS}(n_{j-1}) \cdot (N - n_{j-1}) d + P \cdot N d
\]

In the same fashion,

\[
R_{CON} = -P_{CON}(n_1) \cdot (n_2 - n_1) d - P_{CON}(n_2) \cdot (n_3 - n_2) d - \ldots - P_{CON}(n_{j-1}) \cdot (N - n_{j-1}) d + P \cdot N d
\]

From \( P_{DIS}(b) \leq P_{CON}(b) \), we get that \( R_{DIS} \geq R_{CON} \).

Therefore, we get that the rework for system DIS is equal to or higher than the rework for system CON. \( \Box \)

In a modular system, modules with high probability-impact ratio are sequenced early on in the process. When the risks are highly concentrated, i.e. when only a few modules have high probability-impact ratio, we only need to work on a small number of modules in order to take out a significant amount of uncertainty from the process. The concentration of risks provides us with a leverage from the point of view of risk management.

In other words, Theorem 4.5 confirms the result in the literature concerning the benefits of risk concentration in a development process. The more concentrated the risks are, the more risk efficient the process becomes and the better the process performance.
4.4 Product Architecture

The previous section shows how the probability parameters and impact parameters affect the optimal policy. By focusing on a modular system, the functional mapping is temporarily left out of the picture. This section takes what is left off to show how the functional mapping can play a key role in determining the structure of the optimal process and the process performance. To do this, I focus the analysis on another special type of a single-interface system, that with a nested functional mapping.

*Nested Functional Mapping*

The next system to be considered is that of a nested functional mapping. In such a mapping, the sets of modules $F_1, \ldots, F_N$, that correspond to functions represent an increasing series of sets. In other words, the set of modules that correspond to a function is a subset of the set of modules that correspond to another function. To simplify the analysis, assume that there are $N$ functions, $N$ modules and that they are labeled in such a way that $F_n \subseteq F_{n+1} = F_n \cup \{n-1\}$.

This type of mapping represents the other extreme of the spectrum when compared to modular mapping. Whereas in the previous case, no two functions share the same module, in this case, every pair of functions shares some modules.

This could be considered of an integrated architecture as appeared in the literature [Ulrich and Eppinger (2000)]. In such an architecture, a module is supposed to contribute to multiple functions. Viewed in another way, two functions might have some modules in common. Such dependencies are perceived in the literature as the source of complexity in the management of the development process. But at the same time, they can also give rise to a superior technical performance compared to the modular architecture.

![Figure 4.16 System with nested mapping](image)

*Chapter 4: Model Analysis*
In this case, we can make some statements regarding these functions. In the modular mapping, all functions can be thought of as being on the same level as they do not share any modules. In the nested mapping, however, we can arrange functions in a hierarchy which depends on the number of modules they consist of.

Figure 4.17: Hierarchy of Functions in a Nested Functional Mapping
In a nested mapping, a function with a particular number of modules can be thought of as the combination of another function with smaller number of modules with a set of additional modules. Figure 4.17 illustrates this structure. From this picture, we can see that function \( n \) can be thought of as function \( n-1 \) plus an additional module \( n \).

In this fashion, we can organize functions in levels. \( F_1 \) is at level 1, the bottom of the rung, as its only module belongs to all other functions. \( F_2 \), which is \( F_1 \) plus \( M_2 \) is at level 2 above \( F_1 \), and so on. Then, \( F_N \), which consists of all the modules of the system, is at the highest level.

In this fashion, lower-level function is considered as a subsystem of the higher-level function. The analogy of this is in a computer system where hardware sits at the bottom, on top of which lies the operating systems, system libraries, and so on, with the highest level being the applications.

**Problem formulation**

The system we will focus our study on in this section has the following properties. There are \( N \) functions and \( N \) modules. The modules and functions are labeled in such a way that function \( n \) consists of module \( n \) and all modules of function \( n-1 \). In this case, the functional mapping matrix, \( H \), becomes an upper-triangular matrix.

Risk-mitigation work is \( L \) unit per iteration. Functions have probability parameters given by \( p_1, p_2, \ldots, p_N \). Modules have impact parameters given by \( d_1, d_2, \ldots, d_N \). From our specification, the model parameters therefore consist of \( L, d, p_1, \ldots, p_N, d_1, d_2, \ldots, d_N \).

The system matrix is shown in Figure 4.18.

![Figure 4.18 Risk matrix of a single-interface system with nested mapping](image)
From the structure of the functional mapping, we obtain the first result in Lemma 4.3

\[ \mathfrak{S}(\hat{S}) = \{ 1, 2, \ldots, n - 1 \} \cup \left\{ n = \min_{m \in \{1, 2, \ldots, n-1\}} \left[ m \not\in \hat{S} \right] \right\} \]

**Proof:**

We begin with the definition \( \mathfrak{S}(\hat{S}) = \{ p | F_n \subseteq \hat{S} \} \). Then, from our specifications, \( F_n \subseteq F_{n-1} \), which implies that

\[ n \not\in \mathfrak{S}(\hat{S}) \rightarrow n + 1 \not\in \mathfrak{S}(\hat{S}) \]

Therefore, \( \mathfrak{S}(\hat{S}) \) is in the form \( \{1, 2, \ldots, m\} \) where \( F_m \subseteq \hat{S} \) and \( F_{m+1} \not\subseteq \hat{S} \).

But this, in turn, implies that \( \{1, 2, \ldots, m\} \subseteq \hat{S} \), \( m + 1 \not\subseteq \hat{S} \). In other words, \( m + 1 = \min_{m \in \hat{S}} \left[ m \not\in \hat{S} \right] \), hence the result we try to obtain.

From Lemma 4.3, the fixed-iteration formulation reads

\[ W' = L \cdot J + \min_{(\hat{S}', \hat{S}'', \ldots, \hat{S}''')} \sum_{j=1}^{J} \left( \sum_{n \in \{ \min_{m \in \hat{S}'''} \ldots \min_{m \in \hat{S}'''}} \right) \cdot \left( \sum_{m \in \hat{S}'''} d_m \right) \]

**Optimal sequence**

We see from this formulation that there is likelihood that a particular allocation makes some of the iteration probability become zero. This is unfavorable from the risk-management point of view as it means that we increase impact without reduce the probability of change.

Similar to the modular mapping, it turns out that the optimal policy calls for a certain kind of sequence. When we allocate modules out of sequence, the result is a sub-optimal performance which is exhibited by zero iteration probability as mentioned. Theorem 4.6 proves this result in a rigorous manner.
**Theorem 4.6:** In the optimal partitioning of modules, \( m \in \hat{S}^j \) implies that \( \{1, 2, ..., m\} \subseteq \hat{S}^j \)

**Proof:**

Proof by contradiction. First, in an optimal partitioning, suppose \( m \in \hat{S}^j \).

Let \( m_0 < m \) be the module with the smallest index which is absent from \( \hat{S}^j \). Then, we have \( \hat{S}^j \setminus \{m_0, m_0 + 1, ..., m\} = \emptyset \) due to the nested structure of the functional mapping. Assume that all modules have non-zero impact parameters, we can improve the objective by moving all modules whose index falls anywhere from \( m_0 + 1 \) to \( m \) to iteration \( j + 1 \).

To see this, first notice that such movement does not change the iteration probability of any iteration before \( j + 1 \) because such probability can only be the sum of probability parameters of functions whose index is less than \( m_0 \). This is due to our assumption that \( m_0 \) is not in \( \hat{S}^j \) and to the nested structure of the functional mapping.

Next, the movement reduces the iteration impact for those iterations to which the modules that are moved belong. On the other hand, the movement does not change anything from iteration \( j + 1 \) onward.

This therefore results in a decrease of the objective, contradicting the assumption of optimality. Hence, we obtain the above result.

Compared to the modular functional mapping, where the functional mapping is trivial, in this case we see the overriding importance of the functional mapping, or more precisely, of the functional hierarchy, in determining the sequence of the modules.

In particular, we can see that regardless of the probability parameters and impact parameters, modules which belong to the lower-level function are always sequenced first to enable other higher-level functions to be available upon risk-mitigation in future iterations.

To see this, note that in the risk dynamics model, a function is available for testing only if all its modules are present in that iteration. Without the presence of all modules that belong to a function, we are in no position to engage in testing, validation activities which provide us with knowledge to reduce the uncertainty in the process.

This is a hard constraint which manifests itself clearly in this nested mapping, when functions are hierarchical. When a module is placed out of sequence, it renders all the functions to which it belongs unavailable for risk-mitigation. This means that we increase impact as we add modules to the system, but we learn nothing from such activities.

*Chapter 4: Model Analysis*
These two cases of modular mapping and nested mapping again illustrates our guiding principle of the risk-efficiency. They illustrate the differences in how such efficiency might be obtained. In the modular case, the parameter values determine how sequencing can be done. In such case, function/modules with higher probability parameters receive higher priority, i.e. are sequenced earlier in the process.

In the nested mapping, however, the positions in the hierarchy of functions are the determining factors. Here, functions which are of lower levels receive priority as they enable other functions of higher levels to be assessed in the risk-mitigation activity.

**Solution algorithm**

From such sequence given in Theorem 4.5, we are again in a position to replicate the methodology for arriving at the optimal solution shown in the previous section. Here, again, we use the optimal sequence to sort modules and functions. The problem is then reduced to that of how to split these modules into iterations.

Note that whereas the probability parameters determine the sequence in the modular mapping case, the functional mapping determines the sequence in this nested structure. Probability parameters and impact parameters play no role in determining the sequence in this case. They later play the roles in determining how to split the sequence into iterations, however.
Iteration parameters

Because the sequence is dictated solely by the functional mapping, the characteristics of the optimal policy for the nested structure might be different from that of the modular structure, whose characteristics derive from the fact that functions/modules are ordered so that probability parameters are in descending order.

To illustrate this, let us take at an example in which the probability parameters of function in the sequence are in ascending order. In particular, this example has 100 modules and 100 functions. The probability parameters and the impact parameters are generated in such a way that the sequence gives rise to probability parameters which are increasing, as oppose to decreasing in modular case.

The chart in Figure 4.19 show iteration probability, iteration impact, iteration rework, and the ratio of iteration probability and iteration impact for each iteration at the optimal solution of the example.

From this, we clearly see that the trend is the reverse of the previous case. In other words, iteration probability is increasing, iteration impact is decreasing, while the ratio is decreasing.

The purpose of this example is to show that the trend we established in the modular mapping is a direct result of the trend in the parameters at the function/module level. In such case, probability parameters are found to be decreasing in the optimal solution. This gives rise to the pattern we
saw in the last section. Here, however, such trend in the parameters at the function/module level does not exist. Functional mapping, not the value of the parameters, determines the sequence. Therefore, the iteration parameters can exhibit a pattern which is totally different from that we found in the modular mapping.

Modular vs. Integral Functional Mapping

Another important aspect of a product in the management of the process has to do with the mapping between functional and physical domains. This corresponds to the traditional definition of product architecture given in Ulrich and Eppinger (2000). In this sense, product architecture is classified into two major categories of modular versus integral architecture. This dichotomy represents an effective way to visualize architectures.

In general, a modular architecture is perceived as preferable to an integral architecture when it comes to managing a development process. With regard to the product architecture, Ulrich (1990) explains that the alternative architectures affect the level of efforts required in system-level design and the amount of rework in the detailed design and product test and refinement.

While a modular architecture reduces the efforts of engineers and testers during the last two phases by allowing the engineers to work on different parts of the system simultaneously, it also requires significant efforts on the architects during the system-level design. Therefore, the decision to choose between a modular architecture versus an integral architecture needs to bear this trade-off in mind.

Such effects have to do with the fact that the information requirements among different parts in a modular architecture is generally less than that required for an integral architecture. Generally, we have more flexibility in the assignment of modules when their functions are independent.

However, the underlying assumption from such argument lies in that architecture specified for a modular architecture is accurate and is not subject to change. In our model parlance, this implies that the interface specified is accurate and is not subject to change.
The issue I will try to explore in this section is related to the case when such assumption is not true, i.e. when interfaces are subject to change. Does modular architecture still yield benefit? and for what reasons?

![System MOD and System NES matrices](image)

**Figure 4.20 System matrix of system MOD and system NES**

To illustrate the implications of architecture in this case, let us consider two systems (Figure 4.20) with different functional mappings. System MOD has a modular functional mapping. System NES has a nested functional mapping. Both have \( N \) functions and \( N \) modules. I assume that their functions have the same probability parameters. All modules in each system have identical impact parameter, \( d \).

In this context, system MOD represents a modular architecture whereas system NES represents one of the possible integrated architecture. More specifically, it represents the extreme of integrated architecture in which all functions are related. In both systems, we see that there is the remaining interface, which is subject to change.
I employ the result obtained earlier on the shortest-path algorithm to compare the optimal solution between these two systems.

**Theorem 4.7:** The optimal expected total work of system MOD is no greater than that of system NES.

**Proof:**

We recognize that both systems can be mapped to shortest-path problem as we explained earlier. Such shortest-path problem corresponds to a particular sequence performed on the functions and modules of the system. For system MOD, we sequence the module such that probability parameters are decreasing. For system NES, the modules are sequenced based on the functional mapping.

We notice that the optimal sequence of system MOD results in a network which is one of the possible sequences of system NES. In other words, we can sequence the modules in system MOD to have the identical network (identical arc lengths) as that of system NES. But this is only one possible sequence which might not satisfy the decreasing probability parameters rule. So, this might or might not be an optimal sequence for system MOD.

However, since it results in the identical sequence as that of system NES, the shortest paths that results has the same length. Therefore, such sequence (optimal for system NES, not necessarily for MOD) yields the same objective for both system NES and MOD. So, the optimal objective for system MOD, which is better than that of this potentially suboptimal sequence, is equal to or less than this, which is the optimal objective for system NES. In other words, the optimal objective for system MOD is equal to or less than that of system NES.

The essence of this proof lies in the comparison of the solution spaces of the two systems. Here, I find that that of system MOD is the superset of that of system NES. Therefore, the optimal work of system MOD is at least as good as system NES. From this, I conclude that modular architecture outperforms integrated architecture even in the context when there is remaining uncertainty in the form of uncertain interface.

As can be seen, the advantage of modular architecture when remaining uncertainty is present derives from a different source from what is mentioned in the literature. In the case of fixed architecture, we derive the benefit from the ease in dividing up work among development teams. When there is remaining uncertainty in architecture, which affects all the modules, this advantage is compromised.

In the case when uncertainty is present, we clearly see that modular architecture still has an advantage due to the flexibility at which we can execute the development tasks. In other words, we can sequence the task in the way that more uncertainty is taken out early on in the process. When the architecture is integrated, e.g. a nested mapping, the sequence is dictated by the functional mapping. Thus, the system with a modular functional mapping fares better because there is more flexibility in managing the remaining risks in the system.
4.5 Interactions of Risks

*Problem Formulation*

In our model, interfaces represent the uncertainty in the development process. They correspond to the issues to be dealt with in risk management. So far, we focus our attention on system in which there is only one uncertainty. We looked into how such issue can be managed given our knowledge about how each part of the system contributes to the reduction in the probability of change and the increase in the impact of change.

In this section, we turn our attention to the system which possesses more than one interface. This is closer to reality in which there are multiple issues that need to be dealt with. According to our model, these issues could in fact interact with different parts of the system in a very complicated way. The problem remains the same as in the case of single interface. We need to determine what piece of the system to be developed first, second, and so on in order for us to manage the risk in the most efficient manner.

![Diagram of a system with multiple interfaces]

*Figure 4.21 System with multiple interfaces*

First, consider the fixed-iteration formulation of work-minimization problem for a system with multiple interfaces.
\[
W^* = \min_{S^1, S^2, \ldots, S^J} \left[ \sum_{j=1}^{J} \sum_{q=1}^{Q} \sum_{m \in S^j} d_{qm} \cdot f_q \left( \left\{ P_{mq} \right\}_{m \in S^j \cup [i]} \left( S^{j-1} \right) \right) \right]
\]

\[
= L \cdot J + \min_{S^1, S^2, \ldots, S^J} \left[ \sum_{j=1}^{J} \sum_{q=1}^{Q} \sum_{m \in S^j} d_{qm} \cdot f_q \left( \left\{ P_{mq} \right\}_{m \in S^j \cup [i]} \left( S^{j-1} \right) \right) \right]
\]

Notice that each term in the bracket under summation for a given \( q \)

\[
R_q \left( \hat{S}^j, \hat{S}^{j-1} \right) = \sum_{m \in S^j} d_{qm} \cdot f_q \left( \left\{ P_{mq} \right\}_{m \in S^j \cup [i]} \left( S^{j-1} \right) \right)
\]

represents the expected rework that results from interface \( q \) in the development. The total expected rework is the sum of such terms from all \( q=1, \ldots, Q \). Note that the summation form is a direct result of our assumption that the impact from interfaces are independent (see section x, chapter 3)

In this regard, we can see that this case is similar to the case of single interface in that we need to manage pattern in the reduction of probability and the increase of the impact. But here, we need to balance such pattern across several interfaces. This might not result in a simple maximal reduction in probability for minimal increase in impact as we saw in the case for single interface. The work-minimization problem is now made complicated by the fact that several issues are at play.

We will not attempt to solve the problem in the general case. Our focus, instead, is to show the features of model which result from the fact that there are multiple interfaces. In the following two sections, we analyze two special cases of system with multiple interfaces. These cases illustrate particular insights for the risk management of such systems.

In particular, in the first example, we learn that in special circumstances, we can transform this multiple-interface system into the equivalent single-interface system using the weighting parameters to combine multiple interfaces into a single interface. In the second example, we learn how in some cases, there is a natural grouping of interfaces in the optimal iteration policy.
**Interface-based weighting**

To gain insight into the problems of this type, we begin this section by looking at a particular type of multiple-interface system with the following characteristics.

a) The functional mapping is modular, i.e. one function corresponds with one module.

b) The impact parameters of all the modules for an interface are identical. This is shown by the identical rows of the lower right matrix. Under this assumption, all modules are affected by the interfaces in the same way. Note, however, that the effect from these interfaces might not be identical.

c) Probability function is of additive form.

This is a system with $N$ functions, $Q$ interfaces, and $N$ modules. The following system matrix shows the parameters of the system.

```
<table>
<thead>
<tr>
<th></th>
<th>F₁</th>
<th>F₂</th>
<th>F₃</th>
<th>...</th>
<th>Fₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>p₁₁</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>p₁₂</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>pₙ₁</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>pₙ₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>pₙₙ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>
```

This is a risk matrix where $d₁$, $d₂$, ..., $dₙ$ represent the impact parameters for each interface.
From this specification, the problem can be written as

\[ W^{J^*} = L \cdot J + \min_{s^1, s^2, \ldots, s^l} \left[ \sum_{j=1}^{J} \sum_{m \in \delta^j} \sum_{q=1}^{Q} \sum_{n \in \delta^j} d_q \cdot p_{nq} \right] \]

\[ = L \cdot J + \min_{s^1, s^2, \ldots, s^l} \left[ \sum_{j=1}^{J} \sum_{m \in \delta^j} \sum_{n \in \delta^j} \sum_{q=1}^{Q} d_q \cdot p_{nq} \right] \]

\[ = L \cdot J + \min_{s^1, s^2, \ldots, s^l} \left[ \sum_{j=1}^{J} \sum_{m \in \delta^j} \sum_{n \in \delta^j} \pi_n \right] \]

where

\[ \pi_n = \sum_{q=1}^{Q} d_q \cdot p_{nq} \]

which is the weighted probability for function/module \( n \). Note that \( \pi_n \) need not be less than one in this case. But the resulting problem is simply the work-minimization for a system with single interface where the probability parameters are given by \( \pi_n \) and impact parameters all equal to unity.

This amounts to the reduction of the multiple-interface system to a single-interface problem. The probability parameters of the single-interface problem correspond to the weighted-sum of the multiple-interface system over the interfaces using the impact parameters \( d_q \) as the weighting factors.

So, what does this mean? Recall that in the original problem, each function/module (recall that the system is modular) contributes to the risk reduction in the form of probability of change in the interfaces and to the growing impact in the form of impact parameters.

Because all the impact parameters are assumed to be identical across function/modules, we only need to be concerned with the probability parameters. For a system with single interface, this calls for a sequence in which we sort functions from the highest probability parameters to the lowest. In this case of multiple interfaces, however, the problem is a little more complicated since we have more than one probability parameter for each function/module.

However, due to the structure of the problem, it turns out that we can construct the single-interface equivalent in which the probability parameters are the weighted sum of the original
probability parameters. We can then use the old trick and solve the problem by prioritizing the modules from the highest "modified" probability parameters to the lowest.

That the weighting parameters are the impact parameters suggests that we give more emphasis to the interfaces with higher impact parameters. Because impact parameters are the weighting factors for the modified probability parameters, a large impact parameter for an interface amplifies all the probability parameters which correspond to that interface. This implies that a function/module with relatively higher probability parameters for that particular interface is now more likely to be placed towards the beginning of the process due to such amplification.
Interface-based grouping

To illustrate another type that interfaces come into picture, let’s now turn our attention towards a system with multiplicative probability function. The system parameters are shown in the system matrix below.

\[
\begin{array}{cccccccc}
F_1 & F_2 & \ldots & F_{N_1} & F_{N_1+1} & F_{N_1+2} & \ldots & F_{N_1+N_2} \\
p_1 & p_1 & p_1 & 0 & 0 & 0 & 0 & I_1 \\
0 & 0 & 0 & p_2 & p_2 & p_2 & p_2 & I_2 \\
1 & & & d_1 & d_2 & M_1 & & \\
1 & & & d_1 & d_2 & M_2 & & \\
& \ldots & & \ldots & & \ldots & & \\
1 & & & d_1 & d_2 & M_{N_1} & & \\
1 & & & d_1 & d_2 & M_{N_1+1} & & \\
1 & & & d_1 & d_2 & M_{N_1+2} & & \\
1 & & & d_1 & d_2 & M_{N_1+N_2} & & \\
\end{array}
\]

Figure 4.23 Risk matrix

From the matrix, we can see that this system has the following characteristics:

a) The system has a modular functional mapping.
b) The system has two interfaces.
c) There are two types of functions/modules.
d) Type 1 has probability parameters of \( p_1 \) for interface 1 and 0 for interface 2.
e) Type 2 has probability parameters of 0 for interface 1 and \( p_2 \) for interface 2.
f) Both types have \( d_1 \) and \( d_2 \) as impact parameters corresponding to interface 1 and 2, respectively.
g) There are \( N_1 \) and \( N_2 \) functions/modules for Type 1 and Type 2, respectively.

We look in the case of **two iterations**, i.e. we decide to split the functions/modules into two iterations. In this case, since within a particular type, functions/modules are identical, the decision parameters come down to the number of functions/modules allocate to iteration 1 for each type.
Let \( n_1 \) be the number of functions/modules of type 1 allocated to iteration 1. Let \( n_2 \) be the number of functions/modules of type 2 allocated to iteration 1. Note that for a product-type probability function, the iteration probability can be written as

\[
P_q = 1 - \prod_{m \in \Delta'} (1 - p_m)
\]

From this, we can write the expected rework as follows.

\[
R(n_1, n_2) = R^1 + R^2
\]

\[
= \left( 1 - \left( 1 - p_1 \right)^{n_1} \right) \left( n_1 + n_2 \right) d_1 + \left( 1 - \left( 1 - p_2 \right)^{n_2} \right) \left( n_1 + n_2 \right) d_2
\]

\[
+ \left( 1 - \left( 1 - p_1 \right)^{N_1 - n_1} \right) \left( N_1 + N_2 \right) d_1 + \left( 1 - \left( 1 - p_2 \right)^{N_2 - n_2} \right) \left( N_1 + N_2 \right) d_2
\]

This expected rework is an explicit function of \( n_1 \) and \( n_2 \). To gain an insight into the problem, let's change our variable so that it becomes an explicit function of \( n_1 \) and \( n = n_1 + n_2 \).

\[
R(n_1, n) = \left( 1 - \left( 1 - p_1 \right)^{n_1} \right) \cdot n_1 \cdot d_1 + \left( 1 - \left( 1 - p_2 \right)^{n_2} \right) \cdot n_2 \cdot d_2
\]

\[
+ \left( 1 - \left( 1 - p_1 \right)^{N_1 - n_1} \right) \cdot N_1 \cdot d_1 + \left( 1 - \left( 1 - p_2 \right)^{N_2 - n_2} \right) \cdot N_2 \cdot d_2
\]

Furthermore, let's now assume that \( N_1 \) and \( N_2 \) is a very large number and we estimate \( R \) as a function of \( \alpha_1 = n_1/N \) and \( \alpha = n/N \) where \( N = N_1 + N_2 \).

\[
R(\alpha_1, \alpha) = \left( 1 - \left( 1 - p_1 \right)^{\alpha_1} \right) \cdot N \cdot \alpha_1 \cdot d_1 + \left( 1 - \left( 1 - p_2 \right)^{\alpha} \right) \cdot N \cdot \alpha \cdot d_2
\]

\[
+ \left( 1 - \left( 1 - p_1 \right)^{N_1 - \alpha_1} \right) \cdot N_1 \cdot d_1 + \left( 1 - \left( 1 - p_2 \right)^{N_2 - \alpha} \right) \cdot N_2 \cdot d_2
\]

Differentiate w.r.t. \( \alpha_1 \) twice, we get

\[
\frac{\partial^2 R(\alpha_1, \alpha)}{\partial \alpha_1^2} = -N^2 \cdot \log(1 - p_1) \cdot 2 \cdot \alpha_1 \cdot d_1 \cdot \left( 1 - p_1 \right)^{\alpha_1} - N^2 \cdot \log(1 - p_2) \cdot 2 \cdot \alpha \cdot d_2 \cdot \left( 1 - p_2 \right)^{\alpha}
\]

\[
- N^2 \cdot \log(1 - p_1) \cdot d_1 \cdot \left( 1 - p_1 \right)^{N_1 - \alpha_1} - N^2 \cdot \log(1 - p_2) \cdot d_2 \cdot \left( 1 - p_2 \right)^{N_2 - \alpha}
\]

Hence, \( R \) has a negative second partial derivative w.r.t. \( \alpha_1 \). This means that \( R \) is a concave function of \( \alpha_1 \) and that the minimum takes on the extreme point. Figure x shows this geometrically. There are many possibilities for such extreme points. But in either case, this represents what we call an "exhaustive policy", in which modules of a certain type is exhausted before we allocate the module of the other type.
This represents the case when there is a benefit for completely handling one issue before turning on to the next issue. In the previous case, the decision comes in the form of the weighting factor. In other words, we decide what to do based on how much it contributes to the overall risk reduction. In this case, however, it is optimal to handle only one issue completely before shifting the attention to another issue.
4.6 Process Models

The earlier discussions have centered on the use of the risk dynamics model and the resulting work-minimization problem to provide answers related to the development process and the risk management. In this section, we will explore the application of the model. First, we will discuss the prototyping problem to show how the model can be applied as a decision support tool.

The problems related to data collection are then discussed. Such problems, in turn, lead to the development of a modified risk matrix, a simplified version of the model with fewer requirements on the data. The modified risk matrix is then used to revisit our discussions on the process models and reveal some of the fundamental problems encountered by these process models.

*Model application: prototyping problem*

Here, let us consider the development of a system whose risk matrix is shown in Figure 4.25. The system consists of two parts, represented by module $m$ and module $M$. Let $p_{nq}$ be the probability parameter for function $n$ and interface $q$. Let $d_q$ and $D_q$ be the impact parameters for interface $q$ and module $m$ and module $M$, respectively. Let $h_n$ and $k_n$ be the functional mapping parameters for function $n$ and module $m$ and module $M$, respectively.

\[
\begin{array}{cccccc}
F_1 & F_2 & \ldots & F_N & I_1 & I_2 \\
P_{11} & P_{21} & \ldots & P_{N1} & I_1 & \ldots \\
P_{12} & P_{22} & \ldots & P_{N2} & I_2 & \ldots \\
P_{1Q} & P_{2Q} & \ldots & P_{NQ} & I_Q & \ldots \\
h_1 & h_2 & \ldots & h_N & d_1 & d_2 \\
k_1 & k_2 & \ldots & k_N & D_1 & D_2 \\
\end{array}
\]

*Figure 4.25: Risk matrix in a prototyping problem*

Consider a situation where the smaller part of the system, module $m$, is proposed as a prototype to be built to mitigate certain risk issues prior to the development of the rest of the system, module $M$. Let $L$ be the risk-mitigation work, the amount of work needed to perform necessary testing and other activities on module $m$. The question to be answered is whether we should build the prototype or not.
Using the total expected work as the objective function, this problem is reduced to that of comparing the objective in two cases. In the base case, we develop the whole system in one iteration. In the prototype case, we develop the system in two iterations, with module $m$ in iteration 1 and module $M$ in iteration 2.

Let $W$ be the total expected work of the base case which carries out everything in one iteration. Let $W^*$ be the total expected work of the prototype case, i.e. the task corresponding to module $m$ is executed in the first iteration and the corresponding risks mitigated before the rest of the system, $M$, is executed.

From this formulation, we can write $W$ and $W^*$ as expressions of the above parameters. Through some simple algebraic work, we can derive the difference between $W$ and $W^*$ as follows.

$$W - W^* = \sum_{q=1}^{Q} \left( \sum_{n,h_n=1,k_n=0} p_{nj} \cdot D_q \right) - L$$

$$= \sum_{n,h_n=1,k_n=0} \left( \sum_{q=1}^{Q} p_{nj} \cdot D_q \right) - L$$

$$= \sum_{n,h_n=1,k_n=0} \alpha_n - L$$

$$\alpha_n = \sum_{q=1}^{Q} p_{nj} \cdot D_q$$

This difference represents the net saving in the amount of work from the prototype. From the expression, the net saving is the difference between the avoided rework and the risk-mitigation work, $L$. We decide to build a prototype if the net saving is positive, i.e. the avoided rework is higher than the risk-mitigation work, $L$. This example shows how the model helps us making a decision in planning the development process.

**Problems with data**

The prototyping example shown here represents a very simple application of the model as a decision support tool in the development process. The generality of the risk dynamics model and the work-minimization problem allows us to handle cases which are much more complicated, such as those with a large number of modules, an arbitrary sequence of modules, an arbitrary number of iterations, a complex functional mapping, and so on.
In certain circumstances, however, the application of the model to these complicated cases might encounter some major data problems. We might not always be able to obtain the data necessary for deriving the model parameters. This is especially true when we try to apply a very complicated model at the early phase of a development process.

The problem is partly due to the nature of the design activity, in which the system is elaborated in more refinement as the process unfolds. Early on in the development process, the system is typically decomposed only to the level that can be used by engineers and developers to carry out their tasks. The details are left for them to decide for themselves. In this case, it is unlikely that the detailed data needed for the model will be available, let alone the cost and the time involved in obtaining such data.

Two observations on the design process are useful in making simplifications on the model to help alleviate the data shortage problem. First, from the field experience, data related to functions and interfaces seem to be easier to obtain than that related to modules. Often times, developers think of the system more in terms of functions than modules, especially for a top-down design. The data related to functions are usually captured explicitly in the design documents.

Interfaces, on the other hand, represent the underlying assumptions in the design. Although sometimes they are not explicitly captured in the design document, the interfaces of a system can be obtained easily by asking the right questions to the system architects. Modules, on the other hand, seem to be the results of the refinement of system specifications and are not available until later in the process. In this regard, a simplified version of the model which only needs data related to functions and interfaces is more likely to find practical use than those which insist on mapping out all the modules of the system.

Second, the process planning can be done in a rolling-horizon basis. In a typical development process, there are some formal plans which spell out in details the number of iterations in the process and the tasks to be done in each iteration. However, the developers also make plan to handle risks within such iterations as well. As more information becomes available and the risk issues surface in the development, they sometimes need to make decisions on the risk mitigation as the process unfolds. From this perspective, a two-iteration model can be used on a rolling-horizon basis to make plans for multiple iterations as well.

Chapter 4: Model Analysis
**Modified risk matrix**

To build a simplified model, let us take a second look at the prototyping problem discussed earlier in this section. The prototyping decision is determined by the avoided rework and the risk-mitigation work. The avoided rework in turn is the sum of \( a_n \), the avoided rework from function \( n \), for all \( n \) that is associated only with the prototype module \( m \) (i.e. \( h_n=1, k_n=0 \)). The avoided rework from each function, in turn, is the weighted sum over the interfaces of the impact parameters of \( M \) with the probability parameters as the weighting parameters.

Let us now consider a *modified risk matrix* in figure 4.26. Here, we have the original probability matrix, the row corresponding to \( M \) of the impact matrix, and the additional top row. Each element of the top row corresponds to a function and is the weighted sum of the probability parameter in the column corresponding to that function and the impact-parameter column.

![Figure 4.26 Modified risk matrix](image)

Each of the elements in the top row is called a **risk index**, and is the sum product of the underlying column of probability parameters and the column of impact parameters.

In the prototyping problem, the risk indices help us decide whether a prototype should be built. Each risk index represents the avoided rework that corresponds to each function. The total avoided rework is obtained by summing the risk indices that correspond to functions associated exclusively to the prototype \( m \). Subtracting this with the addition cost, \( L \), for the risk-mitigation work, we can make a decision whether such prototype is warranted.

More generally, the risk index is a figure of merit of risk efficiency for each function. The larger the risk index of a function, the higher the avoided rework is obtained from the risk mitigation related to that function. When we have a choice of working on some of these functions, we should therefore work on functions with higher risk index early on. In other words, the risk index helps us determine the sequence of functions to be dealt with.
This notion of risk index also makes sense from the conceptual point of view. In thinking about risk, there are two things to be considered, the uncertainty to be reduced and the impact to be incurred if such risk is not dealt with. In ranking the issues to be dealt with, the rule of thumb among practitioners is that the higher the uncertainty reduction and the higher the impact, the better it is to rank such issues higher in importance or earlier in time. Risk index, as a product of the probability parameter and the impact parameter combines these two concepts mathematically to create one single index to measure the importance of different risk issues.

The modified risk matrix and the risk index can help us solve the problem mentioned above by providing a tool to help us decompose the system in an appropriate manner. Instead of relying on the traditional, often times arbitrary, way to decompose the system, say into product and process, and map such decomposition into tasks, this method calls for a more systematic analysis of risk before making such decomposition.

We are first forced to think in terms of the system functions and the interfaces that can cause changes in the process. These functions and interfaces can correspond to any parts of the system of their combinations. Since a system of any type can be thought in terms of the generic concepts of functions and interfaces, this method should be general enough to be applicable across the industries.

From such functions and interfaces, we are to derive the probability parameters that corresponds the amount of reduction in the uncertainty from each function, and the impact parameters that correspond to the consequence, in terms of costs, rework, or other measures, from the change in the interfaces.

The risk index is then calculated to tell us the significance of the functions from the risk management point of view. From such information, we are then in a position to decide how we should prioritize the issues to be dealt with and how to design a process policy that corresponds to the risk profile of the system in question.

We can also generalize the concept of risk index further by noting that the type of impact in risk management is sometimes things other than rework. Often times, we think of the impact in terms of financial costs, among other measures. By replacing the impact parameter, in the form of rework, we come up with a more general definition of risk index, which can measure any type of objectives that are of concerned in the development process.
Process models revisited

Now, let us use this simplified tool to look back at the concurrent engineering process. In particular, I will demonstrate how the question of a concurrent engineering versus an overlapping process arises.

<table>
<thead>
<tr>
<th></th>
<th>F_1</th>
<th>F_2</th>
<th>F_3</th>
<th>F_4</th>
<th>F_5</th>
<th>F_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>a_2</td>
<td>a_3</td>
<td>a_4</td>
<td>a_5</td>
<td>a_6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Product

Mfg. Process

Figure 4.27 Modified Risk Matrix for concurrent engineering example

The system to be developed here consists of the product and the corresponding manufacturing process. Figure 4.27 shows the risk index that would have been derived from the system (hiding the probability and the impact parameters) and the functional mapping that relates the functions to either product or manufacturing process. The function is arranged in such a way that the risk index is sorted in a descending order, i.e. a_n > a_{n+1}. From this, the ideal process is that which considers F_1, F_2, ..., F_6 in that sequence.

Now, assume that the planning is done with the decomposition (into product and manufacturing process) in mind. In other words, the planner thinks in terms of tasks corresponding to the product and the manufacturing process, without knowledge of the risk index.

Using the ‘throwing over the wall’ approach, product is designed to its completion before the process design takes place. From the risk index, this is certainly not a good way to structure the development process. Although this puts F_1 and F_2 before other tasks, it also lumps together F_3, F_4, F_5, F_6. (F_3 and F_4 will not be handled until the design of the manufacturing process).

Recognizing the problem, now we switch to concurrent engineering, doing everything in parallel. This has the effect of shifting F_3 and F_4 ahead of F_5 and F_6 (if the design of manufacturing process is sequenced the right way) and improves the process. But there is another problem as it also has the effect of lumping F_1, F_2, F_3, and F_4 together.

To solve this problem, now we consider the overlapping process, where product is designed a little earlier (to take care of F_1 and F_2) before the process design kicks in and carried out in
parallel to product design. Here, $F_1$ and $F_2$ are shifted earlier in the process, then $F_3$ and $F_4$, and finally, $F_5$ and $F_6$. Overlapping process seems to solve the problem in this case.

The purpose of this example is to show that the problem in structuring the process arises from the way we conceive of tasks in the development process. We usually conceive of them in a certain way according to how we divide up the system, in this case into product and manufacturing process, which in turn leads to the decomposition tasks. We then proceed to structure the process by arranging such tasks along the time domain.

But the mechanism of risks is more complicated than that. We see that risks can arise from all the subsystems and at varying degrees. And the way we carve up the tasks does not necessarily lend itself to the management of such risks.

In this example, we see that the overlapping process seems to solve the problem that we had. But it is not hard to see that this is due to the special structure of the system. It is not difficult at all to construct another system like that shown in Figure 4.28 which possesses such a complicated functional mapping that no matter how we arrange the tasks, we will not derive at the appropriate sequence suggested by the risk index.

<table>
<thead>
<tr>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>$F_4$</th>
<th>$F_5$</th>
<th>$F_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_1$</td>
<td>$\alpha_2$</td>
<td>$\alpha_3$</td>
<td>$\alpha_4$</td>
<td>$\alpha_5$</td>
<td>$\alpha_6$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Subsystem 1
Subsystem 2
Subsystem 3
Subsystem 4
Subsystem 5

*Figure 4.28 Examples of a system with complicated functional mapping*

The issue dealt with in process models is that of the two process parameters, task sequence and information timing. In Chapter 1, we see that process models seem to have arisen from different industries. Each is limited because it only deals with the context limited to that particular industries. In addition, some of the process models do not completely solve the problem. One of the examples is the concurrent engineering. We see that even after the concurrent engineering improves the process by prescribing that product and process be designed concurrent, an issue arises as to the degree of concurrency that is appropriate.

*Chapter 4: Model Analysis*
The fundamental problem which gave rise to the process models have to do with the fact that we still rely on the traditional way to decompose a system to plan the tasks in the process. This is seen in concurrent engineering which decomposes tasks into product and process. System-focused development for R&D arises in reaction to the decomposition of activity into components (closer to research) and the system (development).

From the analysis that has been shown in this chapter, it is clear that to solve this problem completely, we need to look at the risk to be managed rather than the modules to be developed. Instead of relying on the intuitive decomposition, we should look at the system from the point of view of risk and plan the process accordingly.

4.7 Conclusions

This chapter derives important insights from the work-minimization problem formulated in the previous chapter. The optimal process policy is shown to have the characteristics of risk efficiency, with a large reduction in probability for a given increase in impact. This chapter also shows how the risk concentration, the product architecture (functional mapping), and the interactions of risks (multiple interfaces), affect the process policy and the process performance.

Process models are seen in this light as a way to manage the risk in the development process. They arise in reaction to the practice which relies on the tradition way to plan the process. While effective in many cases, process models still have some limitations. The limitations are due to the fact that, in proposing alternative ways to structure the process, the process models still rely on the traditional decomposition, which is the root cause of the very problems they were meant to solve.

The model proposed in this thesis is seen as a way to solve this fundamental problem by avoiding the traditional decomposition entirely, instead looking at the system purely from the risk perspective. However, the data that are needed for the model might be difficult to obtain. Therefore, this chapter proposes the modified risk matrix and the risk index, which is more practical from the data point of view.

With the understanding of the model, the readers are ready to explore the practicality of the model in the next two chapters. Chapter 5 shows how we can apply the model in a case study. Chapter 6 explores the numerical approach related to the model.
CHAPTER 5: CASE STUDY: SUPERPAGES.COM

5.1 Introduction

The case study is a result of interactions with the development team of SuperPages.com under the e-Business group of Verizon. The interactions span the time between mid-April to October 2001. During the interactions, I have a chance to talk to variety of people under the organization and monitor the project, which is the center of this case study, the migration of back-office to a new platform.

The first section gives a background of SuperPages.com. The second section gives a background on the migration plan for back-office system. We will cover two important subtasks for the migration plan. The first, which is covered in the next section, is the development of a pilot system called SuperCommerce. Here, we explore how the intuition from our model is used in the pilot planning.

The other project covered is ELP, which is an acronym for a new subsystem. This will be covered in three chapters. The first among the three gives a background of the project. The next section covers how we derive the model parameters from the project. Then, the third gives the numerical results. Finally, following these discussions of the case study, we will conclude the chapter in the last section.

5.2 Project Background

SuperPages.com

SuperPages.com is the equivalent of the Yellow Pages on the Internet. The site list contact information of businesses which can be searched under specified criteria by the Internet users. Some of these businesses are paid customers who pay service fees for some special display features of their own information.

The site has been developed first in 1995 in what was then GTE. The company initially leverages its advantage as the provider of the paper-based yellow pages in the project. The web site has gained more and more prominence as the web-based directory as the popularity of the internet
grew. Figure 5.1 shows the example of web pages from superpages.com. For more examples, please see (http://www.superpages.com)

At present, this website is under the responsibility of eBusiness division of Verizon. In addition to providing the web-based yellow pages under Superpages.com, the company is the provider of director information for several other leading web-based yellow pages as well.

**Figure 5.1 Example of Superpages.com**

**Project timeline**

We obtain the initial assistance from Professor George Kourc in contacting the Superpages.com team. We first had a discussion with the leader of Superpages.com, Lev Koyfman. After the initial conversation, we attended several meetings to identify the appropriate project for an example of our model.

We later decided to focus on the migration of back-office system to a new platform. Since then, we have worked with the back-office team to obtain information and develop the model. Table 5.1 shows the project timeline.
<table>
<thead>
<tr>
<th>Month</th>
<th>Project Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2001</td>
<td>Initiated first contact.</td>
</tr>
<tr>
<td></td>
<td>Finalized the non-disclosure agreement.</td>
</tr>
<tr>
<td>May</td>
<td>Began conversation with key people in the development organization.</td>
</tr>
<tr>
<td></td>
<td>Attended team meetings to identify the appropriate project to work on.</td>
</tr>
<tr>
<td></td>
<td>Decided to focus the migration of back-office system to a new platform.</td>
</tr>
<tr>
<td></td>
<td>Presented the project plan to the team</td>
</tr>
<tr>
<td>June</td>
<td>Interviewed with the developer on the iteration plan</td>
</tr>
<tr>
<td></td>
<td>The pilot study of SuperCommerce began.</td>
</tr>
<tr>
<td></td>
<td>The development of ELP under old platform began.</td>
</tr>
<tr>
<td>July</td>
<td>The pilot study of SuperCommerce continued.</td>
</tr>
<tr>
<td></td>
<td>The development of ELP under old platform finished.</td>
</tr>
<tr>
<td>August</td>
<td>The pilot study of SuperCommerce finished.</td>
</tr>
<tr>
<td>September</td>
<td>The development of ELP under new platform began.</td>
</tr>
<tr>
<td></td>
<td>Began the study of design documents for ELP.</td>
</tr>
<tr>
<td></td>
<td>Interview with ELP developer began.</td>
</tr>
<tr>
<td>October</td>
<td>Interview with ELP developer continued.</td>
</tr>
<tr>
<td></td>
<td>Began obtaining model parameters for ELP.</td>
</tr>
<tr>
<td>November</td>
<td>Design of ELP finished</td>
</tr>
<tr>
<td></td>
<td>Finalized the model parameters for ELP</td>
</tr>
</tbody>
</table>
5.3 System and Organization

_Overall Architecture_

The system can be divided into two major subsystems, the front-end subsystem and the back-office subsystem. The front-end subsystem serves the end users of SuperPages.com and was referred to within the organization as superpages system. The back-end subsystem serves the internal users, primarily sales people, who need to have an access to and make changes to the data regarding the customers. Figure 5.2 shows the overall scheme of the system.

![Diagram of Superpages.com system]

Figure 5.2 Overall scheme of Superpages.com

_Development Team_

As can be seen from figure 5.2, these two subsystems are linked only through the central database. Therefore, they are relatively independent. In fact, these two systems are under the responsibility of the two separate groups within the organization.

The first group is called superpages team and is responsible for the superpages system or front-end system. This group has around 12 members. Some of these members work on project other than Superpages.com as well. The second group is called back-office team and is primarily responsible for the back-office system. The group has 6 people including a database administrator.
Back-office Subsystem

Back-office subsystem is the system that supports the internal users, i.e. the salespeople who represent the customers, i.e. advertisers of Superpages.com. The system is also a web-based system.

![Diagram of back-office system](image)

**Figure 5.3 Scheme of back-office system.**

Figure 5.3 shows the scheme of the back-office system. As can be seen, the system is a cluster of identical application servers clustered together to handle traffic and connected to a common database server. The development efforts primarily involve the application components that rest on these clustered application servers.

The old architecture the application server is shown in Figure 5.4. This architecture is based on Netscape application server.

![Diagram of application server](image)

**Figure 5.4 Architecture of Application server**

Chapter 5: Case Study: SuperPages.com
As seen in the picture, the application server sits between the browser and the database server. Within the application server, various types of application components sit on top of the Netscape application server, which provides the platform on which these components deliver their functionalities.

The application components can be divided into three tiers.

1. The first tier, consisting of tagged-html and AppLogic Java classes, handles the user interface and the flow of the applications. Tagged HTML are files which look like HTML files except for special tags that allow the server to paste the external values from other system components and make it into a HTML files that then are passed to the browser. AppLogic are Java classes that extend the AppLogic classes of Netscape Application Server. The AppLogic classes are simply a mechanism to process a request and turn it into a response back to the users. It is the means by which the end users access the internal functionalities of the servers.

2. The second tier, consisting of Workflow Engine, Object Cache, and XML rules, handles the application logic and the transaction features of the application. Without going into the details of this tier, we simply state that the three basic components mentioned work together to perform the basic data operations for the system. In particular, it provides the transaction features of the system which ensure that a given series of operations are performed in such a way that do not compromise the integrity of the data. This is where the core functions of the system lie.

3. The third tier, consisting of SQL files and database API, handles the data manipulation. This tier provides the interface to the database server and is called upon by various components in the second tier.

From the point of view of system operations and maintenance, the second tier represents the major concern for the back-office team. This is the most critical part of the system as it embeds the core data operations and provides the transactional integrity of the system. However, it also has the most complexity and inconsistency (for the reasons to be seen in the next section). As system grow, this became increasingly worse. Such characteristics mean that this part of the system required increasingly more time and efforts to maintain. It became clear that something need to be urgently done to eliminate this problem.
5.4 Platform Migration

Project initiation

Due to organization change and staff turnover, the old back-office system has passed through many hands, each inheriting the system without having a complete knowledge about its design. Due to the success of Superpages.com business, new functionalities were hastily added to the system despite the incomplete architectural knowledge. The team was not in a position to re-architect the system.

This results in a system that which possesses unnecessarily high complexity and rampant inconsistencies. The maintenance of the workflow engine and associated xml files were time consuming. The system as a whole was error-prone and very difficult to maintain. In addition to that, it was difficult to develop applications using the old platform based on AppLogic model, especially when compared to the new, more flexible platform available in the market. More importantly, after this division of Nestcape (which develops web servers) was bought by Sun Microsystems, the company decided that to discontinue the business of Netscape Application server. This means that the old system will no longer be supported by the software vendor.

By May 2001, when we started our conversation with the team, a decision was made to move the whole back-office system onto a new platform, the iPlanet server, which was a product of IBM. This server is based on J2EE platform, also by Sun Microsystems. This platform is one of the new-generation enterprise-class platforms available in the market (the other one is .NET technology by Microsoft) aimed to facilitate the development for enterprise-class applications.

J2EE architecture

The J2EE platform (Java™2 Platform, Enterprise Edition) is the set of specifications for enterprise-class platform. This, together with other competing platforms, such as Microsoft’s .NET, represents a new technological direction in the development of enterprise applications. The salient benefit of such platforms lies in the flexibility and the reduced cycle time. Those platforms are based on standardized component that can be reused or purchased from third-party developers. They also automatically handle a lot of application behavior such as transactions, thus freeing the developer’s time to focus on the application logic.
The core of J2EE relies on the concept of Enterprise Java Beans (EJBs), which are components that reside in the bean container. There are many types of EJBs, namely Session Beans, Entity Beans, Data Access Beans.

The bean container is the underlying infrastructure that takes care of application behavior such as life cycle management, transaction management, etc. of the application. The developers can easily develop components that implement the application logic and let the bean container handles other application behaviors.

In addition to the EJBs, the platform also includes components such as JSP and Servlet which handle the user interface and the flow of applications and JDBC's which provide an interface to the database server. Figure 5.5 shows the overall concept of J2EE platform.

![Figure 5.5: Overall concept of J2EE server](image)

**Project in early stage**

To understand the migration plan, we first note to the reader that the team in fact benefits from a very good understanding of the system requirements. In contrast to other development projects in which the requirements specifications take a long time early on in the process, the back-office team can skip through this stage, as the development is purely based on the old requirements of the existing system.

In addition to that, we also note that the team can make use of some components under the old system. Although the old platform and the new platform are generally very different, there are many parts of the system that can be easily migrated to the new platform. If this is not the case, the team can rely on the logic that is embedded in the components of the old system to aid the
development of those under the new platform. Table 5.2 shows the components which implement equivalent functions in the old system and in the new system.

Table 5.2: Comparison of equivalent components in two platforms

<table>
<thead>
<tr>
<th>Role</th>
<th>Old platform</th>
<th>New platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td>Tagged HTML</td>
<td>JSP</td>
</tr>
<tr>
<td>Control</td>
<td>AppLogic</td>
<td>Servlet</td>
</tr>
<tr>
<td>Transactions</td>
<td>Workflow</td>
<td>Session Beans, Entity Beans</td>
</tr>
<tr>
<td>Data Access</td>
<td>SQL</td>
<td>Data Access Beans</td>
</tr>
</tbody>
</table>

Even though the team can draw on a lot of experience and components from the old system, they still have to put a lot of efforts into the migration. Only a few of the team had extensive experience in the development under J2EE platform. This means that the technical uncertainty, here the unfamiliarity with the new technology, will be the main source of uncertainty in this case.

In particular, the bulk of uncertainty lies in the design of Served and Session Beans, whose roles are to implement most of the application logics and, with the aid of the bean container, accommodate desired application behaviors such as transactions and life cycle management. In addition, some of the basic mechanisms provided through library calls also serve as the source of uncertainty.

The team was clear about one thing. They are clear that this project needs to be done in an iterative manner. Given the amount of risk involved and the limited resources available, the traditional waterfall process was out of the question. They favor the risk management benefit of the iterative process.

By June 2001, the team decided to do a pilot development. This in fact represents iteration no. 1 in our model parlance. In fact, this is what we call the ‘major’ iteration as opposed to iteration which resides within the major iteration. The pilot development was to be done on a subsystem named Super Commerce. As it turned out afterwards, the second major iteration will be on a subsystem called ELP.

Figure 5.6 shows the decomposition of back-office system. Figure 5.7 shows the project timeline for these two iterations. We will look at the details of these two iterations in the following sections.

Chapter 5: Case Study: SuperPages.com
Figure 5.6 Back-office system and subsystems

Figure 5.7 Project Timeline
5.5 Pilot Study: Super Commerce

Around early June 2001, the back-office team was swamped as usual with routine tasks and special requests. Everybody knew that the migration needed to be done at some point. They knew it needed to be done iteratively. Yet there was no consensus as to how to move forward.

Albert, a seasoned developer, had been with the company for almost twenty years and had been with the superpages operation since its inception. He knew that someone needed to take the initiative or the team would be overwhelmed with the regular tasks and the migration will not proceed. He had a plan for the pilot development to resolve several design issues for the new platform. However, he himself was very overwhelmed with tasks at hand and could not spare time to do it themselves. So, he needed to find someone to do the job for him.

Around early summer, the superpages team got several new summer interns to work for three months. The back-office team got one summer intern to work for them. Typically, the assignment for these summer interns was informal. At the beginning, the plan was to have this summer intern work on some of the maintenance tasks for the old system. Albert, with his pilot study in mind, thought otherwise.

He took the initiative to propose that the summer intern work on the pilot study for the new platform. This was a radical decision given that the study would be the foundation for the development of everything that followed. In addition to that, the intern himself did not have an experience with the new platform and needed to be guided and educated about it.

On the other hand, this assignment also made a lot of sense from other points of view. Since this is a pilot study and will be done on an independent system, there is little impact to the ongoing operation of the back-office system. He was the only one in the team who can devote the whole time to this pilot development and get it done by the end of summer. In terms of the guidance, Albert was comfortable with his ability to guide the intern on the new platform. He also was confident that he could capture the learning done on this study before the intern left the company after the summer.

So, the decision was reached that the summer intern will work on the pilot study. Now, the question came to which of the subsystems to be under the pilot study.
Albert in fact had this in mind as well. He planned on using the SuperCommerce subsystem as a test bed for the migration. SuperCommerce was a subsystem which was used by the customers to change some of their own data. They would be able to change their contact information and some services under their accounts. In this sense, SuperCommerce can be thought of as a stripped down version of the back-office system.

We interview Albert for the reason for choosing SupperCommerce as the test bed. The answer to the question is presented in the following quote

"SuperCommerce is a good candidate for the pilot system. The system in fact has a lot of the basic functionalities of the whole back-office system. This means that if we were able to develop the system, it is very likely that the remaining system should work. To develop SupperCommerce, we needed to work on the detailed design of Entity Beans and some of the Session Beans. These represent the major issues underlying the migration to the new platform. In addition to that, SupperCommerce is compact. The smaller size meant that we should be able to finish the pilot study in a short period of time.

Source: Interview at Verizon June, 2001

We found that this answer to corroborate the validity of the modified risk matrix method developed in section 4.6, which focuses on functions and interfaces as the two drivers of risks. Here, the expert developer thought of the system in terms of functions and the interfaces of the system to be developed. Functions in this case are the features of the back-office systems. Interfaces are the design of Entity Beans and Session Beans, which at this point of the process represent the most uncertain issues.

The heuristics he uses are no different from that of risk index. Instead of using a quantitative tool, here, he relies on his expertise in the technology and the system to come up with the appropriate subsystem to serve as a pilot study for the subsequent development work. These heuristics give him an idea of how much risk, in terms of the expected rework, he can take out from the system. Besides thinking about the saving in terms of the amount of rework he can save, he also thought in terms of the size of the subsystem for the pilot study.

SupperCommerce in this case is then a very good natural choice as the subsystem to be developed early on in the process. It has a high saving, as can be seen from the analysis of functions and interfaces, while at the same time has a compact size, a small \( L \).
In Figure 5.8, I recreate the potential heuristics he used while deciding whether to use SuperCommerce as a prototype for the migration plan. Due to a late arrival in the process, I do not have an opportunity to propose this method for him to make his decision. But the heuristics seem to conform to the method of modified risk-matrix

![Diagram](image)

**Figure 5.8 Potential heuristics for SuperCommerge**

<table>
<thead>
<tr>
<th></th>
<th>F₁</th>
<th>F₂</th>
<th>F₃</th>
<th>...</th>
<th>Fₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>α₁</td>
<td>α₂</td>
<td>α₃</td>
<td>α₄</td>
<td></td>
<td>αₙ</td>
</tr>
</tbody>
</table>

Subtract from the size-related measure of SuperCommerce to get a figure of merit for making a decision for **pilot study**.
5.6 ELP: Background

Requirements

Around May, the business team decided to launch a new product to their advertisers named ELP which allows their information to be displayed in more prominent ways with little additional cost. This requires the back-office team and the superpages team to jointly develop corresponding subsystems under both back-office system and superpages system to add the capability of the existing system.

The subsystem of superpages system will implement the features corresponding to this new display. The subsystem of back-office system will allow the salespeople to manage the customer information related to this new service.

The development of subsystem under back-office system (from now referred to as ELP system), is assigned to a new developer under an alias Peter, who just recently joined the back-office team, with a fresh perspective and a strong enthusiasm.

The ELP system needed to be developed under the old platform first due to the product launch time. However, due to the anticipated migration to the new platform, the developer developed the system with the intention to migrate it to the new platform later on. This meant that he needed to have a relatively good understand of the new platform to design the system in such a way that minimize the migration efforts.

At any rate, the development of ELP system under the old platform began in early June and finished around the end of July. The system was to be in operation until the system (hardware, operating systems, application servers etc.) for the new platform and the ELP system under the new platform become available. It is the development of the ELP system under the new platform that will be the focus of our quantitative example in the sections that follow.

Data collection procedure

System parameters for ELP were collected through several interactions with the developer. First, we present the work-generation model and its features to the developer. The focus was on clarifying the concepts and terminology rather than the model details.
Since the model involves several concepts that can be used in several ways, the presentation helped developer understand the type of data we need from him.

This presentation in fact was given to the whole back-office development team long before we engaged in the data collection around the end of May 2001, although not before the decision to use SuperCommerce in the pilot study was made. At the time, we had not decided to use ELP in our quantitative example for the model.

After the SuperCommerce project ended, the team resolved major architectural issues and was ready to move on to the real migration of the remaining subsystems. At the time, ELP was one of a few subsystems which were in the pipeline for the migration.

This is due to several factors. Although ELP was developed later, it was developed with the intention of moving to the new platform later. The developer had thought about the migration issues long before it happened and, therefore, was in a very good position to proceed with the migration. The fact that the developer joined the team recently meant that he was relatively free of routine time-consuming tasks which other developers had to tend to. He therefore had more time to spend on the new development like ELP system. In addition, the ELP subsystem is relatively independent of other, making it less vulnerable to changes that occur in other subsystems.

By mid-September, the developers began developing the architecture of ELP under the new platform using the knowledge gained from SuperCommerce. We also decided to use this in our quantitative example at that time. The data collection for ELP in fact ran in parallel with the architectural design. We read the design documents as the developer finished them to learn about the architecture of the system. With his understanding of the model and our understanding of architecture, we engaged in conversations which span several weeks to come up with the model elements and parameters needed for the model.

*Design under new platform*

Figure 5.9 shows the overall design of ELP system (and, in fact, most of other systems of back-office under the new platform). The development efforts primarily focus on the part in the middle, which includes JSP, Servlet, Session Beans, and Entity Beans. These represent the modules to be developed.
Modules can be thought of in terms of where they stand between the user and the data. Modules which are closer to user are those which play key roles in the user interface. Modules which are closer to the data play key roles in manipulating the data in the database system. In this regard, the modules run from JSP, the closest to the user interface, to Servlet, Session Beans, and Entity Beans, which is the closest to the database.

For a system with moderate size, we can think of the functions as consisting of a ‘thread’ of modules which stretch from JSP through Entity Beans. Each thread consists of modules which work together to perform certain operations on the data, such as adding data, modifying data, and so on.

In this case of ELP system, a lot of the uncertainty surrounding the design of the Entity Beans and Session Beans was resolved with the knowledge gained from SuperCommerce. We found that the major uncertainty lies in the library calls of J2EE server. Although a lot of the functionalities reside within methods of each module, these modules still need to make certain calls to the library to perform certain operations. Without a perfect knowledge of the technology, the developer needs to make certain assumptions about how these library calls work (despite the documentation). Such library calls therefore represent uncertain issues to be dealt with in the development.

Figure 5.9 Diagrams depicting various elements (functions, modules, interfaces) of ELP System

A Risk Dynamics Model of Complex System Development
5.7 ELP: System Parameters

*Functions*

The first model parameters we will discuss are the functions. As mentioned, we can think of the functions of the system as those related to the operations on the underlying data. To derive the functions of the system, we ask the developers about how he thinks about the partitioning of the system, in other words, what is the way he thinks the system is divided into from the functional viewpoint. From the answer and the discussions that follow, we found that ELP system can be thought to consist of four functions, each corresponding to specific data operations.

<table>
<thead>
<tr>
<th>Label</th>
<th>Name</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>F₁</td>
<td>Create</td>
<td>Take input from users, create a new items in the database and verify the results</td>
</tr>
<tr>
<td>F₂</td>
<td>Query</td>
<td>Take the query fields from users, send the request for query to the database server, retrieve the data and display to users</td>
</tr>
<tr>
<td>F₃</td>
<td>Update</td>
<td>Take the identification fields from the users, retrieve the data from the user, wait for input from users, send the data to be updated in the database and verify.</td>
</tr>
<tr>
<td>F₄</td>
<td>Delete</td>
<td>Take the identification fields from the users, send the request to delete the data from the database, make other necessary adjustments</td>
</tr>
</tbody>
</table>

*Modules*

The modules of the system are, in fact, the direct output of the design process. We found from the design documents ELP system has many “types” of modules. Some of the module types are in accordance with the J2EE specifications. Some other functions are created by the developer to reflect different roles of modules in ELP system.

Following are the modules types, their abbreviations, and their roles in the system. The sequence is done so that modules closer to user interface appear first. Note that JSP are left out from modules under consideration because it is relatively risk-free from the development view points. This means that JSP is unlikely to be subject to rework like other types of modules.
Table 5.4: Modules of ELP

<table>
<thead>
<tr>
<th>Label</th>
<th>Name</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD</td>
<td>Dispatcher</td>
<td>Interact with the JSP to dispatch the request to corresponding set of object types.</td>
</tr>
<tr>
<td>MH</td>
<td>Helper</td>
<td>Implement some operations which are shared across many modules.</td>
</tr>
<tr>
<td>MF</td>
<td>Façade</td>
<td>Serve as the façade that hide the details of operations from outer-layer modules</td>
</tr>
<tr>
<td>MS</td>
<td>Session Beans</td>
<td>Implement the transaction-oriented data operations, i.e. those set of operations which need to be completed all or nothing.</td>
</tr>
<tr>
<td>ME</td>
<td>Entity Beans</td>
<td>Implement elementary data operations, i.e. read, write, delete, and so on. Each bean typically corresponds with each data type in the database.</td>
</tr>
<tr>
<td>MO</td>
<td>Other Types</td>
<td>Other functions</td>
</tr>
</tbody>
</table>

From the design documents, we identify several modules that are related to ELP. We eliminate some modules which are 'risk-free', i.e. that is not subject to uncertain issues and thus requires little changes or rework. Table 5A.1 and 5A.2 shows the result for the modules.

**Interfaces**

From the foregoing discussion, the early phase prototyping activities removed a lot of the design uncertainty from the development. We found that the library calls represent major uncertainty in this case. Therefore, interfaces in ELP system correspond primarily with library calls.

![Figure 5.10 Relationships between interfaces with modules](image-url)
We found that we can divide the uncertain issues into two types, using the direction of data flow between web tier (related to user interface) and the EJB tier (related to data operations). The first one is related to the uncertainty surrounding library calls that involve inbound data flow, i.e. those which involve the operations that take the data from the user to the database. The second is related to the library calls that involve outbound data flow, i.e. those which involve the operations that take the data from the database out to the users. Table 5.5 shows these interfaces.

Table 5.5: Interfaces of ELP

<table>
<thead>
<tr>
<th>Label</th>
<th>Name</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1$</td>
<td>Inbound data flow</td>
<td>Interfaces related to the sending of information</td>
</tr>
<tr>
<td>$l_2$</td>
<td>Outbound data flow</td>
<td>Interfaces related to the retrieving of information</td>
</tr>
</tbody>
</table>

**Functional mapping**

From our functions and modules, we asked the developer to document the functional mapping, i.e. to mark which modules contribute to a given function. The first set of data is obtained as shown in table 5A.3.

From this, we get deeper into the functional mapping by asking the developer to document the proportion of each module's code base devoted to individual functions and the proportion that are shared across all the functions. The result is shown in table 5A.4.

From this, we generated a new set of modules from the original set by splitting each original module into several modules according to its mapping. The resulting set of modules with the functional mapping is shown in table 5A.6. Note that the functional mapping is neither modular nor nested.

**Impact functions and parameters**

From our inspection of the ELP under the old platform and from our conversation with the developer, we found that the two interfaces can be considered independent. Inbound data flow and outbound data flow are mostly done in separate places in the code base. Therefore, we feel comfortable in making the assumption that impact functions is of additive form, i.e. the impacts resulting from different interfaces are additive.
Impact parameters cannot be derived directly, however. We found that the developer had a hard time estimating how much rework of each module will result from a change in an interface. We found that it is easier for the developer to think of impact parameter in terms of the amount of nominal work and the proportion of change.

Therefore, to estimate the impact parameters, we first ask the developer to estimate the amount of time it takes to create a given module. Then, we ask him to estimate the proportion of the module that will be affected, i.e. need rework, should a change in the interface happen. The impact parameters are obtained as the product of these two parameters. Table 5A.6 shows the result.

*Probability functions and parameters*

We first asked the developer for his best estimate of the probability that a given function will create a change in the interfaces. We got the answer in the following table.

**Table 5.6: Probability parameters**

<table>
<thead>
<tr>
<th></th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>$F_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>0.40</td>
<td>0.00</td>
<td>0.40</td>
<td>0.40</td>
</tr>
<tr>
<td>$I_2$</td>
<td>0.00</td>
<td>0.80</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

The probability parameters are quite intuitive in the following sense.

For what we know, the three functions, $F_1$, $F_3$, $F_4$, which have positive probability parameters on $I_1$ all have operations which require the ‘writing’ of information into the database. When we can to ‘create’ or ‘update’ or ‘delete’ the data, we need to involve the inbound data flow mechanism. Therefore, each of these functions contributes positive parameters to interface $I_1$.

On the other hand, $F_2$, which corresponds to ‘query’, involves the outbound data flow mechanism, $I_2$. This is because the main function of $F_2$ is to get the existing data out to show the user. Correspondingly, function $F_2$ contributes a positive parameter to interface $I_2$.

The probability functions relate to the manner in which a combination of functions results in the effective probability of change to a given interface. In this regard, we extract information by asking the developer about the nature of uncertainty reduction in the process. In other words, we ask how he conceives of the pattern of change in overall uncertainty after successive iterations of working on individual functions.
In asking this question, we expect that if the probability function is of additive form, the answer would be such that the overall uncertainty, i.e. the probability that the interface change will happen after a given iteration, declines in a linear fashion. On the other hand, if it is of multiplicative form, the answer will be that the overall uncertainty declines in a concave manner, i.e. the reduction is lower at the beginning and faster as we move on.

The answer we obtained pointed us to the direction of a concave reduction in overall uncertainty. However, there are many types of probability function that can give rise to such a function. To simplify the analysis, we simply assume that the probability function takes the multiplicative form.

**Risk-mitigating Work**

According to the developer, risk-mitigation activities consist mainly of compiling the code, integrating the code to the system, creating test scenarios, and testing these scenarios to the functions for which the test was designed. The amount of work for compiling and integration, which can be shared across different functions, is minimal compared to the testing activities, which are function-specific.

This means that the risk-mitigation work depends primarily on the functions for a given iteration. The amount of risk-mitigation work is the sum of the risk-mitigation work corresponding to functions in a given iteration. Table 5.7 shows the risk-mitigation corresponding to each function of the system.

<table>
<thead>
<tr>
<th>Risk-mitigation work (man-hr)</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>2.5</td>
<td>4.0</td>
<td>3.0</td>
<td></td>
</tr>
</tbody>
</table>

**System Matrix**

From the above information, we are now in a position to create a system matrix. Table 5.8 shows the resulting system matrix. Due to the additive nature of impact function, we can simplify the matrix by combining modules with the same functional mapping into one module by adding up the impact parameters. From such simplification, we derive the system matrix as shown in table 5.8.
Table 5.8: Risk Matrix of ELP

<table>
<thead>
<tr>
<th></th>
<th>F_1</th>
<th>F_2</th>
<th>F_3</th>
<th>F_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_1</td>
<td>0.40</td>
<td>0.00</td>
<td>0.40</td>
<td>0.40</td>
</tr>
<tr>
<td>l_2</td>
<td>0.00</td>
<td>0.80</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>M_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M_2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M_3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M_4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M_5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M_6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.11: Graphical functional mapping
5.8 ELP: Numerical Solutions

**Solution methodology**

Using the parameters thus obtained, we then proceed to solve the resulting work-minimization problem. We make the observation that the size of the problem is relatively small. Since we have four functions to be allocated to 1, 2, 3, or 4 iterations, the total number of possible allocations is 76. This means that we can easily use the enumeration procedure in solving the problem. And this is exactly the approach we will take in this study.

We perform enumeration of all possible process policies. From these enumerations, we can find the iteration probability and iteration impact for each iteration and each interface, which are then used to calculate the objectives, which help us compare the alternative allocations.

**Objective function**

In contrast to the previous chapter, where we focus on the ‘expected work’, we take advantage of the enumeration procedure in this case to look at the problem with a little more complicated objective function.

To see this, we first note that a given objective function is a function of the risk-mitigation work and the eight binary random variables corresponding to the 2 interfaces and 4 iterations, i.e.

\[ Y = f(L^1, \bar{T}_1, \bar{T}_2, L^2, \bar{T}_2, L^3, \bar{T}_3, L^4, \bar{T}_4) \]

The risk-mitigation work for an iteration is the sum of that corresponding to the functions in the given iteration. From the specification in the previous section, the sum of risk-mitigation work is a constant, i.e. not dependent on iteration policy, i.e.

\[ L^1 + L^2 + L^3 + L^4 = L \]

Each random variable \( \bar{T}_q \) is characterized by its **probability**, \( P_q \) (the probability of change of an interface in an iteration) and its **impact**, \( D_q \) (the sum of impact parameters of that interface for all the modules that need to be reworked in that particular iteration).

From these parameters, we can find the expected rework and the variance of rework corresponding to each random variable.
1. The expected rework is simply the product of probability and impact of a binary random variable.

\[ E(\tilde{I}_q^j \cdot D_q^j) = \Pr[\tilde{I}_q^j = 1] \cdot D_q^j = P_q^j \cdot D_q^j \]

2. The variance of rework is the product of probability, one minus probability, and impact of a binary random variable.

\[ V(\tilde{I}_q^j \cdot D_q^j) = E[(\tilde{I}_q^j \cdot D_q^j)^2] - (E(\tilde{I} \cdot D))^2 \\
= \Pr[\tilde{I}_q^j = 1] \cdot D_q^j - (\Pr[\tilde{I}_q^j = 1] \cdot D_q^j)^2 \\
= P_q^j \cdot (1 - P_q^j) \cdot D_q^j \\
\]

From this, we construct the function as the sum of risk-mitigation work, the expected rework, and a scaled variance of rework, which can be written as

\[ Y = \sum_{j=1}^{4} L_j + \sum_{j=1}^{4} \sum_{q=1}^{2} (P_q^j \cdot D_q^j + \alpha \cdot P_q^j \cdot (1 - P_q^j) \cdot (D_q^j)^2) \\
= L + \sum_{j=1}^{4} \sum_{q=1}^{2} P_q^j \cdot D_q^j + \alpha \cdot \left( \sum_{j=1}^{4} \sum_{q=1}^{2} P_q^j \cdot (1 - P_q^j) \cdot (D_q^j)^2 \right) \\
= L + E(W) + \alpha \cdot V(W) \\
\]

As can be seen, the objective function turns out to be a function of \( E(W) \), \( V(W) \), and the parameter \( \alpha \). We believe that the reasonable value of parameter \( \alpha \) should be positive as the variability of expected rework is an undesirable thing to the development process. In fact, the parameter \( \alpha \) reflects the degree of risk aversion of the developer. The higher the value of \( \alpha \), the more risk-aversion is implied.

From this, there is a possibility that we might get different optimal solutions for different values of \( \alpha \) as some policies might give a lower expected value with a high variance whereas others give a higher expected value with a low variance. In fact, we expect to see how different policies might be optimal under different degree of risk aversion.
Numerical results

Figure 5.12 shows the plot of the values of $E(W)$, $V(W)$ for all the alternative allocations. We encounter an expected result that one candidate allocation dominates all others in that it has the lowest expected rework and the lowest variance of rework among all feasible solutions. This means that this candidate allocation will be optimal for any reasonable, i.e. non-negative, value of $\alpha$. In other words, the allocation is a robust optimal solution to the problem.

![Figure 5.12 Expected rework and Variance of rework for all possible policies](image)

Such candidate allocation is shown in Table 5.9.

Table 5.9 Optimal allocation

<table>
<thead>
<tr>
<th></th>
<th>Iteration 1</th>
<th>Iteration 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions</td>
<td>$F_2$</td>
<td>$F_1, F_3, F_4$</td>
</tr>
</tbody>
</table>

We first make an observation of the fact that the optimal policy is a two-iteration process, as opposed to four-iteration process. This might seem counter-intuitive at first. Since the sum of the risk-mitigation work is constant regardless of the number of iterations, this should prompt us to use as many iterations as possible to minimize the amount of rework.
This is, however, complicated by the fact that the probability function is of multiplicative form. From section 4.6, we found that such probability function tend to favor the grouping of functions which affect the same interface. This is in fact what happens in this case. We see that F₁, F₃, and F₄ all affect interface I₁. And they are all grouped within the same iteration (no. 2) in the optimal policy.

Figure 5.13 shows the range (minimum to maximum) of expected rework and variance of rework obtained from all possible iteration policies with 1, 2, 3, and 4 iterations. We see that the minimum expected rework and the minim variance of rework for two-iteration are lower than the corresponding values for other cases (1, 3, and 4 iterations). In this case, therefore, the optimal number of iterations is two, a result of the multiplicative probability function.

![Figure 5.13 Expected rework and variance of rework for different iteration numbers](image)

From our optimal solution, we discuss with the developer regarding the iteration policy he had in mind. As it turned out, we found that the optimal policy as suggested by our model is in fact one of the several possible iteration policies he is considering. Such policies are the result of his own intuition given his knowledge of the system.

After the explanation of how we came up with this allocation as the optimal one, and the relative merit of this as compared to other allocations he is considering, we convince him to choose the iteration plan according to our optimal solution, which he did. This saved him a good proportion of work in the process, though not much by absolute terms due to the small size of the system. In this fashion, our model helps him choose among several policies he is considering by giving a quantitative support for his decisions.
5.9 Conclusions

Continuing from the previous chapter, which focuses on the analytical aspect of our model, this chapter explores the practical aspect of our model through a case study. The role of this chapter is to provide an example to the applications of the model to a real development process.

The migration of the back-office system of superpages.com to a new platform provides us with a context for understanding the issues related to the model applications and those related to iterative process in general. In this particular example, the technical uncertainty surrounding the new platform is driving the decision to use the iterative process as the development model. We have tried to apply the model, conceptually and mathematically, to aid in the project planning and execution.

I have found that the decisions taken by the system architect in the early phase of migration reflect a mental framework much similar to what we have in the model. The decision to choose SuperCommerce subsystem in the pilot development is based on a trade-off between the amount of uncertainty reduction to be gained and the amount of impact it will have. More specifically, SuperCommerce reduced a significant amount of the uncertainty in the underlying EJB designs while adding a relatively small impact, due to its compact size. This reflects very much the result obtained in the previous chapter regarding the uncertainty-impact trade-off.

In addition, I have also performed a detailed study of one of the subsystems that were migrated soon after the pilot study ended. I mapped the ELP subsystem to our model framework through a series of questionnaire and interviews with the developer. I successfully identified the model elements, i.e. system functions, interfaces, and modules, and derived all the model parameters. The resulting model help the developer resolve his ambiguity in the iteration plan through the numerical results thus obtained. He proceeded according to the solution, thereby saving a certain amount of work in the process.

In a sense, the concept of iterative development, or more particular the spiral process model, is nothing new in the eyes of the developers. Few people did not hear about it. In fact, such process seems to be viewed as the typical mode of development, particularly for the ongoing development efforts as in the case of superpages.com.

The planning of the iterative process, however, is normally done in a rather informal fashion. I have found in this case that an expert is relied on to make a critical decision in the process
planning. Experts have deep knowledge about the architecture and the resulting dependencies of system and are in better positions to make a call on such important matters.

I found that the use of a formal model can be of great assistance in the planning process. The very activity of deriving model elements and parameters helps the developer think through a lot of the issues in the development. This makes them clearer of the relative importance of issues in such a process. In addition, in the case when the system is highly complex or when the developers have relatively little experience in designing the system, such a model can serve as a tool that help the developers make a better, more informed decision in planning the process.

I find that the major challenges in applying the model lie in the identification of model elements. Functions and interfaces to be put in the model are not readily available. This necessitates some thoughtfulness on the part of the analyst, who might or might not be an expert in the system under study, to carefully identify system elements that matter to the process planning.
5.10 Appendix

Table 5A.1: Module types and results of interviews in two rounds

<table>
<thead>
<tr>
<th>Name</th>
<th>Label</th>
<th>Characteristics</th>
<th>Number of modules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1&lt;sup&gt;st&lt;/sup&gt; round</td>
</tr>
<tr>
<td>Dispatcher</td>
<td>MD</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Handler</td>
<td>MH</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Façade</td>
<td>MF</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Session Beans</td>
<td>MS</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Entity Beans</td>
<td>ME</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>Other Classes</td>
<td>MO</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td><strong>27</strong></td>
</tr>
</tbody>
</table>
Table 5A.2: Details of modules

<table>
<thead>
<tr>
<th>Label</th>
<th>Name</th>
<th>Related to ELP</th>
<th>No uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MD1</td>
<td>BackOfficeDispatcher</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MH1</td>
<td>SubscriptionHandler</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MH2</td>
<td>ELPHandler</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MH3</td>
<td>NAHandler</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MH4</td>
<td>NAMHandler</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MH5</td>
<td>ApprovalHandler</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MF1</td>
<td>SubscriptionFacade</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MF2</td>
<td>ELPFacade</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MF3</td>
<td>NAFacade</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MF4</td>
<td>NAMFacade</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MF5</td>
<td>ApprovalFacade</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MS1</td>
<td>ListingManager</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MS2</td>
<td>ELPSetManager</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MS3</td>
<td>NAMembership</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MS4</td>
<td>Approver</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ME1</td>
<td>Subscription</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ME2</td>
<td>Style</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ME3</td>
<td>ExtraLine</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ME4</td>
<td>Priority</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ME5</td>
<td>NA</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ME6</td>
<td>NAM</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ME7</td>
<td>ServInstace</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ME8</td>
<td>BLOB</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ME9</td>
<td>Pres</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MO1</td>
<td>ListingAttributes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MO2</td>
<td>ELPListingAttributes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MO3</td>
<td>StyleAttributes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MO4</td>
<td>ExtraLineAttributes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MO5</td>
<td>ELPAttributes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MO6</td>
<td>NAAttributes</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MO7</td>
<td>NAMAttributes</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MO8</td>
<td>ELPDisplay</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MO9</td>
<td>ELPDisplayElement</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MO10</td>
<td>ELPDisplayFactory</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MO11</td>
<td>ELPSelections</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MO12</td>
<td>ELPUtility</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
### Table 5A.3: Functional mapping

<table>
<thead>
<tr>
<th></th>
<th>F₁ Create</th>
<th>F₂ Query</th>
<th>F₃ Update</th>
<th>F₄ Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MH2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MF2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MS1</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>MS2</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ME2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ME3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ME4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ME7</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ME9</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MO8</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MO9</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MO10</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MO11</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>MO12</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

### Table 5A.4: Proportion of each module allocated to individual functions and shared functionality

<table>
<thead>
<tr>
<th></th>
<th>F₁ Create (%)</th>
<th>F₂ Query (%)</th>
<th>F₃ Update (%)</th>
<th>F₄ Delete (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MH2</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>MF2</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>MS1</td>
<td>0</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS2</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>ME2</td>
<td>25</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>ME3</td>
<td>25</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>ME4</td>
<td>25</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>ME7</td>
<td>25</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>ME9</td>
<td>25</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>MO8</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MO9</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MO10</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MO11</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MO12</td>
<td>50</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Code (man-hr)</td>
<td>Test (man-hr)</td>
<td>Code + Test (man-hr)</td>
<td>Proportion Impact by I&lt;sub&gt;1&lt;/sub&gt; change (into database)</td>
</tr>
<tr>
<td>-----</td>
<td>----------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>MD1</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>.10</td>
</tr>
<tr>
<td>MH2</td>
<td>8</td>
<td>6</td>
<td>14</td>
<td>.20</td>
</tr>
<tr>
<td>MF2</td>
<td>12</td>
<td>10</td>
<td>22</td>
<td>.25</td>
</tr>
<tr>
<td>MS1</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>MS2</td>
<td>24</td>
<td>12</td>
<td>36</td>
<td>.10</td>
</tr>
<tr>
<td>ME2</td>
<td>12</td>
<td>8</td>
<td>20</td>
<td>.10</td>
</tr>
<tr>
<td>ME3</td>
<td>12</td>
<td>8</td>
<td>20</td>
<td>.10</td>
</tr>
<tr>
<td>ME4</td>
<td>12</td>
<td>8</td>
<td>20</td>
<td>.10</td>
</tr>
<tr>
<td>ME7</td>
<td>12</td>
<td>8</td>
<td>20</td>
<td>.20</td>
</tr>
<tr>
<td>ME9</td>
<td>12</td>
<td>8</td>
<td>20</td>
<td>.20</td>
</tr>
<tr>
<td>MO8</td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>MO9</td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>MO10</td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>MO11</td>
<td>8</td>
<td>4</td>
<td>12</td>
<td>.25</td>
</tr>
<tr>
<td>MO12</td>
<td>16</td>
<td>12</td>
<td>28</td>
<td>.20</td>
</tr>
</tbody>
</table>
### Table 5A.6: Impact parameters and functional mapping of splitted modules

<table>
<thead>
<tr>
<th></th>
<th>Mapping</th>
<th>Impact (man-hr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F_1$ Create</td>
<td>$F_2$ Query</td>
</tr>
<tr>
<td>MD1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MH2-S</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MH2-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MH2-2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MH2-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MH2-4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MF2-S</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MF2-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MF2-2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MF2-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MF2-4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MS1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MS2-S</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MS2-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MS2-2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MS2-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MS2-4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME2-S</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ME2-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ME2-2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ME2-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME2-4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME3-S</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ME3-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ME3-2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ME3-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME3-4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME4-S</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ME4-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ME4-2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ME4-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME4-4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME7-S</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ME7-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ME7-2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ME7-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME7-4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME9-S</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ME9-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ME9-2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ME9-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME9-4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MO8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MO9</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MO10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MO11</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MO12-S</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MO12-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MO12-2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MO12-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MO12-4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Chapter 5: Case Study: SuperPages.com*
CHAPTER 6: NUMERICAL APPROACH

6.1 Introduction

This chapter will explore the remaining issues of our model, that of the practical method for deriving the solution of our model. We first take a look at the complexity analysis of the work-minimization problem. In particular, we prove that the problem is in fact NP-complete, which justifies the use of heuristics for solving the problem. Then, we propose several heuristics that can be used to tackle the work-minimization problem. We continue with the numerical experiments that test these heuristics before we conclude this chapter.

6.2 Complexity Analysis

We first investigate the complexity of the work-minimization problem to determine the necessity of using heuristics in solving the problem. More specifically, in this section, I will prove the NP-completeness of the problem.

We approach this using the conventional method as described in Gary and Johnson (1979). In other words, we will prove that a known NP-complete problem can be transformed via a polynomial transformation to our problem.
The known NP-complete problem in our proof is called MINIMUM SUM OF SQUARES in Gary and Johnson (1979). The details of this problem are shown below.

**INSTANCE**: Finite set \( A \), a size \( s(a) \in \mathbb{Z}^* \) for each \( a \in A \), positive integers \( K \leq |A| \) and \( J \).

**QUESTION**: Can \( A \) be partitioned into \( K \) disjoint sets \( A_1, A_2, \ldots, A_K \) such that

\[
\sum_{i=1}^K \left( \sum_{a \in A_i} s(a) \right)^2 \leq J
\]

**Comment**: NP-complete in the strong sense. NP-complete in the ordinary sense and solvable in pseudo-polynomial time for any fixed \( K \). Variants in which the bound \( K \) on the number of sets is replaced by a bound \( B \) on either the maximum set cardinality or the maximum total set size are also NP-complete in the strong sense [Wong and Yao, 1976]. In all these cases, NP-completeness is preserved if the exponent 2 is replaced by any fixed rational alpha > 1.

**Figure 6.2 Description of MINIMUM SUM OF SQUARES (from Gary & Johnson 1979)**
On the other hand, the problem we will transform into is the $J$-iteration work-minimization problem for a single-interface system with modular functional mapping and a special condition on the system parameters (as will be explained later). (Note that this $J$ is different from that in Figure 6.2) This problem is a subset of the set of work-minimization problem for all possible systems and therefore serves as the basis of our NP-completeness proof for work-minimization problem.

First, we take a look at the formulation of the work-minimization problem for a single-interface system with modular functional mapping and general system parameters.

$$W^* = \min_{S_1, S_2, \ldots, S_J} \left( \sum_{j=1}^{J} L + \sum_{n \in S_j} p_n \cdot \sum_{n \in S_j} d_n \right)$$

$$= J \cdot L + \min_{S_1, S_2, \ldots, S_J} \left( \sum_{n \in S_j} p_n \cdot \sum_{n \in S_j} d_n \right)$$

Because risk-mitigation work is fixed, we can solve the problem by minimizing the rework only. Now, we specify a special condition that the probability parameter of a function/module is identical to its impact parameters, i.e.

$$p_n = d_n \quad \forall n \in \Omega$$

The problem can then be rewritten as a rework-minimization problem as

$$R^* = \min_{S_1, S_2, \ldots, S_J} \left( \sum_{j=1}^{J} \sum_{n \in S_j} p_n \cdot \sum_{n \in S_j} p_n \right)$$

$$= \min_{S_1, S_2, \ldots, S_J} \left( \sum_{j=1}^{J} \sum_{n \in S_j} p_n \cdot \sum_{n \in S_j} p_n \right)$$

$$= \min_{S_1, S_2, \ldots, S_J} \left( \sum_{j=1}^{J} P(S^j) \cdot P\left( \bigcup_{i=1}^{J} S^i \right) \right)$$

Here, for notational convenience, we define $P(S^j) = \sum_{n \in S^j} p_n$

Next, observe that we can write the product of two numbers in terms of their squares as follows.
\[ [P(S_A)] [P(S_B)] = \frac{1}{2} \left\{ [P(S_A) + P(S_B)]^2 - [P(S_A)]^2 - [P(S_B)]^2 \right\} \]
\[ = \frac{1}{2} \left\{ [P(S_A \cup S_B)]^2 - [P(S_A)]^2 - [P(S_B)]^2 \right\} \]

From this, we rewrite each term under the summation as

\[ P(S^j) \cdot P\left( \bigcup_{i=1}^{j} S^i \right) = P(S^j) \cdot \left\{ P(S^j) + P\left( \bigcup_{i=1}^{j-1} S^i \right) \right\} \]
\[ = P(S^j)^2 + P(S^j) \cdot P\left( \bigcup_{i=1}^{j-1} S^i \right) \]
\[ = P(S^j)^2 + \frac{1}{2} \left\{ P\left( \bigcup_{i=1}^{j} S^i \right)^2 - P(S^j)^2 - P\left( \bigcup_{i=1}^{j-1} S^i \right)^2 \right\} \]
\[ = \frac{1}{2} \left\{ P\left( \bigcup_{i=1}^{j} S^i \right)^2 + P(S^j)^2 - P\left( \bigcup_{i=1}^{j-1} S^i \right)^2 \right\} \]

Then, we can write the objective of the rework-minimization problem again as

\[ \sum_{j=1}^{J} P(S^j) \cdot P\left( \bigcup_{i=1}^{j} S^i \right) = \frac{1}{2} \sum_{j=1}^{J} \left\{ P\left( \bigcup_{i=1}^{j} S^i \right)^2 + P(S^j)^2 - P\left( \bigcup_{i=1}^{j-1} S^i \right)^2 \right\} \]
\[ = \frac{1}{2} \sum_{j=1}^{J} P(S^j)^2 + \frac{1}{2} P\left( \bigcup_{i=1}^{J} S^i \right)^2 \]
\[ = \frac{1}{2} \sum_{j=1}^{J} P(S^j)^2 + \frac{1}{2} P^2 \]

Notice the cancellation of the first and the third terms between two successive index \( j \). \( P \) is simply the sum of all probability parameters, which we know as a constant.

But this expression in the summation is nothing but the objective in the MINIMUM SUM OF SQUARES as we mentioned earlier. This problem is NP-complete. Hence, we have proven that we can make a polynomial (in fact, a trivial one) transformation from a known NP-complete problem into a subset of work-minimization problem. Thus, the work-minimization problem as a whole is said to be of NP-completeness, the result we set out for in this section.
6.3 Problem Structure

*Problem Matrices*

In using heuristics to solve the work-minimization problem, we benefit from using the matrix-form of the problem. We start from the integer-programming formulation in section 3.5

\[
W^j = L \cdot J + \min_{y_1^j, \ldots, y_n^j} \sum_{j=1}^{J} \sum_{j=1}^{J} y_j^i \cdot G \cdot z_j^i
\]

s.t. \[y_n^j \leq \sum_{i=1}^{J} z_n^i \quad \forall m \in F_n \text{ for } \forall j, n\]

\[\sum_{j=1}^{J} y_n^j = 1 \quad \forall n\]

\[\sum_{j=1}^{J} z_n^j = 1 \quad \forall m\]

\[y_j^i, z_n^i \in \{0,1\} \quad \forall n, m, j\]

where \( G = P'D' \) and the decision parameters are defined as

\[
y_j^i = \begin{bmatrix}
y_1^i \\
y_2^i \\
\vdots \\
y_n^i 
\end{bmatrix}
\]

\[
y_n^j = \begin{cases}
1; & \text{function } n \text{ in iteration } j \\
0; & \text{otherwise}
\end{cases}
\]

\[
z_j^i = \begin{bmatrix}
z_1^j \\
z_2^j \\
\vdots \\
z_M^j 
\end{bmatrix}
\]

\[
z_n^j = \begin{cases}
1; & \text{module } m \text{ in iteration } j \\
0; & \text{otherwise}
\end{cases}
\]

To simplify the formulation, we define

\[
y = \begin{bmatrix}
y^1 \\
y^2 \\
\vdots \\
y^J
\end{bmatrix} \quad \text{(function vector)}
\]
\[
\mathbf{z} = \begin{bmatrix}
  z^1 \\
  z^2 \\
  \vdots \\
  z^J
\end{bmatrix}
\quad \text{(module vector)}
\]

\[
\mathbf{\Gamma} = \begin{bmatrix}
  P' D' & 0 & 0 & 0 \\
  P' D' & P' D' & 0 & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  P' D' & P' D' & \cdots & P' D'
\end{bmatrix}
\quad \text{(rework matrix)}
\]

From this, the objective can be rewritten as
\[
\sum_{j=1}^{J} \sum_{i=1}^{J} y^j_i \cdot G \cdot z^j_i = y \cdot \mathbf{\Gamma} \cdot \mathbf{z}
\]

In addition, other constraints can be written in the matrix form as well

1. Unity Sum Constraint

\[
\mathbf{A}_N \cdot \mathbf{y}' = \mathbf{1}_N \\
\mathbf{A}_M \cdot \mathbf{z} = \mathbf{1}_M
\]

2. Functional Mapping Constraint

\[
\text{From } y_n^j \leq \sum_{i=1}^{J} z_m^i \quad \forall n, m \text{ s.t. } h_{m,n} = 1
\]

We can write \( \mathbf{\Phi}_y \cdot \mathbf{y}' + \mathbf{\Phi}_y \cdot \mathbf{z} \leq \mathbf{0} \)

Thus, the problem can be written simply as

\[
\min_{y,z} \quad y \cdot \mathbf{\Gamma} \cdot \mathbf{z}
\]

s.t.
\[
\mathbf{A}_N \cdot \mathbf{y}' = \mathbf{1}_N \\
\mathbf{A}_M \cdot \mathbf{z} = \mathbf{1}_M \\
\mathbf{\Phi}_y \cdot \mathbf{y}' + \mathbf{\Phi}_y \cdot \mathbf{z} \leq \mathbf{0} \\
y, z \quad 0 - 1 \text{ integer}
\]
Neighborhood Structure

The algorithms that we will explore in the following sections involve the concept of neighborhood of a candidate solution. This represents the set of potential solutions we should explore from a given candidate solutions. In most cases, the neighborhood structure determines the efficacy of the resulting algorithms and needs to be selected carefully.

The possible alternatives for selecting neighborhood are quite numerous. In this study, we will take advantage of the structure of the work-minimization problem in coming up with neighborhood structure that is easy to derive and that preserves the feasibility of the solution.

More specifically, we look at the set of adjacent movements, which represent the movements of functions and/or modules across an adjacent pair of iterations, as shown in Figure 6.3.

![Diagram of adjacent movements](image)

**Figure 6.3 Adjacent movements**

As can be noticed, there are two types of adjacent movements. The forward movements involve movements to a later iteration, or the iteration with higher index. The backward movements, on the other hand, involve movements to an earlier iteration, the iteration with lower index.
In a system with modular functional mapping, such adjacent movements are easy to derive due to the one-to-one correspondence between functions and modules. For a system with general functional mapping, this can be tricky. We need to make sure that we move a combination of functions and modules in a way that preserve the feasibility of the resulting candidate solution.

To do this, we propose the methods for deriving adjacent movements as shown in Figure 6.4 and Figure 6.5 for forward movement and backward movements, respectively.

![Figure 6.4 Procedure for generating adjacent forward movements](image)

![Figure 6.5 Procedure for generating adjacent backward movements](image)

As can be seen, forward adjacent movements center on modules while backward adjacent movements center on functions. The procedure proposed make it easy for us to go through each function (module), find all its associated modules (functions) in that iteration, and if necessary, take an additional set of functions (modules) which are implicitly taken anyway to the destination iteration.

This procedure will be used in all the heuristics that follow to generate neighborhood structure.
6.4 Descriptions of Heuristics

Gradient-based Local Search

The first algorithm we consider is a local search based on the gradient of the objective. This type of algorithm is used to tackle optimization problems with nonlinear objective.

The general idea of the algorithm is to begin with a candidate solution to the problem. In each cycle, we use the candidate solution to calculate the gradient of the objective at that point. We can use the information derived from gradient to determine the direction we should move to the next candidate solution. Then, we repeat the step all over again until we converge to a particular solution.

![Diagram](image)

**Figure 6.6** The calculate-update cycle of gradient-based local search algorithm

There are two important aspects of this algorithm we would like to mention here. The first important aspect has to do with the ease of calculation of the gradient. To effectively apply this algorithm, the gradient should be easy to compute at any point of the solution.

In fact, it turns out that this is the case in our work-minimization problem. To see this, we first look at the objective of the work-minimization problem as expressed in section 6.2

\[ R(y,z) = y \cdot \Gamma \cdot z \]

Note that this has the quadratic form. The gradient of the objective can be found easily as follows
\[ R_y(z_0) = \Gamma \cdot z_0 \]
\[ R_z(y_0) = y_0 \cdot \Gamma \]

From this, we can write a problem where the objective is linearized around the point \((y_0, z_0)\) as follows

\[
\begin{align*}
\min_{y,z} & \quad (\Gamma \cdot z_0) \cdot y' + (y_0 \cdot \Gamma) \cdot z \\
\text{s.t.} & \quad A_N \cdot y' = 1_N \\
& \quad A_M \cdot z = 1_M \\
& \quad \Phi_y \cdot y' + \Phi_z \cdot z \leq 0 \\
& \quad y, z \quad 0-1 \text{ integer}
\end{align*}
\]

The second aspect has to do with the update mechanism, i.e. given the gradient, what kind of information can we derive from it and how we can use the information to update the candidate solution to the problem?

To answer this, we first state that the update mechanism used here is based on the assumption that the movements caused by the mechanism involve adjacent iterations. In other words, we only move functions and modules between two adjacent iterations.

In fact, such restriction allows us to use the procedure for finding a set of adjacent movements described in Section 6.3. Such procedure gives out a set of adjacent movements from a given candidate allocation, i.e. the sets of functions and modules allocated to the iterations.

We can readily use the gradient calculated above to help us decide among these possible adjacent movements. We first note that each candidate adjacent movement calls for the movement of a set of functions and a set of modules from an iteration to the adjacent iteration. This adjacent iteration can be prior to or after the iteration in question.

Without loss of generality, let us assume that we are considering the forward movement, i.e. the move of a set of functions and the corresponding set of modules from a given iteration, say iteration \(j\), to the next iteration \(j+1\). Let the set of functions and modules to be moved be \(\Delta\) and \(S'\), respectively. Also, let \(\nabla x\) denote the gradient of the decision variable \(x\).

We note that the movement from iteration \(j\) to iteration \(j+1\), from the point of view of the decision parameter, involves turning off, i.e. changing to zeros, all decision variables.
corresponding to functions and modules in iteration \( j \) and turning on, i.e. changing to ones, all decision variables corresponding to functions and modules in iteration \( j-1 \).

Since our objective is simply the weighted sum of the zero-one variable corresponding to function, module and iteration, the difference in objective (before and after the movement) can be written in terms of the difference of coefficients. In this case, we can write the difference as

\[
\delta(\Delta, S) = \sum_{n \in \Delta} \left( \nabla_{x_n^{r+1}} - \nabla_{x_n^r} \right) + \sum_{m \in S} \left( \nabla_{z_{m}^{r+1}} - \nabla_{z_{m}^r} \right)
\]

In other words, we can write the difference in objective that results from a given adjacent movements as a function of the gradients derived from the current candidate solution. Since our goal is to ‘minimize’ the objective function, I will use the greedy approach to pick the adjacent movements which give us the greatest negative objective difference.

We therefore come up with the update procedure for the gradient-based algorithm. To pick the next candidate solution, we first calculate the objective difference \( \delta(\Delta', S') \) for each adjacent movement. We pick the adjacent movement which gives us the largest negative objective difference. From such adjacent movement, we can then calculate the next candidate solution, and repeat the cycle all over again (i.e. calculate gradient, use the gradient to calculate the objective difference, pick one of the adjacent movement, then calculate the next candidate solution.)

Such algorithm will entail one adjacent movement at a time. However, when we have numerous iterations, there is a way that we can combine several adjacent movements in one cycle without violating the functional mapping constraint.

![Diagram of Best Backward and Forward Movements](image)

**Figure 6.7 Best backward movements and best forward movements**

*Chapter 6: Numerical Approach*
From figure 6.x, we first calculate the best forward movements and the best backward movements for each pair of adjacent iterations, where 'the best' in this case means the movement with the largest negative objective difference.

From such movements, we would like to combine a set of them with two conditions

1. First, we do not violate the functional mapping. We can do this by selecting the movements which do not 'cross'. In other words, we pick a set which does not involve the forward movement and the backward movement of the same pair of iterations.

2. In addition to that, we would like to pick them in such a way that results in the largest negative objective difference, which in this case is simply the sum of the objective difference of constituent movements.

These two objectives can be met by transforming this into a shortest-path problem as shown in figure 6.8. Here, the arc length is defined by its sink node (where the arrow head is at). Each sink node in turn corresponds with the 'best' movements as mentioned earlier. Nodes on the top of the chart corresponds to the best forward movements; those at the bottom, the best backward movements. Note that the link between the top and the bottom nodes are in such a way that the crossing do not occur.

Here, it is easy to see that a path that runs from the left dummy node and the right dummy node corresponds to a set of movements which do not involve crossing. In addition, the length of the path, here the sum of the arc lengths, correspond to the sum of objective difference corresponding to the set of movements of that path.

![Figure 6.8 Shortest-path corresponding to set of best movements](image-url)
Figure 6.9 summarize the procedure for gradient-based algorithm described thus far.

**Pseudo-Program**

1. Begin with an initial guess for $y_i$ and $z_i$, Set $i = 0$

2. Calculate the gradient corresponding to $y_i$ and $z_i$

   \[ R_y(z_i) = \Gamma \cdot z_i \]

   \[ R_z(y_i) = y_i \cdot \Gamma \]

3. Generate the set of adjacent movements using procedure described in section 6.3.

4. Calculate the best backward movements and the best forward movements for each pair of iterations in terms of the objective difference

   \[ \delta (\Delta, S) = \sum_{n \in A} (\nabla y_n^{i+1} - \nabla y_n^i) + \sum_{n \in B} (\nabla z_n^{i+1} - \nabla z_n^i) \]

5. Construct a tree that correspond to the best movements as Figure 6.x

6. Find the shortest-path from the tree.

7. Find the set of movements corresponding to the shortest path.

8. Set $i = i + 1$

9. Generate the next candidate solution $y_i, z_i$ from the shortest path

10. Repeat from step 2 using the new candidate solution

---

**Figure 6.9 Gradient-based algorithm**
Simulated-Annealing Algorithm

The next algorithm to be considered in this chapter is the simulated annealing algorithm, which is widely used to solve the general types of combinatorial optimization problems.

This algorithm involves repeated cycles of updating mechanism. From a given candidate solution, we can generate its neighbors, which represent the possible next candidate solutions. Associated with each neighbor is the cost which can be calculated from the objective function. Based on the costs thus obtained, we calculate the transition probability to each neighbor. From the probability, we can randomize to generate the next candidate solution before repeating the cycle. Figure 6.10 summarizes operations in each cycle.

![Figure 6.10 The simulated-annealing local search algorithm](image)

The neighbors are given by all the adjacent movements as explained in section 6.3 and shown in Figure 6.11

![Figure 6.11 Neighbors consist of all the adjacent movements explained in section 6.3](image)
Thus, the neighbors consist of all the adjacent forward movements from iteration 1, all the adjacent forward movements from iteration 2, ..., all the adjacent forward movements from iteration J-1, all the adjacent backward movements from iteration 2, ..., all the adjacent backward movements from iteration J. This is viewed in contrast to the gradient-based local search algorithm in which we pick the best adjacent movements for a given pair of iterations and then try to combine them without ‘crossing’. Here, each of all the adjacent movements represents a neighbor to the current candidate solution.

The cost, on the other hand, is given by the objective.

\[ C(j) = C(y, z) = R(y, z) = y \cdot \Gamma \cdot z \]

where the decision parameters \( y, z \) correspond to candidate solution \( j \). Assume that the number of neighbors is \( |R| \), the transition probability is calculated as follows.

\[
P_{n}(c) = \begin{cases} 
\frac{1}{|R|} & A_{n}(c) \\
1 - \sum_{i=1 \atop i \neq n}^{|R|} \frac{1}{|R|} A_{i}(c) & \end{cases}
\]

\[
A_{y} = \begin{cases} 
\exp\left( -\frac{C(j) - C(i)}{c} \right) & \text{if } C(j) > C(i) \\
1 & \text{if } C(j) \leq C(i) 
\end{cases}
\]

Here, index \( i \) denotes the current candidate solution, where index \( j \) denotes one of the neighbors.

Parameter \( c \) represents the cooling parameter, which changes as we proceed with the algorithm. In our algorithm, we assume that the cooling schedule is given by a geometric series as follows.

\[
c_{0} \text{ represents the original cooling parameter}
\]

For the \( i^{th} \) cycle, \( c_{i} = c_{i-1} \cdot \alpha \)

**Figure 6.12: Cooling schedule**

Figure 6.13 summarize the procedure for simulated annealing algorithm described thus far.
**Pseudo-Program**

1. Begin with an initial guess for $y_i$ and $z_i$, Set $i = 0$

2. Set $c = c_0$.

3. Generate the set of adjacent movements from $y_i$ and $z_i$ using procedure described in section 6.3.

4. Calculate the cost and then the probability of each of the $|R|$ adjacent movements (denoted $j$, as opposed to the original point which is denoted $i$) according to

$$P_i(c) = \begin{cases} \frac{1}{|R|} A_j(c) \\ \frac{1}{1 - \sum_{j \in R} \frac{1}{|R|} A_j(c)} \end{cases}$$

$$A_j = \begin{cases} \exp \left( \frac{-C(j) - C(i)}{c} \right) & \text{if } C(j) > C(i) \\ 1 & \text{if } C(j) \leq C(i) \end{cases}$$

5. Set $i = i + 1$

6. Generate a random transition to a new candidate solution $y_i$, $z_i$ according to the foregoing probability.

7. Set $c_i = c_{i-1} * \alpha$

8. Repeat from step 3 using the new candidate solution

---

Figure 6.13 Simulated annealing algorithm
Lower-bound Approximation

While the previous two sections propose algorithms that aim to provide the best candidate solution by preserving the feasibility of the original problem, this section algorithm is aimed to provide a lower bound to the problem. To achieve this, we need to relax some constraints to make the problem easier to solve. To see this, we first take a look at the formulation of the work-minimization problem derived in section 6.2

\[
\begin{align*}
\min_{\gamma, z} & \quad y \cdot \Gamma \cdot z \\
\text{st.} & \quad A_N \cdot y' = 1_N \\
& \quad A_M \cdot z = 1_M \\
& \quad \Phi_y \cdot y' + \Phi_y \cdot z \leq 0 \\
& \quad y, z \quad 0 \text{-} 1 \text{ integer}
\end{align*}
\]

The following factors make this problem a hard one to solve using the traditional method

1. Nonlinear objective, \( y \cdot \Gamma \cdot z \).

2. Functional mapping constraint: \( \Phi_y \cdot y' + \Phi_y \cdot z \leq 0 \)

3. Integrality constraint, \( y, z \quad 0 \text{-} 1 \text{ integer} \)

In section 6.4, we have shown how to solve this problem by linearizing the objective and using repeated cycles of update-calculate so that the solution converge to a point.

In this section, we are interested in finding the lower bound to the problem. We therefore focus our attention in relaxing either or both of the constraints represented by 2. and 3, keeping the option of relaxing the linear objective (no. 1) in mind.

We first try relaxing the functional mapping constraint represented by no. 2. Using the lagrangian relaxation, we reduce the above problem to
\[
\begin{align*}
\min_{y,z} & \quad y \cdot \Gamma \cdot z + p \cdot (\Phi_y \cdot y' \cdot \Phi_y \cdot z) \\
\text{subject to} & \\
\Lambda_N \cdot y' &= 1_N \\
\Lambda_M \cdot z &= 1_M \\
y, z &\in 0-1 \text{ integer} \\
p &\geq 0
\end{align*}
\]

Here, \( p \) is the lagrangian vectors for functional mapping constraints. In the case where the objective is linear, we would be able to solve the resulting problem using the sub-gradient optimization technique. In fact, if this is the case, the relaxed problem will be a trivial one due to the structure of the resulting constraint:

\[
\begin{align*}
\Lambda_N \cdot y' &= 1_N \\
\Lambda_M \cdot z &= 1_M \\
y, z &\in 0-1 \text{ integer}
\end{align*}
\]

However, in this case where the objective, \( y \cdot \Gamma \cdot z \), is nonlinear, we cannot use the technique for solving linear programming to tackle the relaxed problem. The resulting problem remains difficult to solve due to the nonlinear objective.

Therefore, we resort to the integrality constraint (no. 3). After we relax this integrality constraint, the problem becomes

\[
\begin{align*}
\min_{y,z} & \quad y \cdot \Gamma \cdot z \\
\text{s.t.} & \\
\Lambda_N \cdot y' &= 1_N \\
\Lambda_M \cdot z &= 1_M \\
\Phi_y \cdot y' + \Phi_y \cdot z &\leq 0 \\
y, z &\in [0,1]
\end{align*}
\]

Here, the last constraint says that each decision variable is now free to take any real value between zero and one. This means that we can take apart functions and modules to their partial forms and allocate them to the iterations, i.e. each iteration can have a proportion of either function or module as opposed to the 'whole' function or module.
However, it turns out that we can take advantage of the gradient-based local search algorithm to solve the problem with an additional condition. That condition specifies that we only allow the function and module to 'straddle' only between adjacent iterations. This is to take advantage of the gradient-based algorithm which handles the adjacent movements.

There are several differences of the gradient-based approach applied to the relaxed problem as opposed to that applied to the original problem

1. Instead of turning off (changing to zero) and turning on (changing to one) the decision parameter corresponding to function or module of a particular iteration, we add and subtract it with a real value between zero and one.

2. In each cycle, we have a number $\delta$, which represents the maximum change in the decision parameter for that cycle. The actual change in the decision parameter is the minimum between $\delta$ and the remaining 'amount' of function or module in the iteration from which that function or module is moved. The number $\delta$ decreases in a geometrical series with a ratio of $\alpha$.

3. Instead constructing tree to determine a combination of movements, we avoid the crossing by picking a single movement in a cycle, i.e. we pick the movement which gives the largest negative objective difference among all possible adjacent movements.

Figure 6.14 summarize the procedure for gradient-based algorithm described thus far.


**Pseudo-Program**

1. Begin with an initial guess for \( y_i \) and \( z_i \). Set \( i = 0 \)

2. Set \( \delta \) the maximum change of value of decision parameter in a given cycle

3. Calculate the gradient corresponding to \( y_i \) and \( z_i \)

\[
R_y(z_i) = \Gamma \cdot z_i
\]
\[
R_z(y_i) = y_i \cdot \Gamma
\]

4. Generate the set of adjacent movements using procedure described in section 6.3.

5. Calculate the best backward movements and the best forward movements for each pair of iterations in terms of the objective difference

\[
\delta(\Delta, S) = \sum_{n \in A} (\nabla y_{i+1} - \nabla y_i) + \sum_{n \in B} (\nabla z_{i+1} - \nabla z_i)
\]

6. Find the best movements, i.e. those with largest negative objective difference

7. Set \( i = i + 1 \)

8. Generate the next candidate solution \( y_i, z_i \) from 7 and \( \delta \). The actual change in the decision parameter is given by the minimum between \( \delta \) and the remaining ‘amount’ of function or module in the iteration from which that function or module is moved.

10. Repeat from step 3 using the new candidate solution

---

*Figure 6.14 Lower-bound approximation*
6.5 Numerical Results

Reference Problems

The numerical examples used to test the effectiveness of the propose algorithms should reflect the dimensions of problems in practice. We believe that the number of iterations involved in iteration plan of a development process is between 2-3 iterations. The number of functions and modules involved depend on the size of the problem and the level of details. We believe that the practical figure lies within the range of 10-100 for both functions and modules.

In addition to the size of real problem, we generate several problems to cover the different possibilities of functional mapping and number of interfaces. Here, we have the total of 12 problems to perform numerical study. From these problems, we cover the cases of 2-3 iterations, 10-20 functions/modules, modular functional mapping and general functional mapping, and the case of one interface and 4 interfaces.

Table 6.1 shows the details of the reference problems in our study. Problem 13-16 are large problems for which the optimal solutions are not available for comparison. The appendix of this chapter shows the details of the risk matrices of these problems.

Numerical Results

Table 6.2-6.4 shows the results of our numerical experiments on the proposed algorithms. These tables include the results from the enumeration procedures (from which the optimal solution is obtained), the gradient-based local search, the simulated annealing, and the lower bound approximation. The data include the objective values of the final candidate solutions, the percent deviation from the optimal objective values (the effectiveness), and the amount of time to run the algorithms (the efficiency). All experiments are performed under MATLAB on a PC with Intel™ Celeron 466 128MB.

From the experiments for problem 1-12, the gradient-based local search performs remarkably well from both the effectiveness and the efficiency perspectives. In 9 out of 12 of the examples (75%), the algorithm gives the optimal result. In the remaining case, the deviation from optimality is relatively small (1.6%, 2.7%, 5.3%). This is even more remarkable when viewed from the efficiency perspective as in all those cases, none of them take more than 0.5 sec of the running time.

Chapter 6: Numerical Approach
Table 6.1 Details of reference problems

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Number of Iterations</th>
<th>No. of Functions</th>
<th>Functional Mapping</th>
<th>No. of Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>15</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>15</td>
<td>M</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>15</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>15</td>
<td>G</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>20</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>20</td>
<td>M</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>20</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>20</td>
<td>G</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>10</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>10</td>
<td>M</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>10</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>10</td>
<td>G</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>50</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>50</td>
<td>M</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>50</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>50</td>
<td>G</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 6.2 Objective of final solution

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Enumeration (optimal)</th>
<th>Gradient-based</th>
<th>Simulated Annealing</th>
<th>Lower Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5901</td>
<td>0.5901</td>
<td>0.5972</td>
<td>0.4793</td>
</tr>
<tr>
<td>2</td>
<td>2.5456</td>
<td>2.5456</td>
<td>2.5991</td>
<td>2.2955</td>
</tr>
<tr>
<td>3</td>
<td>0.7826</td>
<td>0.8237</td>
<td>0.8237</td>
<td>0.7701</td>
</tr>
<tr>
<td>4</td>
<td>3.6009</td>
<td>3.8009</td>
<td>3.6461</td>
<td>3.1836</td>
</tr>
<tr>
<td>5</td>
<td>0.5833</td>
<td>0.5833</td>
<td>0.6021</td>
<td>0.5747</td>
</tr>
<tr>
<td>6</td>
<td>2.6116</td>
<td>2.8116</td>
<td>2.6633</td>
<td>2.4159</td>
</tr>
<tr>
<td>7</td>
<td>0.8984</td>
<td>0.9228</td>
<td>0.9096</td>
<td>0.8213</td>
</tr>
<tr>
<td>8</td>
<td>3.7721</td>
<td>3.7721</td>
<td>3.8110</td>
<td>3.3820</td>
</tr>
<tr>
<td>9</td>
<td>0.2946</td>
<td>0.2946</td>
<td>0.2946</td>
<td>0.2806</td>
</tr>
<tr>
<td>10</td>
<td>2.0495</td>
<td>2.0495</td>
<td>2.1200</td>
<td>1.8839</td>
</tr>
<tr>
<td>11</td>
<td>0.7331</td>
<td>0.7331</td>
<td>0.7411</td>
<td>0.6331</td>
</tr>
<tr>
<td>12</td>
<td>3.3126</td>
<td>3.3670</td>
<td>3.3326</td>
<td>2.9259</td>
</tr>
<tr>
<td>13</td>
<td>n.a.</td>
<td>0.3852</td>
<td>0.4618</td>
<td>0.5340*</td>
</tr>
<tr>
<td>14</td>
<td>n.a.</td>
<td>2.275</td>
<td>2.4471</td>
<td>2.4216*</td>
</tr>
<tr>
<td>15</td>
<td>n.a.</td>
<td>0.9606</td>
<td>0.9771</td>
<td>0.3557</td>
</tr>
<tr>
<td>16</td>
<td>n.a.</td>
<td>3.8739</td>
<td>3.9108</td>
<td>2.6514</td>
</tr>
</tbody>
</table>
Table 6.3 Percent deviation from optimal objective

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Gradient-based</th>
<th>Simulated Annealing</th>
<th>Lower Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0%</td>
<td>1.2%</td>
<td>18.8%</td>
</tr>
<tr>
<td>2</td>
<td>0.0%</td>
<td>2.1%</td>
<td>9.8%</td>
</tr>
<tr>
<td>3</td>
<td>5.3%</td>
<td>5.3%</td>
<td>1.6%</td>
</tr>
<tr>
<td>4</td>
<td>0.0%</td>
<td>1.3%</td>
<td>11.6%</td>
</tr>
<tr>
<td>5</td>
<td>0.0%</td>
<td>3.2%</td>
<td>1.5%</td>
</tr>
<tr>
<td>6</td>
<td>0.0%</td>
<td>2.0%</td>
<td>7.5%</td>
</tr>
<tr>
<td>7</td>
<td>2.7%</td>
<td>1.2%</td>
<td>8.6%</td>
</tr>
<tr>
<td>8</td>
<td>0.0%</td>
<td>1.0%</td>
<td>10.3%</td>
</tr>
<tr>
<td>9</td>
<td>0.0%</td>
<td>0.0%</td>
<td>4.8%</td>
</tr>
<tr>
<td>10</td>
<td>0.0%</td>
<td>3.4%</td>
<td>8.1%</td>
</tr>
<tr>
<td>11</td>
<td>0.0%</td>
<td>1.1%</td>
<td>13.6%</td>
</tr>
<tr>
<td>12</td>
<td>1.6%</td>
<td>0.6%</td>
<td>11.7%</td>
</tr>
</tbody>
</table>

Table 6.4 Running time (sec)

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Enumeration</th>
<th>Time (sec)</th>
<th>Gradient-based</th>
<th>Simulated Annealing</th>
<th>Lower Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>39.10</td>
<td>0.14</td>
<td>6.34</td>
<td>24.37</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>41.28</td>
<td>0.22</td>
<td>6.73</td>
<td>24.03</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>39.88</td>
<td>0.14</td>
<td>9.49</td>
<td>8.41</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>41.52</td>
<td>0.14</td>
<td>11.25</td>
<td>8.37</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1,581.80</td>
<td>0.40</td>
<td>9.53</td>
<td>34.47</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1,458.50</td>
<td>0.42</td>
<td>12.70</td>
<td>48.67</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1,349.00</td>
<td>0.17</td>
<td>16.85</td>
<td>17.68</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1,472.10</td>
<td>0.32</td>
<td>15.92</td>
<td>18.57</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>87.23</td>
<td>0.23</td>
<td>6.79</td>
<td>28.48</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>77.27</td>
<td>0.18</td>
<td>6.86</td>
<td>24.90</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>75.44</td>
<td>0.20</td>
<td>6.54</td>
<td>11.71</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>77.84</td>
<td>0.12</td>
<td>7.01</td>
<td>14.01</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>n.a.</td>
<td>12.84</td>
<td>208.45</td>
<td>571.65</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>n.a.</td>
<td>12.84</td>
<td>235.95</td>
<td>709.87</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>n.a.</td>
<td>3.70</td>
<td>85.00</td>
<td>1084.60</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>n.a.</td>
<td>2.75</td>
<td>135.64</td>
<td>876.40</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 6: Numerical Approach
The simulated-annealing performs quite well. In our problems, the deviation from the optimality stays in the range of 0-5%, with the majority (75%) lying somewhere in the range of 0-2%. The running time stays primarily between 6-15 seconds. (Note that in this implementation, the algorithm selects the best solution from 15 algorithm runs)

Between gradient-based local search and simulated-annealing, gradient-based local search beats simulated annealing most of the time (9 out of 12) in terms of percent from optimality. There are only 2 problems where simulated annealing performs better and one problem when both fare equally.

We find that the tightness of the lower bound approximation varies significantly from problem to problem. The percent deviation ranges from 1.6% in the best case to 18.8% in the worst case. The range of running time is around 10-30 seconds.

In problems 13-16, the size of the problems does not allow us to obtain the optimal results. From the results, we can see that the effectiveness of the gradient-based local search and the simulated annealing are quite close, although in all cases, gradient-based local search outperform the simulated annealing. The lower bound algorithm seems to have a problem in this case as we see that the final values in problem 13 and 14 are in fact higher than the final value of the gradient-based local search. This can stem from the fact that the algorithm gets stuck at the local optima before it reaches the global optimal. This can be a subject for future investigation of the algorithm.
6.6 Conclusions

This chapter takes us through several issues related to the numerical approach in solving the work-minimization problem. We first establish that the problem is NP-complete and heuristics can be helpful in solving the problem. Then, we propose three algorithms that can be used to tackle the problem, namely the gradient-based algorithm, the simulated annealing algorithm, and the lower bound approximation.

To test the validity of these algorithms, we perform numerical experimentations on a set of selected work-minimization problems. The problems cover the range of the size typically found in practice. From the numerical experimentation, we found that the gradient-based algorithm performs remarkably well, both from the point of view of effectiveness (percent deviation from optimality) and the efficiency (the run time). The performance of simulated annealing is quite good, although not anywhere near that of the gradient-based algorithm. The performance of lower bound approximation varies significantly from problem to problem. However, this is more than make up for by the effectiveness of the gradient-based algorithm.
6.7 Appendix: Model Parameters
Table 6A.1: System Matrix for Problem 1 (2 iterations, 15 functions, modular functional mapping, 1 interface)

<table>
<thead>
<tr>
<th></th>
<th>0.080</th>
<th>0.050</th>
<th>0.105</th>
<th>0.055</th>
<th>0.075</th>
<th>0.031</th>
<th>0.069</th>
<th>0.082</th>
<th>0.067</th>
<th>0.069</th>
<th>0.079</th>
<th>0.068</th>
<th>0.096</th>
<th>0.055</th>
<th>0.017</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.092</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.077</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.069</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.089</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.020</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.011</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.092</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.052</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.157</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.068</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.037</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.049</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.018</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.148</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.021</td>
</tr>
</tbody>
</table>

179
### Table 6A.2: System Matrix for Problem 2 (2 iterations, 15 functions, modular functional mapping, 4 interface)

<table>
<thead>
<tr>
<th>0.014</th>
<th>0.067</th>
<th>0.089</th>
<th>0.072</th>
<th>0.004</th>
<th>0.051</th>
<th>0.089</th>
<th>0.110</th>
<th>0.124</th>
<th>0.069</th>
<th>0.067</th>
<th>0.119</th>
<th>0.072</th>
<th>0.029</th>
<th>0.022</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.038</td>
<td>0.032</td>
<td>0.033</td>
<td>0.020</td>
<td>0.111</td>
<td>0.086</td>
<td>0.120</td>
<td>0.049</td>
<td>0.082</td>
<td>0.003</td>
<td>0.102</td>
<td>0.009</td>
<td>0.139</td>
<td>0.082</td>
<td>0.095</td>
</tr>
<tr>
<td>0.028</td>
<td>0.049</td>
<td>0.037</td>
<td>0.055</td>
<td>0.145</td>
<td>0.013</td>
<td>0.117</td>
<td>0.014</td>
<td>0.027</td>
<td>0.064</td>
<td>0.151</td>
<td>0.014</td>
<td>0.111</td>
<td>0.038</td>
<td>0.138</td>
</tr>
<tr>
<td>0.119</td>
<td>0.020</td>
<td>0.111</td>
<td>0.012</td>
<td>0.060</td>
<td>0.003</td>
<td>0.007</td>
<td>0.093</td>
<td>0.125</td>
<td>0.099</td>
<td>0.016</td>
<td>0.017</td>
<td>0.078</td>
<td>0.108</td>
<td>0.132</td>
</tr>
</tbody>
</table>

| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.043 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.045 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.054 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.093 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.041 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.064 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.068 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0.058 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0.039 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0.163 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0.051 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0.044 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0.111 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0.060 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.066 |
Table 6A.3: System Matrix for Problem 3 (2 iterations, 15 functions, general functional mapping, 1 interface)

<table>
<thead>
<tr>
<th></th>
<th>0.121</th>
<th>0.074</th>
<th>0.054</th>
<th>0.042</th>
<th>0.079</th>
<th>0.015</th>
<th>0.125</th>
<th>0.083</th>
<th>0.072</th>
<th>0.042</th>
<th>0.065</th>
<th>0.097</th>
<th>0.003</th>
<th>0.002</th>
<th>0.126</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.114</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.119</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.018</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.046</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.141</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.117</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.029</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.034</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.036</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.074</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.073</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.021</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.064</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.073</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.040</td>
</tr>
</tbody>
</table>

181
Table 6A.4: System Matrix for Problem 4 (2 iterations, 15 functions, general functional mapping, 4 interface)

<table>
<thead>
<tr>
<th></th>
<th>0.012</th>
<th>0.019</th>
<th>0.095</th>
<th>0.129</th>
<th>0.049</th>
<th>0.064</th>
<th>0.092</th>
<th>0.011</th>
<th>0.135</th>
<th>0.017</th>
<th>0.086</th>
<th>0.081</th>
<th>0.090</th>
<th>0.009</th>
<th>0.110</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.106</td>
<td>0.012</td>
<td>0.031</td>
<td>0.046</td>
<td>0.007</td>
<td>0.025</td>
<td>0.116</td>
<td>0.097</td>
<td>0.037</td>
<td>0.091</td>
<td>0.095</td>
<td>0.102</td>
<td>0.128</td>
<td>0.087</td>
<td>0.022</td>
</tr>
<tr>
<td>0</td>
<td>0.032</td>
<td>0.046</td>
<td>0.123</td>
<td>0.057</td>
<td>0.125</td>
<td>0.096</td>
<td>0.083</td>
<td>0.092</td>
<td>0.018</td>
<td>0.100</td>
<td>0.007</td>
<td>0.003</td>
<td>0.010</td>
<td>0.075</td>
<td>0.132</td>
</tr>
<tr>
<td>0</td>
<td>0.116</td>
<td>0.044</td>
<td>0.079</td>
<td>0.144</td>
<td>0.051</td>
<td>0.023</td>
<td>0.029</td>
<td>0.092</td>
<td>0.032</td>
<td>0.007</td>
<td>0.114</td>
<td>0.031</td>
<td>0.138</td>
<td>0.076</td>
<td>0.023</td>
</tr>
</tbody>
</table>

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | 0.062 | 0.016 | 0.068 | 0.087 |
| 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 0  | 0  | 0  | 0.107 | 0.012 | 0.063 | 0.013 |
| 0  | 0  | 0  | 0  | 0  | 1  | 1  | 0  | 1  | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0.042 | 0.087 | 0.024 | 0.057 |
| 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0.015 | 0.065 | 0.111 | 0.035 |
| 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0  | 1  | 0  | 1  | 1  | 1  | 1  | 0  | 1  | 0.072 | 0.092 | 0.068 | 0.058 |
| 1  | 1  | 1  | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0.126 | 0.008 | 0.127 | 0.105 |
| 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0.110 | 0.121 | 0.047 | 0.006 |
| 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 0  | 1  | 1  | 1  | 0  | 0  | 1  | 0.107 | 0.053 | 0.064 | 0.004 |
| 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0.040 | 0.106 | 0.107 | 0.108 |
| 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0.024 | 0.066 | 0.099 | 0.067 |
| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 0  | 1  | 1  | 1  | 1  | 0.071 | 0.101 | 0.033 | 0.109 |
| 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 1  | 1  | 0  | 0  | 1  | 1  | 1  | 0.029 | 0.055 | 0.050 | 0.108 |
| 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 1  | 1  | 1  | 0  | 1  | 0  | 0  | 0.021 | 0.056 | 0.076 | 0.096 |
| 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 1  | 1  | 0  | 0  | 0  | 1  | 0  | 0.097 | 0.129 | 0.054 | 0.042 |
| 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 1  | 0.076 | 0.033 | 0.010 | 0.103 |
Table 6A.5: System Matrix for Problem 5 (2 iterations, 20 functions, modular functional mapping, 1 interface)

|          | 0.001 | 0.019 | 0.102 | 0.081 | 0.050 | 0.028 | 0.111 | 0.036 | 0.016 | 0.074 | 0.018 | 0.027 | 0.038 | 0.108 | 0.018 | 0.049 | 0.047 | 0.039 | 0.033 |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 1        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.007 |
| 0        | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.020 |
| 0        | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.025 |
| 0        | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.045 |
| 0        | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.097 |
| 0        | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.027 |
| 0        | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.055 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.087 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.095 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.047 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.089 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.000 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0.048 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0.029 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0.056 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0.043 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0.083 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0.096 |
| 0        | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0.004 |
|          | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0.045 |
### Table 6A.6: System Matrix for Problem 6 (2 iterations, 20 functions, modular functional mapping, 4 interfaces)

<table>
<thead>
<tr>
<th></th>
<th>0.041</th>
<th>0.032</th>
<th>0.024</th>
<th>0.091</th>
<th>0.100</th>
<th>0.021</th>
<th>0.042</th>
<th>0.089</th>
<th>0.101</th>
<th>0.088</th>
<th>0.050</th>
<th>0.020</th>
<th>0.005</th>
<th>0.102</th>
<th>0.001</th>
<th>0.043</th>
<th>0.063</th>
<th>0.048</th>
<th>0.034</th>
<th>0.004</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.020</td>
<td>0.075</td>
<td>0.061</td>
<td>0.069</td>
<td>0.068</td>
<td>0.022</td>
<td>0.062</td>
<td>0.025</td>
<td>0.069</td>
<td>0.065</td>
<td>0.063</td>
<td>0.063</td>
<td>0.016</td>
<td>0.068</td>
<td>0.048</td>
<td>0.062</td>
<td>0.018</td>
<td>0.047</td>
<td>0.031</td>
<td>0.049</td>
<td></td>
</tr>
<tr>
<td>0.019</td>
<td>0.061</td>
<td>0.102</td>
<td>0.058</td>
<td>0.054</td>
<td>0.050</td>
<td>0.022</td>
<td>0.009</td>
<td>0.079</td>
<td>0.036</td>
<td>0.047</td>
<td>0.016</td>
<td>0.026</td>
<td>0.043</td>
<td>0.066</td>
<td>0.041</td>
<td>0.088</td>
<td>0.090</td>
<td>0.072</td>
<td>0.021</td>
<td></td>
</tr>
<tr>
<td>0.090</td>
<td>0.063</td>
<td>0.081</td>
<td>0.083</td>
<td>0.043</td>
<td>0.091</td>
<td>0.012</td>
<td>0.002</td>
<td>0.072</td>
<td>0.055</td>
<td>0.018</td>
<td>0.000</td>
<td>0.050</td>
<td>0.040</td>
<td>0.001</td>
<td>0.065</td>
<td>0.069</td>
<td>0.038</td>
<td>0.092</td>
<td>0.044</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

0.007 | 0.051 | 0.077 | 0.066 | 0.020 | 0.059 | 0.073 | 0.046 | 0.025 | 0.002 | 0.025 | 0.064 | 0.045 | 0.043 | 0.028 | 0.026 | 0.097 | 0.090 | 0.007 | 0.086 | 0.027 | 0.061 | 0.092 | 0.038 | 0.055 | 0.086 | 0.007 | 0.032 | 0.087 | 0.040 | 0.100 | 0.091 | 0.095 | 0.063 | 0.055 | 0.007 | 0.047 | 0.040 | 0.005 | 0.020 | 0.089 | 0.087 | 0.018 | 0.014 | 0.000 | 0.046 | 0.060 | 0.058 | 0.048 | 0.003 | 0.084 | 0.082 | 0.029 | 0.104 | 0.025 | 0.067 | 0.056 | 0.075 | 0.017 | 0.026 | 0.043 | 0.023 | 0.096 | 0.057 | 0.083 | 0.079 | 0.057 | 0.025 | 0.096 | 0.005 | 0.039 | 0.042 | 0.004 | 0.030 | 0.037 | 0.081 | 0.045 | 0.014 | 0.097 | 0.074 |
Table 6A.7: System Matrix for Problem 7 (2 iterations, 20 functions, general functional mapping, 1 interface)

| 0.036 | 0.118 | 0.074 | 0.051 | 0.051 | 0.092 | 0.001 | 0.038 | 0.043 | 0.097 | 0.090 | 0.019 | 0.071 | 0.032 | 0.006 | 0.071 | 0.007 | 0.013 | 0.010 | 0.082 | 0.043 | 0.005 | 0.062 | 0.047 | 0.086 | 0.017 | 0.010 | 0.047 | 0.062 | 0.024 | 0.076 | 0.065 | 0.051 | 0.026 | 0.096 | 0.048 | 0.075 | 0.082 | 0.039 | 0.038 |
|-------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 1     | 0    | 0    | 1    | 0    | 1    | 1    | 0    | 0    | 0    | 0    | 1    | 0    | 1    | 0    | 1    | 1    | 0    | 0    | 1    | 0    | 0.043 |
| 1     | 0    | 1    | 0    | 1    | 0    | 0    | 0    | 0    | 1    | 0    | 1    | 0    | 1    | 1    | 1    | 0    | 1    | 0    | 1    | 0    | 0.005 |
| 1     | 1    | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 1    | 1    | 0    | 0    | 0    | 1    | 0    | 0    | 1    | 0    | 0    | 0    | 0.062 |
| 1     | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 1    | 1    | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0.047 |
| 1     | 1    | 0    | 1    | 1    | 1    | 1    | 0    | 0    | 0    | 0    | 1    | 1    | 1    | 1    | 0    | 1    | 0    | 0    | 0    | 0    | 0.086 |
| 1     | 1    | 0    | 1    | 0    | 1    | 1    | 0    | 1    | 1    | 1    | 0    | 1    | 0    | 1    | 0    | 1    | 0    | 0    | 0    | 0    | 0.017 |
| 1     | 1    | 0    | 1    | 1    | 1    | 1    | 0    | 0    | 0    | 0    | 1    | 1    | 0    | 1    | 0    | 1    | 1    | 1    | 0    | 0    | 0.010 |
| 1     | 1    | 0    | 1    | 1    | 1    | 1    | 0    | 0    | 1    | 0    | 1    | 0    | 1    | 0    | 1    | 1    | 1    | 0    | 0    | 0    | 0.047 |
| 0     | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 1    | 1    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 1    | 0    | 0.062 |
| 1     | 0    | 1    | 1    | 0    | 1    | 0    | 1    | 0    | 0    | 0    | 0    | 1    | 1    | 1    | 1    | 0    | 1    | 1    | 1    | 0    | 0.024 |
| 1     | 1    | 1    | 1    | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 1    | 0    | 1    | 0    | 1    | 1    | 0    | 1    | 0.076 |
| 0     | 1    | 1    | 1    | 1    | 1    | 0    | 1    | 0    | 0    | 0    | 0    | 1    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0.065 |
| 1     | 1    | 0    | 0    | 0    | 0    | 0    | 1    | 1    | 1    | 1    | 0    | 1    | 0    | 0    | 1    | 1    | 0    | 1    | 0    | 0    | 0.051 |
| 0     | 0    | 0    | 0    | 0    | 1    | 0    | 1    | 1    | 0    | 0    | 1    | 1    | 0    | 1    | 1    | 0    | 1    | 1    | 0    | 1    | 0.026 |
| 0     | 1    | 1    | 1    | 1    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 1    | 0    | 0    | 0.096 |
| 0     | 1    | 0    | 1    | 1    | 0    | 1    | 0    | 0    | 0    | 0    | 1    | 0    | 1    | 0    | 1    | 0    | 1    | 0    | 0    | 1    | 0.048 |
| 1     | 0    | 1    | 1    | 0    | 0    | 1    | 1    | 0    | 1    | 0    | 1    | 0    | 1    | 1    | 0    | 1    | 0    | 1    | 0    | 0    | 0.075 |
| 0     | 1    | 0    | 1    | 1    | 0    | 1    | 1    | 1    | 0    | 0    | 1    | 0    | 1    | 1    | 1    | 0    | 1    | 1    | 0    | 1    | 0.082 |
| 0     | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 0    | 1    | 0    | 1    | 1    | 1    | 0    | 1    | 1    | 0    | 0.039 |

185
<table>
<thead>
<tr>
<th>0.087</th>
<th>0.059</th>
<th>0.005</th>
<th>0.032</th>
<th>0.087</th>
<th>0.036</th>
<th>0.030</th>
<th>0.011</th>
<th>0.039</th>
<th>0.046</th>
<th>0.083</th>
<th>0.012</th>
<th>0.073</th>
<th>0.091</th>
<th>0.036</th>
<th>0.092</th>
<th>0.016</th>
<th>0.038</th>
<th>0.020</th>
<th>0.106</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.022</td>
<td>0.008</td>
<td>0.009</td>
<td>0.080</td>
<td>0.092</td>
<td>0.050</td>
<td>0.057</td>
<td>0.113</td>
<td>0.042</td>
<td>0.086</td>
<td>0.057</td>
<td>0.019</td>
<td>0.000</td>
<td>0.083</td>
<td>0.015</td>
<td>0.112</td>
<td>0.057</td>
<td>0.034</td>
<td>0.037</td>
<td>0.026</td>
</tr>
<tr>
<td>0.020</td>
<td>0.071</td>
<td>0.053</td>
<td>0.048</td>
<td>0.017</td>
<td>0.063</td>
<td>0.063</td>
<td>0.060</td>
<td>0.085</td>
<td>0.033</td>
<td>0.050</td>
<td>0.073</td>
<td>0.014</td>
<td>0.018</td>
<td>0.017</td>
<td>0.023</td>
<td>0.045</td>
<td>0.076</td>
<td>0.081</td>
<td>0.071</td>
</tr>
<tr>
<td>0.039</td>
<td>0.057</td>
<td>0.008</td>
<td>0.052</td>
<td>0.064</td>
<td>0.077</td>
<td>0.060</td>
<td>0.090</td>
<td>0.035</td>
<td>0.080</td>
<td>0.006</td>
<td>0.057</td>
<td>0.038</td>
<td>0.072</td>
<td>0.017</td>
<td>0.032</td>
<td>0.052</td>
<td>0.043</td>
<td>0.052</td>
<td>0.070</td>
</tr>
</tbody>
</table>

Table 6A.8: System Matrix for Problem 8 (2 iterations, 20 functions, general functional mapping, 4 interfaces)

| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

0.059 | 0.065 | 0.046 | 0.067 | 0.023 | 0.027 | 0.045 | 0.043 | 0.047 | 0.083 | 0.054 | 0.003 | 0.063 | 0.068 | 0.038 | 0.041 | 0.017 | 0.077 | 0.055 | 0.004 | 0.010 | 0.059 | 0.023 | 0.035 | 0.078 | 0.007 | 0.019 | 0.044 | 0.034 | 0.046 | 0.009 | 0.066 | 0.006 | 0.087 | 0.061 | 0.084 | 0.065 | 0.047 | 0.032 | 0.033 | 0.052 | 0.022 | 0.094 | 0.077 | 0.071 | 0.012 | 0.052 | 0.076 | 0.010 | 0.008 | 0.013 | 0.073 | 0.053 | 0.089 | 0.040 | 0.059 | 0.080 | 0.022 | 0.065 | 0.046 | 0.051 | 0.069 | 0.081 | 0.052 | 0.036 | 0.037 | 0.089 | 0.082 | 0.043 | 0.039 | 0.023 | 0.031 | 0.075 | 0.042 | 0.069 | 0.034 | 0.088 | 0.094 | 0.094 | 0.047
Table 6A.9: System Matrix for Problem 9 (3 iterations, 10 functions, modular functional mapping, 1 interface)

<table>
<thead>
<tr>
<th></th>
<th>0.096</th>
<th>0.045</th>
<th>0.200</th>
<th>0.172</th>
<th>0.212</th>
<th>0.041</th>
<th>0.150</th>
<th>0.011</th>
<th>0.068</th>
<th>0.005</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.092</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.077</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.069</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.089</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.020</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.011</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.092</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.052</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.157</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.068</td>
</tr>
<tr>
<td></td>
<td>0.038</td>
<td>0.007</td>
<td>0.095</td>
<td>0.037</td>
<td>0.103</td>
<td>0.045</td>
<td>0.204</td>
<td>0.178</td>
<td>0.156</td>
<td>0.138</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>0.118</td>
<td>0.099</td>
<td>0.035</td>
<td>0.072</td>
<td>0.130</td>
<td>0.029</td>
<td>0.002</td>
<td>0.062</td>
<td>0.198</td>
<td>0.254</td>
<td></td>
</tr>
<tr>
<td>0.089</td>
<td>0.113</td>
<td>0.144</td>
<td>0.111</td>
<td>0.086</td>
<td>0.136</td>
<td>0.097</td>
<td>0.071</td>
<td>0.118</td>
<td>0.035</td>
<td></td>
</tr>
<tr>
<td>0.048</td>
<td>0.021</td>
<td>0.065</td>
<td>0.270</td>
<td>0.058</td>
<td>0.015</td>
<td>0.109</td>
<td>0.176</td>
<td>0.192</td>
<td>0.047</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0.075</th>
<th>0.128</th>
<th>0.109</th>
<th>0.003</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.161</td>
<td>0.145</td>
<td>0.035</td>
<td>0.145</td>
<td></td>
</tr>
<tr>
<td>0.167</td>
<td>0.149</td>
<td>0.035</td>
<td>0.046</td>
<td></td>
</tr>
<tr>
<td>0.091</td>
<td>0.043</td>
<td>0.000</td>
<td>0.053</td>
<td></td>
</tr>
<tr>
<td>0.053</td>
<td>0.014</td>
<td>0.142</td>
<td>0.196</td>
<td></td>
</tr>
<tr>
<td>0.069</td>
<td>0.201</td>
<td>0.254</td>
<td>0.174</td>
<td></td>
</tr>
<tr>
<td>0.039</td>
<td>0.197</td>
<td>0.140</td>
<td>0.164</td>
<td></td>
</tr>
<tr>
<td>0.135</td>
<td>0.021</td>
<td>0.244</td>
<td>0.006</td>
<td></td>
</tr>
<tr>
<td>0.105</td>
<td>0.081</td>
<td>0.025</td>
<td>0.080</td>
<td></td>
</tr>
<tr>
<td>0.104</td>
<td>0.022</td>
<td>0.016</td>
<td>0.133</td>
<td></td>
</tr>
</tbody>
</table>
Table 6A.11: System Matrix for Problem 11 (3 iterations, 10 functions, general functional mapping, 1 interface)

<table>
<thead>
<tr>
<th></th>
<th>0.016</th>
<th>0.029</th>
<th>0.132</th>
<th>0.181</th>
<th>0.013</th>
<th>0.163</th>
<th>0.028</th>
<th>0.154</th>
<th>0.091</th>
<th>0.194</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.097</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.107</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.091</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.058</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.113</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.036</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.172</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.109</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.021</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.195</td>
</tr>
</tbody>
</table>
Table 6A.12: System Matrix for Problem 12 (3 iterations, 10 functions, general functional mapping, 4 interfaces)

<table>
<thead>
<tr>
<th></th>
<th>0.042</th>
<th>0.106</th>
<th>0.062</th>
<th>0.101</th>
<th>0.112</th>
<th>0.044</th>
<th>0.171</th>
<th>0.080</th>
<th>0.079</th>
<th>0.204</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.110</td>
<td>0.085</td>
<td>0.006</td>
<td>0.075</td>
<td>0.209</td>
<td>0.155</td>
<td>0.000</td>
<td>0.007</td>
<td>0.131</td>
<td></td>
<td>0.222</td>
</tr>
<tr>
<td>0.024</td>
<td>0.163</td>
<td>0.103</td>
<td>0.020</td>
<td>0.133</td>
<td>0.177</td>
<td>0.058</td>
<td>0.139</td>
<td>0.080</td>
<td>0.104</td>
<td></td>
</tr>
<tr>
<td>0.194</td>
<td>0.027</td>
<td>0.001</td>
<td>0.205</td>
<td>0.121</td>
<td>0.163</td>
<td>0.090</td>
<td>0.130</td>
<td>0.063</td>
<td></td>
<td>0.007</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0.045  0.062  0.157  0.003
0.050  0.098  0.191  0.157
0.060  0.083  0.074  0.160
0.123  0.175  0.057  0.149
0.100  0.040  0.000  0.223
0.169  0.010  0.027  0.011
0.177  0.189  0.173  0.049
0.097  0.184  0.084  0.121
0.172  0.001  0.181  0.028
0.007  0.160  0.056  0.099
CHAPTER 7: CONCLUSIONS AND FUTURE RESEARCH

7.1 Introduction

This chapter concludes this thesis and provides some reflections on the topics that have been discussed thus far. In addition, the model extensions and the avenues for future research will also be discussed. The organization of this chapter is as follows. Section 7.2 summarizes the contributions that have been made in this work: the conceptual development, the risk perspective of a development process, the design and analysis of process models, and the decision support tools. The generalization and the extension of the model are discussed in section 7.3. Section 7.4 provides several possible research avenues from this work. Section 7.5 provides final words for this thesis.

7.2 Summary of Contributions

This thesis provides a new approach in understanding the development process using a risk dynamics model. This approaches call for an explicit modeling of risks inherent in the system and their effects on the development process. The resulting model provides a risk perspective for understanding the management of a development process. The thesis made the following contributions to research in product development.

Conceptual and Model Development

Recognizing the limitations of the extant literature that result from the information model, this thesis attacks the issue in a more direct manner. Instead of relying on the traditional approach where information related to the system is implicitly captured in the model, this work makes these assumptions explicit, using the concepts of functions, interfaces, and modules.

The thesis is founded on an important premise, which states that the relevant information for the system development comes from its functions. The premise makes it possible to model information in the development process in a more direct manner. This allows us to obtain several important insights concerning the relationships between the system and the process.
From the premise and other model assumptions, the thesis then formulates the risk dynamics model, which is a theory of how work is generated in a development process. The model relates the process policy to the amount of work in the process. The risk matrix is created to represent system parameters under the parameterized model. The work-minimization problem is derived from the risk dynamics model by specifying the objective function to be the sum of the risk-mitigation work and the expected rework. The resulting optimization problem is the basis for deriving important insights in the thesis.

*Risk perspective in complex system development*

From the work-minimization problem, this thesis derives several insights which help us understand the effects of risk on a development process.

1. Risk Efficiency

From the analysis of optimal policy of the work-minimization problem, the good process model is found to have the characteristic of risk efficiency. This refers to the situation when the uncertainty in the process is reduced quickly while the impact of risk is added to the system slowly. Risk efficiency depends on the two important process variables, the task sequence and the information exchange. The thesis shows how risk efficiency manifests itself in different ways for different types of systems.

2. Process Policy

For a single-interface system with a modular functional mapping (section 4.3), risk efficiency calls for a sequencing of module in such a way that the probability-impact ratio is decreasing in iterations. In other words, we sequence modules with high ratio early in the process and put those with low ratio later in the process. Such ratio represents the risk efficiency at the level of individual module. By sequencing module with high efficiency early in the process, we obtain a process with high overall risk efficiency, i.e. optimal process with respect to expected work.

For a single-interface system with a nested functional mapping (section 4.4), functions are organized into a hierarchy. In such case, the optimal policy can be obtained by sequencing the functions from the lowest level to the highest level in successive iterations. This sequence ensures that functions are built up as modules are added to the system. This is also the process structure which has the characteristic of risk efficiency.
The above two cases represent the two ends of the spectrum. For both cases, the optimal policy results in the fastest possible reduction in probability with the lowest increase in impact. For a single-interface system with a general mapping, we can use the same principle to come up with the optimal policy. The results for a single-interface system with general mapping are shown for the special case of two iterations in section 4.2.

3. Interactions of Multiple Risks

The results for the case of multiple-interface system are not as clean as those for a single-interface system. In this case, the problem is complicated by the fact the risk efficiency needs to be balanced among the various interfaces of the system. Section 4.5 provides two special cases in which a clean structure can be obtained. In one case, the problem can be translated into a single-interface problem. Here, the balance of risk can be done simply by replacing the parameters of the multiple interfaces with the weighted average parameters. In the second case, the problem results in an exhaustive policy where we group modules by their interfaces and the optimal policy calls for a complete allocation of one type of modules before we move to the other type.

4. Risk concentrations and product architecture

Section 4.3 provides insights concerning the concentration of risks. The comparison is made between a system with distributed risks (all functions have identical probability parameters) and that with concentrated risks (functions have varying probability parameters). The section shows that the system with concentrated risks yields a better process performance. This is due to the fact that, in such system, we can sequence the functions with higher concentration of risk (higher probability parameters) early in the process, thus yielding a better risk efficiency. In other words, the risk concentration gives us leverage in the risk management that does not exist in a system with distributed risks.

Section 4.4 provides insights on the effects of the product architecture on the development process. The section compares a system with a modular mapping to that with a nested mapping. We find that, given the same system parameters, the system with a modular mapping yields a better process performance (lower expected work). This is because the modular system gives more flexibility to the planner in sequencing system functions and modules in the way that minimize the amount of work.
Design and Analysis of Process Models

Since the work-minimization problem needs a volume of data which might not be present in practice, the thesis then goes on to create a tool which can help the development organization design and analyze their process models. The modified risk matrix only needs data that are related to functions and interfaces of the system, the two elements that are more readily available in the development at early stage. The risk indices are calculated from the parameters of the modified risk matrix and can be used to rank the importance of functions from the point of view of risk management.

We can also derive some important insights from this tool. Although process models seem to arise in reaction to the problem caused by the arbitrary decomposition of the system, such methods still rely on that very same system decomposition to prescribe the way for improvement. From this observation, this thesis contends that the only way to solve the problem is to discard such arbitrary decomposition entirely, relying instead on the risk metrics which show us the relative importance of several issues in the development.

Decision Support Tools

Besides the above contributions, the thesis also made two important contributions which assist the development of the decision support tools, the case study and the numerical approach. The case study focuses on the migration of an information system to a new platform at Verizon's SuperPages.com back-office group. The uncertainty resulting from unfamiliarity with technology represents the major challenge and is one of the reasons that the iterative process is used to plan the project.

The early phase of the migration provides some evidences to validate the use of modified risk matrix. In the pilot project, the system architect uses heuristics to pick out SuperCommerce subsystem among numerous other systems to conduct a pilot study. The heuristics we obtained in the interview seem to be very similar to the modified risk matrix and the risk indices proposed in section 4.6.

In the next phase of the project, I derive the numerical parameters of the subsystem and use it to calculate the optimal iteration policy. The resulting model provides guidance to the developer in planning the iteration. This example serves as a proof of concept for the model in the practical settings by demonstrating that the parameters can in fact be obtained in the real world and that the resulting model is useful in the project planning.
In numerical approach, I first provide answers to the question regarding the computational complexity of the work-minimization problem. In particular, I show that the problem belongs to the NP-complete class and thus poses some computational challenges.

In dealing with such a problem, I propose several algorithms that can be used to solve the problem, namely the gradient-based approach, the simulated annealing, and the lower bound approximation. From the numerical study, gradient-based algorithm performs remarkably well, and thus completely overshadows the importance of the other two algorithms.

### 7.3 Model Generalization and Extension

The work-generation model can be generalized and extended easily along many dimensions. In chapter 3, I have indicated some of the examples before I narrow the focus in the thesis to the work-minimization problem. In the following paragraphs, I will outline different possibilities for generalizing the model.

#### Probability and impact functions

I focus the thesis on systems with additive probability functions and additive impact functions to show the structural results of the model. In a more general setting, we can construct an arbitrary probability and impact functions to reflect the way different functions determine the probability of change of the interface and the way different interfaces determine the rework of a particular module.

#### Incremental function sets

In this thesis, I assume that only the interface change is a function of only incremental function set corresponding to the iteration in question, i.e.

\[
\Pr[I' = I_0] = f'(\Delta; I_0).
\]

The underlying assumption for this, as mentioned in chapter 3, is that the risk-mitigation activity does a "perfect" job in deriving the information from functions that become available in each iteration. In other words, I assume no possibility that some of mistakes made in earlier iterations were not detected and come back to create change in later iterations.
Such the case could in fact happen in the real setting. Therefore, we can extend the model to such case by assuming that all the incremental function sets up to that iteration together determine the interface change, i.e.

$$\Pr[I^j = I^0] = f^j(\Delta^i, \Delta^{i-1}, \ldots, \Delta^1; I^0)$$

*Risk-mitigation work*

The risk-mitigation work has been played down in this thesis. I assume that the risk-mitigation work is a fixed constant for a given iteration. In reality, this could be a very complicated function of either module allocations

$$\left(S^j, S^{j-1}, \ldots, S^1\right)$$

and/or incremental function sets

$$\left(\Delta^i, \Delta^{i-1}, \ldots, \Delta^1\right)$$

In fact, the risk-mitigation work can also vary depending on the alternative risk-mitigation activities we can use. For example, prototypes can be constructed using different methods with different lead-time. Such parameters can be entered easily in our model.

*Risk aversion*

Instead of looking at the expected work, we can construct more complicated objective from the probability of change and the impact of change. For example, we can construct an objective that reflects the risk aversion of the project planner in contrast to the underlying assumption of risk neutrality in the case when the objective is the expected work. I provided one example for this in our case study in chapter 5 by constructing the objective as a linear combination of the expected value of work and the variance of work.

*Other costs*

We can also look at other types of cost besides those related to the amount of time spent by the developers as well. For example, the costs related to the risk-mitigation activity might be the financial costs rather than the amount of time to construct the prototype. The impact of a change in interface might entail not only the amount of time to do rework, but other financial impacts as
well. We can model the effects of these different types of costs by substituting them into the work-related parameters of the model (impact parameters, risk-mitigation work, etc.)

Cycle time

Finally, we can also extend the model to analyze the case when the objective is the cycle time rather than the sum of the work. To do so, first notice that cycle time comes into the picture in the way we allocate tasks to development resources, i.e. developers or teams.

Here, we can use the work-generation model as a starting point to come up with the amount of work on development tasks that correspond to some particular modules. Then, we use the allocation scheme that specifies how to allocate tasks to different development resources as the constraint in the optimization problem. The lead-time can then be calculated for each development resource. The cycle time for each iteration can then be calculated from such lead-time (e.g. as the maximum of lead-time across different resources). We can then sum the cycle time across iterations to derive the overall process cycle-time.
7.4 Future Research Avenues

With the results in this thesis as a starting point, I find a number of possible research avenues that represent the natural next steps from this work.

*Model analysis*

This thesis shows several analytical results from our relatively simplified model. These results in fact give many useful insights into the management of development process, especially from the point of view risk management. I believe that the continuation of work in this approach will yield even more interesting results.

This work only provides some partial results that could be generalized further. Examples include the structure of the optimal policy for a single-interface system with a general functional mapping, the structure of the optimal policy for a multiple-interface system, the comparisons of systems with different architectural profiles, the study of the effects of the risk mitigation activities on the optimal process, and so on.

In addition to this, I also believe that an analysis of the model generalizations and extensions shown in the previous section also provides an interesting avenue of future research. It would be interesting to understand how different probability functions and impact functions affect the structure of the optimal policy, to see the effects of the imperfect risk-mitigation activity on the process structure, to explore how more complicated versions of risk-mitigation work change the picture and to determine whether risk aversion has any effects.

*Model application*

We need to do more work to understand the issues related to the application of this model to a real process. I show a small example in this thesis as a proof of concept for the model. In a more complicated case, I believe that the model applications might be more challenging.

The process for obtaining the model parameters could present a significant challenge, especially for a person who is not the expert in the system domain. How could we solicit information necessary to construct the model in this case could be an interesting research question to ask.
Numerical approach

I also believe that further work on numerical approach can be conducted. Here, I show the efficacy of the gradient-based algorithm for a number of numerical examples. Although the results seem to show that such algorithm is very effective, it remains to be seen how effective it will be for larger problems.

In addition, to cope with the generalized and extended models, we will need a more powerful computational tool. There is a need for heuristics or other computational techniques that allow us to find the optimal solution in a reasonable amount of time. This will be an interesting research avenue to follow as well.

7.5 Final words

Two sentences very well capture the essence of this thesis. System and process interact in the form of risk. The key to managing the process is therefore to look at the risks in the system and not at the system itself.

I hope that this thesis provides the reader with a motivation for this new method and a confidence in its applicability and computational feasibility. I hope that the readers who are interested in using this new method in their research can benefit from the work contained herein.
REFERENCE


*Reference*


