MODELING AND ANALYZING CONCURRENT PROCESSES FOR PROJECT PERFORMANCE IMPROVEMENT

LIN JUN

NATIONAL UNIVERSITY OF SINGAPORE

2008

MODELING AND ANALYZING CONCURRENT PROCESSES FOR PROJECT PERFORMANCE IMPROVEMENT

LIN JUN

(*M.Mgt., Xian Jiaotong University, China*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF INDUSTRIAL & SYSTEMS ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2008

ACKNOWLEDGEMENTS

This thesis would have never been completed successfully without the help from those who have supported me throughout the course of my doctoral studies, including family, friends, and colleagues. I would like to take this opportunity to express my appreciation to all of them.

First of all I would like to thank my supervisors. At NUS I would like to thank Dr. Chai and Prof. Wong. It was Dr. Chai who led me into this research field and guided me throughout the whole period. His enthusiasm, patience, encouragement and support have kept me working on the right track with a high spirit. I would like to thank Prof. Wong for his support and encouragement in many ways to finish this thesis. His comments and recommendations of my reports are usually timely and thoughtful. At TU/e I would like to thank Prof. Brombacher. Although he had a tight agenda, he always managed to make time for me every week when I was in TU/e from 2006 to 2007. As a result, we had many efficient and fruitful discussions some of which have been incorporated in this thesis. His critical comments have also helped me to improve this work. Working with my three supervisors is an exceptional experience for me, and I believe such experience will definitely benefit me for the whole life.

I would like to thank the faculty members of Department of Industrial and Systems Engineering, from whom I have learnt not only knowledge but also skills in research as well as teaching. I am also very grateful to my colleagues in ISE Department of NUS and QRE department of TU/e for their kindly help. They include Foong Hing Wih, Zhou Peng, Wang Qi, Li Suyi, Sari Kartika Josephine and others. I benefit a lot through discussion with them about my research methodology, research gaps, and so on.

Special appreciation goes to the staffs in Shanghai Sunplus Communication Technology Co., Ltd., China Techfaith Wireless Communication Technology Ltd., and Haier Electronics Group Co., Ltd. for their support and collaboration in this project, which enriches this research from practical point of view.

Without the support from my family the thesis would have been impossible. Especially, I want to thank my wife, Qian Yanjun, for her patience and support, which helped me overcome all the difficulties faced throughout the course of doctorial studies.

Lin Jun

May 2007

TABLE OF CONTENTS

SUMMARY

Market and technology changes have brought about new characteristics of product development. Developing products faster, better, and cheaper than competitors has become critical to success. In response to these pressures, many industries have shifted from a sequential and functional development paradigm to a concurrent and cross-functional paradigm. Increasing the concurrency, however, also increases the complexity of development projects. Our literature review shows that there is a lack of methods to help management to derive appropriate development policies (such as overlapping degree, communication frequency, and functional interaction level). According to the information dependency and communication cost, we grouped concurrent product development processes into three types and proposed three models to manage them. These models are validated or illustrated with product development case studies in three consumer electronics companies.

The first model presented is an analytical model for managing concurrent development processes with sequential dependence and low communication cost. It is well known that continuous information exchange is optimal when communication cost is low. Therefore the concurrent problem can be simplified into an overlapping problem regardless of communication strategies. Appropriate overlapping degree and functional interaction level for projects with different properties are proposed. This model was applied to examine the development policies in a handset design company.

The second model proposed deals with concurrent development processes with sequential dependence and high communication cost. In this case, the communication policy is extremely important. If information exchange is too frequent, the communication time and cost would increase significantly. However, infrequent information exchange would increase downstream rework. The model aims to optimize project performance by investigating the interactions between overlapping policy and communication strategy. The model was applied to improve the refrigerator development process in a consumer electronics company.

Finally a simulation model for managing overlapped iterative product development (i.e. the overlapped stages are interdependent) is developed. For iterative processes, the interaction is much more complex and analytical approaches have proved to be prohibitively expensive. Consequently, a System Dynamics model is built for modeling overlapped iterative development processes. Using this model we can track the impact of different overlapping degrees and testing qualities on project performance. Therefore, it can help management find appropriate development policies. The model was implemented in a design house and led to marked improvement in project performance, thus demonstrating the viability of the model.

This study is motivated by the needs of companies, and is developed based on previous literature and in-depth case studies. The usefulness and validity of the insights, analytical results, and algorithms proposed in this research have been validated through the case studies done in consumer electronics companies. We believe that the results proposed can also be applied to manage concurrent processes in other industries with similar properties.

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

CE	Concurrent Engineering
CPM	Critical Path Method
DDPM	Dynamic Development Process Model
DES	Discrete Event Simulation
DSM	Design Structure Matrix
MAE	Mean Absolute Error
NPD	New Product Development
PERT	Program Evaluation and Review Technique
PGM	Performance Generation Model
PD	Product Development
RMSE	Root Mean Square Error
SD	System Dynamics

CHAPTER 1

INTRODUCTION

The outline of this chapter is given as follows. In Section 1.1, the research background is explained. In Section 1.2 the research gap is proposed, followed in Section 1.3 by the research objective. The research approaches applied in this research project are discussed in Section 1.4. The structure of this thesis is given in the end.

1.1 Background

In the traditional paradigm, new product development (NPD) process is treated as a series of sequential and functional product development stages (Wheelwright and Clark, 1992). Information generated from one function transfers to the next one only after its completion, which results in poor coordination between development teams and bottlenecks of information flow (Hayes et al., 1988). It can significantly increase project cycle time.

Since the early 1990s, demanding market and short product life cycle in many industries have forced manufacturing firms to develop low-cost and high-quality products at a rapid pace. At the same time, the increasing technical intensity makes product development more complex. In order to deal with these issues, product development undergoes new trends, such as cross-functional team and concurrent product development. These new trends have increased the uncertainty and complexity of product development. Researchers now view product development as a collection of stages which are performed concurrently or iteratively. The product development processes and management practices created for relatively long product life cycle, stable market, and technology-based competition are no longer capable of producing products which can meet customer requirements in terms of time, cost, and quality (Clark and Fujimoto, 1991; Williams, 2005).

Improving development performance is becoming increasingly important and challenging. Part of the difficulty is caused by the internal structure of the product development process (Roberts, 1974; Ford and Sterman, 2003a). Well-intentioned changes to product development process may cause severe unintended side effects. For example, development stages may be concurrently executed to reduce project cycle time. However, in concurrent product development, a change in a stage will cause the rework in other development stages since they are usually dependent or interdependent. In the end, the overall development time is longer than otherwise. Therefore, many tools have been proposed to accelerate the NPD process and control the NPD cost, and prominent among these is the concept of concurrent engineering (CE). It has provided much success towards achieving shorter time-to-market (Clark and Fujimoto, 1991; Wheelwright and Clark, 1992; Smith and Reinersten, 1998; Bhuiyan, 2001). Overlapping of development stages, functional interaction, and frequent information exchange are among the elements that enable CE to improve the performance of product development (Blackburn, 1991; Bhuiyan, 2001).

Overlapping refers to a situation where the downstream development stages start prior to the completion of the upstream development stages. Overlapping is commonly found in many real life cases in order to overcome the obstacles faced in the sequential process (e.g. Krishnan et al., 1997). However, overlapping may increase rework because downstream work, started with preliminary information, may turn out to be wrong, because of changes or new insights in the upstream phase of development. Functional interaction, defined as the involvement of downstream engineers in upstream development, can reduce the rework incurred by the concurrent execution of development stages because upstream engineers can get more accurate input about requirements from later phases. As such, CE converts the sequential process into a more cooperative one, thus creating interdependencies between activities (Liker et al., 1996). Although the potential benefits of CE may be considerable, it becomes more challenging to coordinate such a process.

1.2 Research Gap

Traditional network-based scheduling techniques, such as Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT) (Moder et al., 1983; Badiru, 1993; Golenko-Ginzburg and Gonik, 1996), describe development processes which are relatively stable and sequential. These models were initially developed to control schedule, and later expanded to manage resources and costs. Rooted in the traditional sequential paradigm of product development, CPM disaggregates the development process into activities which are related through their temporal dependencies. In other words, the constraints are described as relationships between the beginning and completion of activities. Each activity is treated as a monolithic block of work described only by its duration. However, these models ignore the interactions between development stages, which are essential for concurrent NPD process (Rodrigues and Bowers, 1996; Ford and Sterman, 1998).

Recently, many analytical and simulation models have been developed to describe concurrent product development process and analyze the trade-offs among project cycle time, quality, and development cost. Smith and Eppinger (1997a, 1997b) developed several analytic models of sequential and parallel design iterations and addressed the effect of iterations among project phases on project cycle time with the Design Structure Matrix. Krishnan *et al.* (1997) proposed a framework to determine the optimal number and timing of information transfers. They showed that “upstream information evolution” and “downstream sensitivity” are the two properties affecting optimal overlapping strategies. Loch and Terwiesch (1998) adapted the concepts of evolution and sensitivity: “upstream information evolution” is defined as the continuous design modification process; “downstream sensitivity” represents the impact of a modification on downstream rework. Based on these concepts, they developed an analytical model and derived the optimal communication strategies for overlapped sequential process. Roemer et al. (2000) analyzed the time-cost tradeoffs in multistage product development. Chakravarty (2001) studied the trade-offs between the overlapping risk and the project time saved. Some special cases were analyzed to establish useful insights for sequential and overlapped processes. Bhuiyan et al. (2004) proposed a stochastic simulation model and discussed the impact of overlapping, functional interaction, upstream information evolution, and downstream sensitivity on three types of rework. Although the results of these efforts are insightful in many respects, we still can not derive appropriate overlapping degrees and functional interaction levels for the projects with different properties. This is because:

(1) Although existing models of concurrent product development describe the effects of upstream changes on downstream rework, most of these models (e.g. Williams

et al., 1995; Williams, 1999; Eppinger et al., 1994; Cho and Eppinger, 2005; Bhuiyan et al., 2004) use rework probability as input parameter which is difficult to be estimated directly since it is determined by the interactions of many parameters (such as completion quality, rework quality, and testing quality) (Krishnan et al., 1997; Joglekar et al., 2001). There is a need to make the interaction between development stages clear and analyze rework according to its root causes which would allow project managers to find appropriate policies for concurrent product development.

(2) While trade-offs among cycle time and development effort are necessary in product development, many studies only concentrate on project cycle time. Project policies which favor project cycle time may significantly affect other performance measures, such as the percentage of tasks requiring rework which is a key component for development effort. Consequently, there is a need to consider the effect on development effort or cost when trying to reduce the development cycle time (Smith and Reinertsen, 1998). Therefore, we need a model to estimate cycle time and development effort simultaneously so that managers can evaluate whether the overall benefit is greater than the investment involved.

(3) While the interaction between overlapping and communication is emphasized by many empirical studies, very few researchers have studied it in detail. It is clear that frequent information exchange can reduce rework in overlapped product development. However, communication also incurs time and cost. Tools are needed to balance these positive and negative effects and thus to derive appropriate overlapping and communication policies.

This thesis approaches the stated problems by explicitly modeling the interaction between consecutive development stages and the time-cost trade-off involved in CE. As a result, appropriate decisions on overlapping, communication, and functional interaction can be proposed.

1.3 Research Objective

Although successful new product development is critical to the survival of many companies, and much of previous research has focused on the development of technology and methods to support NPD management (e.g. Cooper, 1980; Steward, 1981; Eppinger et al., 1994; Repenning, 2001; Williams, 2005), our literature review shows that there is a lack of methods to explicitly model and analyze concurrent development processes. By modeling the effect of project properties (e.g. project uncertainty, dependency between development stages, and upstream information evolution) on project performance (project cycle time and development cost) this thesis investigates and suggests policies for managing and coordinating CE processes, and assesses the optimal or appropriate overlapping degree, communication frequency, and functional interaction level for the projects with different properties. The impact of project characteristics (such as project uncertainty, rework rate, and communication cost) on development policies is analyzed in an attempt to uncover insights on appropriate management of development projects within a given context.

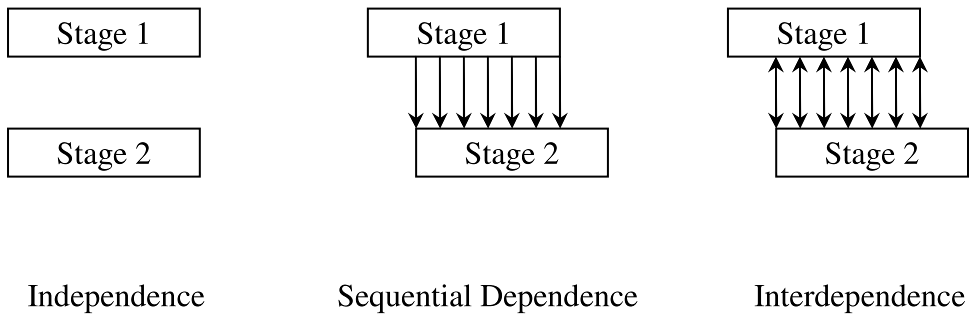


Figure 1.1 Independence, sequential dependence, and interdependence

An information-based view of product development is assumed in this thesis (Clark and Fujimoto, 1991). From this perspective, individual development activities are the information-processing units that receive information from their preceding stages and transform it into new information to be passed on to subsequent stages. Therefore the focus of the models is on the evolution of information and its impact on downstream rework. Information needs create dependencies between development stages which determine the product development structure. According to the information dependency between them the development processes can be classified as (see Figure 1.1): *Independence* if there is no information exchange between development stages; *Sequential dependence* if there is a unidirectional information flow; and *interdependence* if the stages are mutually dependent and the information flows in both ways (Thompson, 1967). Studies of concurrent engineering usually focus on dependent and interdependent development stages since the policies for independent stages are directly available.

Product development process can also be sorted by the communication cost, which is the fixed setup cost per information exchange (Ha and Perteus 1995, Loch and Terwiesch 1998). If a project is done by one team, then the communication cost is usually omitted. Related cases are proposed by Roemer et al. (2000), Krishnan et al.

(1997), Roemer and Ahmadi (2004). If a project is done by different teams, the communication cost should be considered. Related cases are proposed by Loch and Terwiesch (1998), Helms (2004). In this research, the dependent processes with low communication cost and the dependent processes with high communication cost are studied separately, since the models and policies for these processes are different. Consequently, three models are proposed to study the concurrent development processes with different information dependencies and/or communication cost:

- Firstly, this thesis presents an analytical model for managing concurrent development processes with sequential dependence and low communication cost. It is well known that continuous information exchange is optimal when communication cost is low (Roemer et al. 2000). Therefore the concurrent problem can be simplified into an overlapping problem regardless of communication strategies. The decisions on the degree of overlapping and the level of functional interaction are studied. The model has been applied to examine the development policies in a handset design company.

- Secondly, an analytical model for managing concurrent processes with sequential dependence and high communication cost is developed. In this case, the communication policy is extremely important. If information exchange is too frequent, then communication time and cost would increase significantly. However, infrequent information exchange would increase downstream rework. The model aims to optimize project performance by investigating the interaction between overlapping policy and communication strategy. The model was employed to analyze the development process of a large consumer electronics company.

- Finally a simulation model for managing overlapped iterative processes is developed. For iterative processes, the interaction is much more complex and analytical approaches have proved to be prohibitively expensive. Consequently, a System Dynamics model is built to manage concurrent processes composed of interdependent development stages. Using this model we can track the impact of different overlapping degrees and testing qualities on project performance. Therefore, it can help management to identify appropriate development policies. The model was implemented in a design house and led to marked improvement in project performance, thus demonstrating the viability of the model.

Note that depending on their newness to the company and marketplace, product innovations can be incremental or radical (Henderson and Clark, 1990; McDermott, 1999; Hauser et al., 2006). Radical innovation often requires developing products with an entirely new set of performance features (Leifer et al. 2000; Zhou et al. 2005). On the other hand, an extension or improvement of existing products is termed as incremental product innovation. Incremental product innovation plays a major role in the success of many organizations since the majority of so called ‘new’ products are in fact reworked versions of existing products (Ali, 1994; Griffin 1997; Grupp and Maital, 2001). This thesis focuses mainly on incremental innovation.

1.4 Research Approach

Mathematical and System Dynamics modeling methodologies are used to study different concurrent NPD processes. For sequentially dependent process, the interaction between development stages is relatively simple. Therefore, nonlinear programming is used to derive the development policies. Comparing to simulation methods, mathematical modeling is relatively simple. Furthermore, many useful insights can be derived by analyzing the mathematical models. However, for interdependent (or iterative) processes, the interaction is much more complex and thus analytical modeling is not suitable. Therefore System Dynamics modeling methodology is applied. All of the models are illustrated with case studies in consumer electronics industry.

1.4.1 Nonlinear Programming

Nonlinear programming is one of the basic methods of operation research. Through nonlinear programming, the models capture the relationship between project properties, development policies, and project performance. For the projects with low communication cost, a simple non-linear programming model is built. For the projects with high communication cost, a mixed-integer nonlinear programming model is developed.

The fundamental concept of the model is based on the premise that management makes decisions or chose actions (such as overlapping degree, communication frequency, and functional interaction level) that maximize project performance (measured in time and cost in this thesis).

1.4.2 System Dynamics

We simulate concurrent and interdependent product development processes by System Dynamics methodology. As such, the model serves as a framework for experimentation to test the effect of different development policies and activity properties on project performance. A computer simulation model provides several advantages. Firstly, many and various project parameters and dynamic relationships can be modeled more comprehensively with the flexible representation available than with manual or mathematical modeling methods. Secondly, unlike qualitative research, assumptions are made explicit and unambiguous in simulation models by their representation as formal equations. Thirdly, comparing to direct experiment, doing experiment through simulation is safe, replicable, low-cost and fast. Finally, the model's reflection of actual project structure provides an effective means of communicating research work and results.

System Dynamics (SD) methodology is used in this thesis. Discrete event simulation model and continuous time model (System Dynamics) are two methods commonly used to simulate NPD process. The former assumes that the product development process is composed of a finite set of activities and information flow only exists at the beginning or at the end of an activity. In contrast, the SD approach to project management treats the process of each phase as continuous work flow. It is consistent with the assumption in the overlapping models (e.g. Loch and Terwiesch, 1998; Roemer et al., 2000; Roemer and Ahmadi, 2004). Through building the relationship between work flow and information flow, we simulate the continuous upstream information evolution and its effect on downstream rework using SD approach.

1.5 Structure of the Thesis

This thesis consists of six chapters, consisting essentially of three parts, as shown in Figure 1.2. The thesis is organized as follows:

Chapter 1: Introduction presents the motivation for the research and details research objective, methodology, and structure. The research objective is to help management make decisions on overlapping degree, communication frequency, and functional interaction level in concurrent product development.

Chapter 2: Background on Previous Work reviews relevant literature of concurrent processes, traditional models of product development processes, and recent models for concurrent processes. The research gap is identified: current models do not allow explicit and clear modeling of the interaction between concurrent development stages. Consequently, managers can only make decisions on an ad hoc basis, leading to inefficient development policies. This research aims to solve the problem by developing formal models of concurrent processes. Three types of concurrent processes are studied: concurrent and sequentially dependent product development processes with low communication cost; concurrent and sequentially development processes with high communication cost; and iterative processes (or concurrent processes composed of interdependent development stages).

Chapter 3: Managing Concurrent Development Processes with Low Communication Cost presents an analytical model for managing dependent development stages in which the communication cost is low.

Chapter 4: Managing Concurrent Development Processes with High Communication Cost presents an analytical model for managing concurrent and sequentially dependent development processes with high communication cost.

Chapter 5: A System Dynamics Model of Overlapped Iterative Processes develops a simulation model for managing overlapped iterative processes. In Chapters 3 and 4, analytical models are built for managing concurrent and sequentially dependent product development processes. For interdependent product development processes, the interaction is much more complex and thus analytical modeling is not suitable. Consequently, a System Dynamics model is built in this chapter. Note that using this method we can only find the best solution within different scenarios and thus the solution is not globally optimal. The model was illustrated with a case study at a design house.

Chapter 6: Conclusions and Future Study gives a summary of this research. We first summarized the results derived on the models and case studies and discussed the contributions of this study. Then, we point out the limitations of this research. The directions for future study are discussed in the last section.

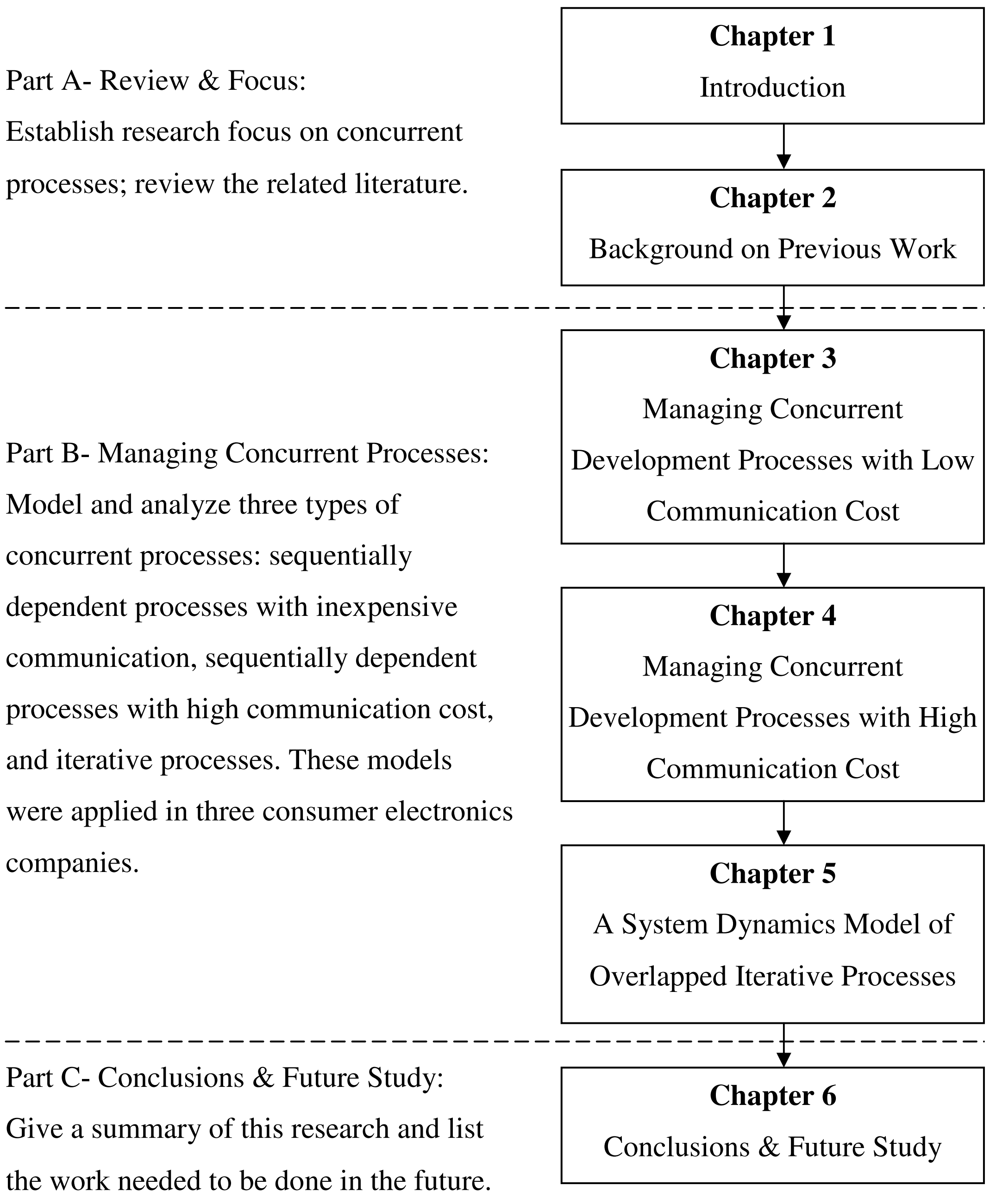

Part A- Review & Focus:
Establish research focus on concurrent processes; review the related literature.
Chapter 1
Introduction
Chapter 2
Background on Previous Work
Part B- Managing Concurrent Processes:
Model and analyze three types of concurrent processes: sequentially dependent processes with inexpensive communication, sequentially dependent processes with high communication cost, and iterative processes. These models were applied in three consumer electronics companies.
Chapter 3
Managing Concurrent Development Processes with Low Communication Cost
Chapter 4
Managing Concurrent Development Processes with High Communication Cost
Chapter 5
A System Dynamics Model of Overlapped Iterative Processes
Part C- Conclusions & Future Study:
Give a summary of this research and list the work needed to be done in the future.
Chapter 6
Conclusions & Future Study

Figure 1.2 Structure of the thesis

CHAPTER 2

BACKGROUND ON PREVIOUS WORK

In this chapter, an extensive review of the relevant theoretical and analytical research in NPD is presented. The chapter begins with a review of research in traditional sequential development processes, followed by research in concurrent development processes which have appeared in the last two decades. These reviews provide the basis for the evaluation of various product development models which investigate the impacts of project properties and development policies on project performance. This is followed by a detailed evaluation of existing descriptive, analytical, and simulation models of NPD processes. Some concepts in the concurrent engineering literature, which are closely related to this research, are illustrated in detail.

2.1 Traditional Sequential Development Processes

As shown in Figure 2.1, traditional models of product development processes are based upon a sequential and functional approach to product development (Wheelwright and Clark, 1992). In the traditional paradigm, the development processes are treated as a series of development activities from conceptualization to mass production. This is represented by the unidirectional arrows between phases in Figure 2.1. Many researchers have described the traditional process and have given examples from different industries (e.g. Wheelwright and Clark, 1992; Womack et al., 1990; Nevins and Whitney, 1989; Hayes et al., 1988). Clark and Fujimoto (1991) argue that this paradigm is appropriate "…when markets were relatively stable, product life cycles were long, and customers concerned most with technical performance."

The sequential process is highly functionally segregated, i.e. different functions have responsibility for different phases, with formal communication between the functions occurring at the end of each phase (at the gates, or the milestones) when one function hands off its work to the next. Typically, the functions responsible for the various phases are: marketing personnel for the concept phase and launch phase, design engineers for design phase, test engineers for the prototype testing phase, and manufacturing personnel for the pilot production phase.

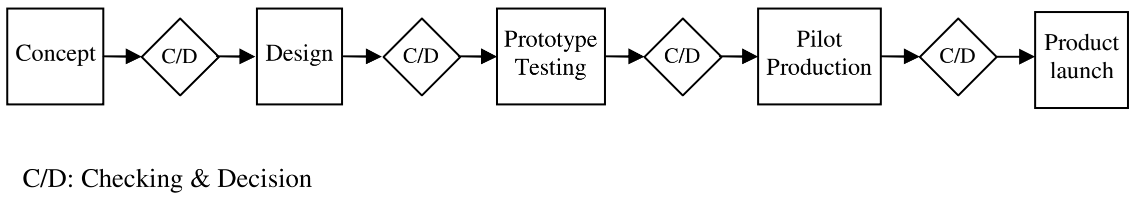


Figure 2.1 A schematic diagram for a phase-milestone NPD process

Substandard project performance under the traditional paradigm generates friction and conflicts among different function groups, resulting in poor coordination and bottlenecks in the flow of information through the product development processes (Hayes et al., 1988). This can extend the project cycle time or consume additional resources, thereby increasing costs.

2.2 Concurrent Development Processes

Market and technology changes have brought about new characteristics of product development. The most significant changes from the traditional paradigm to the new paradigm are from sequential development process to concurrent process. Overlapping and functional interaction are two of the most important components of concurrent development. Researchers now view product development as a collection of highly coupled development stages which are performed iteratively and often simultaneously by cross-functional development teams (Wheelwright and Clark, 1992; Womack et al., 1990).

2.2.1 Overlapping of Development Stages

Overlapping refers to the product development process where the downstream stage starts prior to the completion of the upstream stage. The primary purpose of adopting overlapping approach is cycle time reduction through planning and executing multiple stages simultaneously instead of sequentially as in a sequential development process. This requires starting downstream stage as soon as preliminary information is available. For the overlapped process, the development stages are usually sequentially dependent or interdependent. Information generated by one or more stages poses contingencies for others; thus, all the development stages should be considered simultaneously (Adler, 1995).

Although large reduction in cycle time can be realized by applying overlapping approach (Wheelwright and Clark, 1992; Womack et al., 1990; Nevins and Whitney, 1989), the cycle time reduction comes at the cost of increased complexity. Overlapping increases the dependency between development stages and the number of required information transfers. To deal with the increased interdependencies, intensive coordination is required. However, this may increase the cost of manpower. Because downstream is started on preliminary information in the overlapped process, the amount of rework is likely to increase when new information becomes available. Researchers suggest that iteration in product development is a primary cause of the dynamic nature of product development, a primary driver of project cycle time and a measure of process quality (Cooper, 1994, 1993a, b, c; Bhuiyan et al., 2004). Figure 2.2 shows an overlapped concurrent development process. Information flows between tasks are more frequent than in a sequential process. When quality problems are found by downstream stages, the relevant information is transferred to the stages which are responsible for the quality problems and then rework occurs.

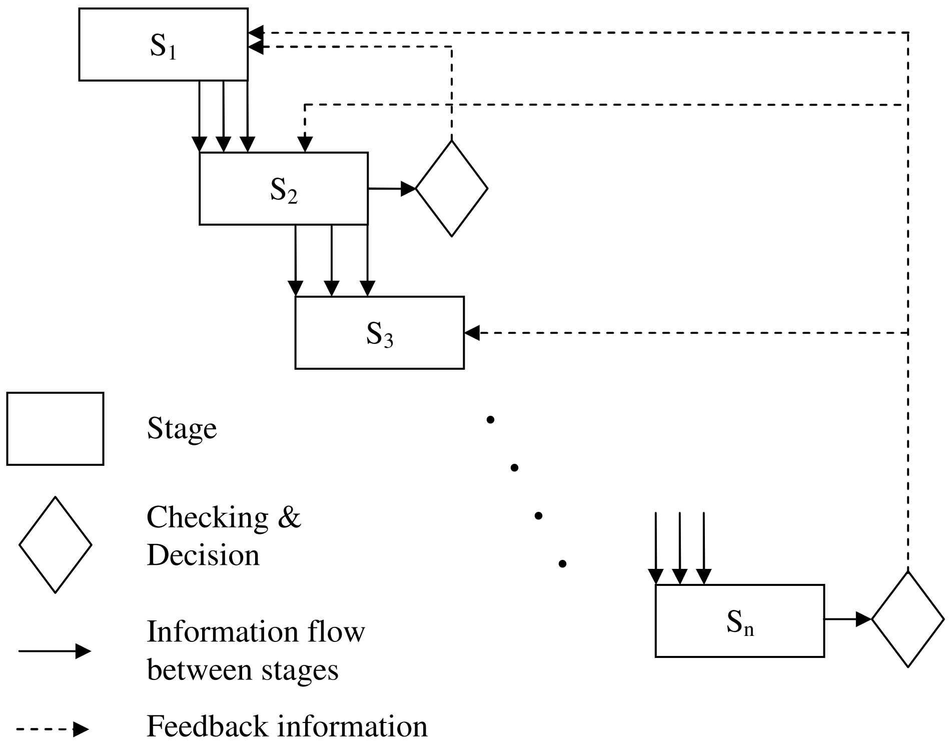


Figure 2.2 Concurrent process

2.2.2 Cross-Functional Teams

In today's product development, functional participation takes place through the formation of teams consisting of representatives from the functions involved. Due to uncertainty in product development processes, the release of preliminary information to downstream functions may introduce the need for rework when there is a change in preliminary information. The goal of functional interaction is to reduce project uncertainty by identifying the potential quality problems as early as possible. The formation of cross-functional teams is an extension of the move away from function-based teams to the matrix structures. Hayes et al. (1988) describe and Wheelwright and Clark (1992) later refine a detailed model of this shift by introducing intermediate steps defined by the level of influence of project managers. Restructuring product development organizations away from function-based groups and toward cross-functional development teams has become a widely used approach to reduce project cycle time (Clark and Fujimoto, 1991).

However, researchers (Clark and Fujimoto, 1991; Dean and Susman, 1991; Takeuchi and Nonaka, 1991) have realized that the formation of cross-functional teams alone does not necessarily reduce time-to-market. They found that over-extended communication and coordination in cross-functional team may lower project performance. Dean and Susman (1991) found that friction between the members from different functions may affect the efficiency of product development. Nevin et al. (1991) listed some other reasons for the cross-functional team failures.

The new development paradigm addresses the increased coordination needs of projects with cross-functional development teams. The apparent assumption is that project uncertainty, which is a driver of rework, can be reduced by using cross-functional teams. However, functional interaction also increases communication time and cost. Empirical studies show that functional interaction may increase (Eisenhardt and Tabrizi, 1995; Von Corswant and Tunälv, 2002), decrease (Bhuiyan et al., 2004; Wagner and Hoegl, 2006), or have no significant effect (Datar et al., 1997) on project performance. These mixed results indicate that cross-functional team is not a panacea for managing NPD projects. The functional interaction policy should be adjusted according to project characteristics. Thus potential risks must be carefully examined to ensure that added time and effort are kept to a minimum (Krishnan et al., 1997).

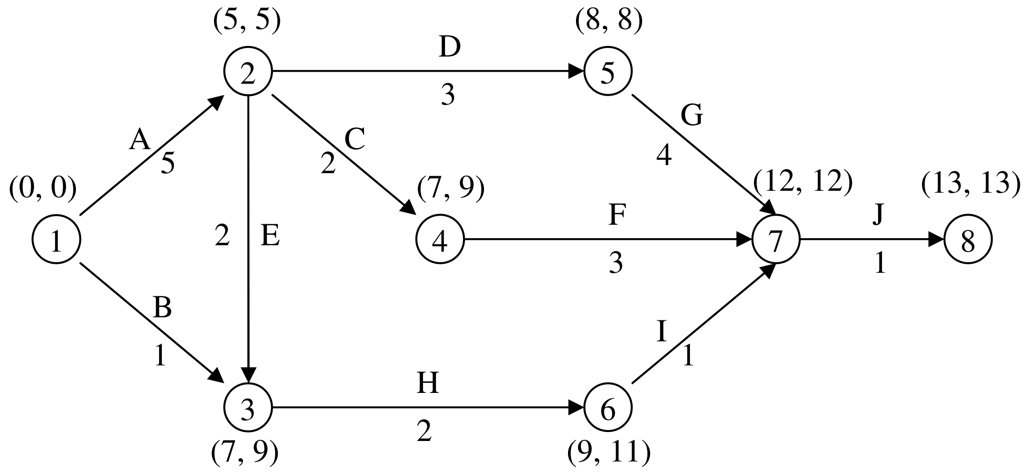
2.3 Previous Models for Managing Development Projects

In order to control project schedule or analyze the effect of different policies on NPD performance (in terms of project cycle time, and cost), various models for NPD process management have been developed. We group these models into five categories: network-based scheduling techniques (e.g. Moder et al., 1983; Badiru, 1993; Golenko-Ginzburg and Gonik, 1996), design structure matrix (DSM) (e.g. Eppinger et al., 1994; Cho and Eppinger, 2005), analytical models (e.g. Smith and Eppinger, 1997a, 1997b), discrete event simulation models (e.g. Bhuiyan et al., 2004), and System Dynamics (SD) models (e.g. Cooper, 1980; Ford and Sterman, 1998; Williams, 2005).

2.3.1 Network-based Scheduling Techniques

The Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT) are two of the most important network-based scheduling techniques which have been widely used to manage development projects. These methods were initially developed to control schedule, and later expanded to manage resources and costs.

Rooted in the traditional paradigm of product development, the Critical Path Method disaggregates the development process into activities which receive upstream information at the beginning and transfer the output to the downstream in the end. Each activity is treated as a monolithic block of work described only by its duration. The temporal dependencies between development activities describe the constraints which upstream activities impose on downstream activities. The logic of the schedule can be represented in a network diagram. A simple example is shown in Figure 2.3.



①②③ ···	Nodes
A, B, C ···	Activities
1, 2, 3 ···	Durations
(0, 0), (7, 9) ···	Earliest and latest event time

Figure 2.3 A network diagram for CPM schedule management

Critical Path Method enables the identification of a project's critical path, which is the sequence of tasks whose combined durations define the minimum project cycle time. Earliest and latest possible start and finish times of all activities determined by the critical path can be calculated, as can the available slack times. Furthermore, the Critical Path Method provides some tools for studying the trade-offs of different performance measures, such as project cycle time and development efforts. For example, durations of activities along the critical path can be shortened by using more resources (Wheelwright and Clark, 1992; Moder et al., 1983). Through Critical Path Method, time-cost trade-offs can be analyzed and the effectiveness of accelerating alternative activities can be determined. In addition, the effects of altering dependencies among development activities on time-to-market reduction can be investigated (Moder et al., 1983).

The Critical Path Method can be easily understood and applied in practice. However, the method has several crucial limitations. It assumes that all quality problems can be discovered and solved before the task is completed, and upstream information only be sent to the downstream activities when it is finalized. As a result, the method can not describe concurrent processes in which upstream changes will cause significant downstream rework. Secondly, the Critical Path Method assumes that the duration of each activity is directly available. This prevents the method from modeling and studying the underlying factors determining activity duration, such as development efficiency, development quality, and project uncertainty. Therefore the Critical Path Method is unable to model the dynamic nature of concurrent development processes.

PERT addresses one of the limitations of the Critical Path Method by incorporating the effect of project uncertainty in the estimates of the duration of development activities. It was developed for processes such as product development (Moder et al., 1983). Three estimates (most likely estimate, optimistic estimate and pessimistic estimate) are used to describe the variability of activity durations. Based on these data, the probability of a project meeting specific schedule objectives can be derived. The incorporation of duration uncertainty makes PERT more valuable in managing the projects with uncertainty. However, for most development projects, the delay is usually caused by rework not by the change of activity duration. Like the Critical Path Method, PERT cannot explicitly represent the dynamic interaction between development activities, as well as the rework caused by upstream changes.

2.3.2 Design Structure Matrix

The iterative nature of product development can be addressed using Design Structure Matrix (an example is shown in Figure 2.4) (Smith and Eppinger, 1997; Eppinger et al., 1994; Steward, 1981). The DSM method is based on the earlier work in large-scale system decomposition (Ledet and Himmelblau, 1970; Sargent and Westerberg, 1964). The DSM provides a compact representation of a complex system by showing information dependencies in a square matrix with the full set of development activities as both row and column labels. Activity names are usually listed to the left of the matrix. A mark in an off-diagonal cell represents an information transfer between two development activities/stages. For each activity, its row represents its input and its column shows its output. When activities are listed in temporal order, sub-diagonal marks represent an input from upstream activities/stages to downstream activities/stages. Super-diagonal marks denote a feedback from downstream activities to upstream activities.

The DSM approach, first introduced by Steward (1981) and further developed for large projects by Eppinger et al. (1994), spawns dozens of research efforts on organizing product development tasks. DSM has been used to map and predict information flows among activities (Morelli, Eppinger and Gulati, 1995). It can also be used to investigate different strategies for managing product development projects. Osborne (1993) applied iteration maps and the Design Structure Matrix to describe product development at a leading semiconductor firm Intel, in terms of cycle time. Osborne's work demonstrates the need for further investigation on the impacts of dependencies among development tasks on project cycle time. It also points to the need for a better understanding of how key factors which impact cycle time can be identified and managed. Smith & Eppinger (1997a, 1997b) presented two analytical extensions of the DSM method. In the first model, they used Eigen-structure analysis to identify controlling features of iteration in product development projects. In the second model, the ordering of tasks was manipulated and an expected duration for each task sequence was calculated using Reward Markov Chain. More recently, Yassine, Falkenburg, and Chelst (1999) utilized a two-dimensional variable to measure the dependency strength between design tasks. Ahmadi et al. (2001) addressed the dynamic rework probabilities. A recent survey by Browning (2001) shows the increasing use of DSM method for project planning and management. Chen et al. (2004) proposed an approach to quantify the dependency between design tasks in a DSM. Abdelsalam & Bao (2006) proposed a framework to determine the sequence of activities that minimizes project cycle time given stochastic task durations.

DSM is potentially a useful tool in describing and investigating information transfer and iteration for cycle time reduction. However, DSM cannot directly model the development process over time. Like the Critical Path Method, DSM assumes that the dependencies between tasks, the development speed of every task, and the probability of rework are fixed.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Prepare DR&O	1	■	×	×		×	×	×							
Create Design Architecture	2	×	■												
Distribute Models and Drawings	3	×		■											
Analyses & Evaluation	4	×			■	×	×	×						×	×
Create Structural Geometry	5				×	■									
Prepare for FEM	6				×		■								
Structural Design Conditions	7				×			■							
Weights & Inertial Analyses	8				×				■					×	×
S&C Analyses & Evaluation	9									■					
Free-body Diagrams & Loads	10										■				
Internal Load Distributions	11											■			
Strength, Stiffness, & Life	12												■		
Manufacturing Planning	13								×					■	
UCAV Proposal	14								×						■

Figure 2.4 DSM representation of UCAV preliminary design process (Browning and Eppinger, 2002)

2.3.3 Analytical Models

Previous empirical studies showed that overlapping of consecutive development stages can reduce project cycle time at the cost of additional development effort (Clark & Fujimoto, 1991; Smith & Reinertsen, 1998; Sobek et al., 1999; Helms, 2004). Eisenhardt & Tabrizi (1995) observed that the effect of overlapping is closely related to the uncertainty of development projects in computer industry. Based on the empirical study of 140 development projects in the electronics industries, Terwiesh & Loch (1999) concluded that overlapping is effective only if uncertainty can be resolved quickly. Clark and Fujimoto (1991) identified that the negative effect of concurrent execution can be reduced through frequent information exchange.

Based on these empirical studies and literature, a significant amount of research has been conducted on how to determine the optimal development strategies for concurrent processes. We group them into three categories: overlapping sequentially dependent stages, overlapping interdependent stages, and communication policies.

- **Overlapping Sequentially Dependent Stages**

Krishnan et al. (1997) proposed a framework to determine the optimal number of information transfers and start time of downstream iteration so as to minimize project cycle time. They proposed that “evolution” and “sensitivity” are the properties which determine optimal overlapping. The former is the rate at which upstream information converges to a final solution, and the information is modeled as an interval that gets refined over time (see Figure 2.5). They distinguish between fast evolution and slow evolution. In the case of slow evolution, major changes still happen in the end of upstream development. Sensitivity describes how vulnerable the downstream stage is to any changes in the upstream information, and is defined by the time needed by the downstream stage to incorporate changes. They also distinguish between high and low sensitivity, where high sensitivity means that a change early in the upstream process has a large impact on the downstream process and low sensitivity means that a change early in the upstream process has a small impact on the downstream process.

Krishnan et al. (1997) addressed the overlapping problem by studying how values of the evolution and sensitivity patterns determine the extent to which overlapping is appropriate between two sequentially dependent stages. An integer program was developed to study the effect of overlapping policies on project cycle time, assuming upstream evolution and downstream sensitivity are known initially. The method is illustrated with an example of the door outer panel development process.

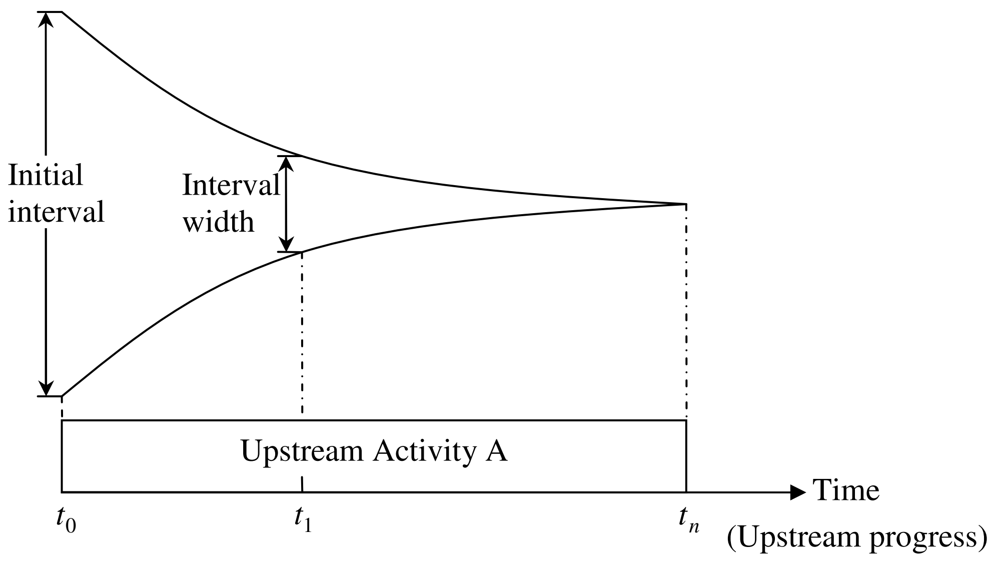


Figure 2.5 Upstream evolution
(Krishnan et al. 1997)

In practice, evolution and sensitivity are not always easy to define quantitatively. Therefore, the authors developed a conceptual framework to address the overlapping and communication strategies. Four communication and overlapping policies for the projects with different evolution and sensitivity properties were proposed (Figure 2.6). When the evolution is slow and sensitivity is low, the stages can be overlapped by starting downstream stage with preliminary information, and incorporating upstream modifications in subsequent downstream iteration. It is defined as *iterative overlapping*. If evolution is fast and sensitivity is high, then the exchanged information should be preempted by taking its final value at an earlier point in time. It is called *preemptive overlapping*. When the upstream evolution is slow and the downstream sensitivity is high, either we can disaggregate the upstream information and transfer part of the finalized information to the downstream at an earlier point in time or sequential process can be applied. This approach is called *divisive overlapping*. Finally, if the upstream evolution is fast and the downstream sensitivity is low, then the downstream stage can start earlier and the upstream information can be preempted. This case is called *distributive overlapping*.

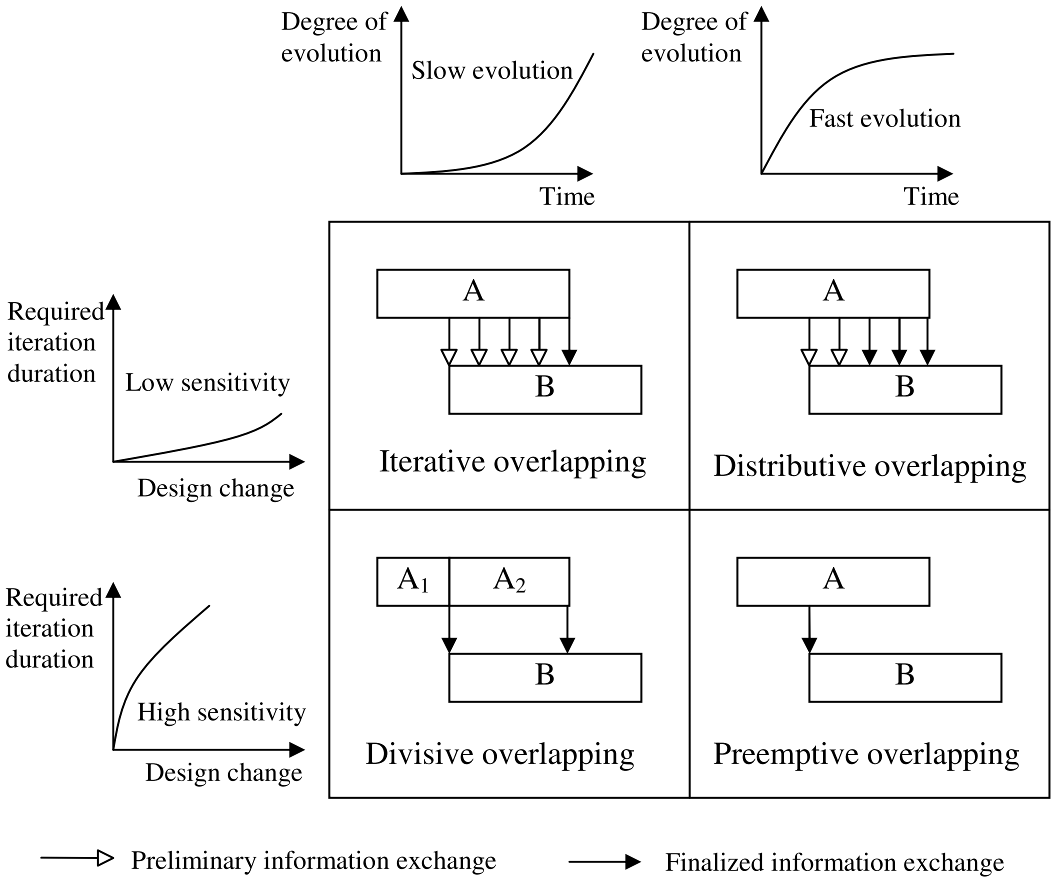


Figure 2.6 Development policies based on evolution and sensitivity (Krishnan et al. 1997)

The aim of overlapping is to reduce the cycle time of a project. Besides benefits there are also risks to the overlapped execution of development processes. This risk is a result of iterations that occur in product development. As long as a modification takes place before the start of the downstream stage it only affects the upstream process.

However, if the modification occurs after the preliminary information is released the upstream as well as the downstream process will be affected by the modification. In the worst case the cycle time of the overlapped process exceeds the lead-time of the sequential process. Therefore, it is important to determine beforehand whether it is worth the risk to overlap processes.

However Krishnan et al.'s framework only addresses the appropriate policies when evolution is extremely slow or fast and sensitivity is extremely low or high. For these cases, the trade-offs are obvious. However, for most development projects, these extreme situations almost never happen. Therefore analytical models are needed to investigate the time-cost trade-off in concurrent processes in detail.

Following Krishnan et al.'s framework, Loch & Terwiesch (1998) have developed an analytical model of concurrent process which consists of two sequentially dependent stages. In their work, "upstream evolution" is defined as the continuous design modification process; "downstream sensitivity" represents the impact of a modification on downstream rework. Based on these concepts, Loch & Terwiesch presented an analytical model to determine appropriate overlapping and communication strategies. They suggest that if engineering changes arise after the start of downstream stage, this poses the risk of redoing the downstream work. The risk can be high if the dependency between the stages is high. They propose that communication can reduce the risk of downstream rework, but at the cost of communication time.

Loch & Terwiesch developed an analytical model that results in optimal overlapping and communication strategies for the projects with different properties, such as project uncertainty, upstream evolution, and downstream sensitivity. Uncertainty is measured by the average modification rate of upstream information. It is defined as a nonhomogeneous Poisson process. Evolution speed represents the rate at which the uncertainty is reduced. The total amount of uncertainty can be reduced through communication in the form of meetings.

One of the key assumptions of Loch and Terwiesch's model is that the later the upstream modifications arrive, the more difficult it is to deal with them. However their mathematical model can not capture this feature, which leads to wrong conclusions in their study. The details will be illustrated in Chapters 3 and 4.

Since then, a number of innovation researchers have studied the optimal overlapping strategies for sequentially dependent product development processes. For example, Roemer et al. (2000) analyzed the time-cost trade-offs in multistage overlapped processes by assuming that the downstream rework can be directly estimated by project engineers. The interdependencies between overlapping and crashing, which are two commonly used methods for reducing project cycle time, were studied by Roemer & Ahmadi (2004). Chakravarty (2001) studied the optimal overlapping policies in three overlapping modes assuming that the incompatibility among development stages is estimable. While these works have shed light on the analysis of product development process, it is still not clear how the probability of rework can be properly assessed in practice.

A common assumption made in this stream of overlapping models is that downstream stages will not feedback information to its corresponding upstream stages. Hence, these models can not effectively deal with interdependent development stages, which are quite common in complex development projects.

- **Overlapping Interdependent Development Stages**

Yassine, Chelst, and Falkenburg (1999) used risk and decision analysis methodology to determine the optimal overlapping policy for a set of activities. Using a probabilistic model consisting of an upstream stage and a downstream stage, their methodology finds the optimal overlapping strategy based on the study of independent, sequentially dependent, and interdependent stages/activities. They proposed three categories of information structures: sequential, partial overlapping, and concurrent. They proposed that sequential process takes place for dependent stages; partial overlapping can take place for either sequentially dependent stages or interdependent stages; concurrent execution is only suitable for independent development stages. These propositions describe how the development stages should be overlapped. However, this paper did not address the key question for project management: how much to overlap. The extreme points of partial overlapping are sequential development and concurrent development. Therefore, the authors didn't make it clear how NPD process should be organized for different projects.

Carrascosa et al. (1998) presented a model to estimate project cycle time for different task sequences and overlapping degrees using concepts of probability of change and impact. However, the assumption made in their study is that there is only one parameter causing a task to change during the evolution of each task, which limits its application. Joglekar et al. (2001) presented the performance generation model (PGM) to derive insights on optimal concurrency strategies between coupled development stages under a deadline. Bhuiyan et al. (2004) proposed a stochastic simulation model and discussed the impact of overlapping and functional interaction on project performance. Wang & Yan (2005) focused on the optimization of the concurrency between an upstream activity and a number of downstream activities. These models were built on the assumption that rework probability is estimable. However, it is still not clear how the probability of rework can be properly assessed in practice.

- **Communication Policies**

Facilitating communication among business functions and/or members in cross-functional teams are commonly used by many companies (Cooper, 1994; Swink et al., 1996; Minderhoud & Fraser, 2005). It is well known that communication among development teams can reduce project uncertainties, but at the expense of additional time and cost for communication. Patrashkova-Volzdoska et al. (2003) reported that communication frequency and performance are nonlinearly dependent with an inverted-U relationship, based on a survey of 60 cross-functional teams. Helms (2004) observed that the information exchange among development teams is time consuming in chemical industries.

In spite of its importance, the issue of communication policies has been addressed only to a limited extent in the analytical literature. Ha & Porteus (1995) developed an analytical model and studied the benefit of early detection of upstream flaws through overlapping and frequent communication between development stages. In their study, the development stages are assumed to be interdependent. In contract to sequentially dependent stages, the nature of interdependent activities requires more frequent communication. If information exchange is too frequent, then communication time and cost would increase significantly. However, infrequent information exchange would delay the identification of the design flaws and increase the amount of rework of the upstream stage. Given these trade-offs, they seek to determine the optimal communication frequency that minimizes the expected project completion time. A dynamic program was developed and it showed that the overlapped development must be accompanied by progress reviews to minimize the risk of downstream rework. Moreover, the frequency of communication or progress reviews must be balanced with the value gained from having them. With appropriate overlapping and communication policies, project cycle time can be reduced without significantly increasing the risk of downstream rework. Loch & Terwiesch (1998) extended the work of Ha & Porteus, and developed an analytical model to determine the optimal overlapping degree and the communication frequency between upstream and downstream stages.

2.3.4 Discrete Event Simulation Models

Another stream of research uses simulation to explore the linkage between task sequences and project performance. Discrete event simulation (DES) model and System Dynamics (SD) model are two methods commonly used to simulate NPD process. Discrete Event Simulation (DES) model usually assumes that the PD process is composed of a finite set of activities and information flow only exists at the beginning or at the end of an activity.

Bhuiyan et al. (2004) developed a DES model to study the impact of rework on development cycle time and effort (man power). This model demonstrates the relationships between overlapping policy, functional interaction strategy, project cycle time, and development effort. However, this stochastic model cannot directly simulate the structure of a development process over time. The dependencies between stages, the development speed of every stage and the probability of rework cannot be adjusted continuously over the development process. This model cannot be used to study complex development processes because the building blocks of the model are developed based on simplified stage-gate processes.

Some other discrete event simulation models have been developed to study product development projects. Browning & Eppinger (2002) highlighted the effects of varying process architecture by simulating NPD process as a network of activities that exchange deliverables. The model outputs sample cost and schedule outcome distributions. Each distribution is used with a target and an impact function to determine a risk factor. Alternative process architectures can be compared to reveal opportunities to trade cost and schedule risk. Gil et al. (2004) simulated the concept development process for semiconductor fabrication facilities, and found that some decision-making postponement can help increase the predictability of concept development duration and reduce resources spent in design without increasing the risk of exceeding project deadlines. Cho & Eppinger (2005) extended the work of Browning & Eppinger (2002) by accounting for resource constraints.

2.3.5 System Dynamics Models

Many models of product development have been built on System Dynamics approach. In this section, we generally introduced the System Dynamics models of product development. For more detailed review, readers can refer to the work done by Ford (1995) and Chi (2001).

Roberts (1974) developed a small model with System Dynamics approach to investigate the management of R&D projects. They assume that each activity or stage is composed of many "job units" which are uniform in size. The completion rate of the "job units" is determined by available manpower and productivity. Management decisions (such as the change of manpower) are based on perceived progress, which includes both actual progress and perceptual errors.

Cooper (1980) and Reichelt (1990) described the framework of large System Dynamics models developed by Pugh-Roberts Associates for claims settlement of large scale shipbuilding operations. The structure of the model was further illustrated in Cooper, 1993a, b, c. Cooper (1980, 1993a, b, c) simulated the major phases of shipbuilding operation and modeled the impacts of rework in projects on cycle time and development cost. He distinguishes between the activities of the initial completion of development tasks and rework and discusses the rework caused by customer changes. Project phases are dependent in this model. Therefore, engineering changes may propagate across project phases if they are not identified on time. A delay in discovering engineering changes increases the total amount of rework, reduces project quality, and slows the completion of the project. Reichelt (1990) describes the dependency of downstream product development stage on its preceding stage in a two-stage process. Cooper and Reichelt's research adds several valuable concepts to the literature of project management:

1) Customer changes will significantly increase the time for coordination and rework, and thus increase project cycle time and cost;
2) Upstream changes will not only increase the rework of that stage but also increase the rework of all the downstream stages depending on the output of the stage;

3) Manpower diverted from other development stages disrupts work flow and requires additional labor, lowering the average skill level and thus reducing development rate and increasing development cost.

Richardson and Pugh (1981) developed a System Dynamics model of single R&D projects. The rework process is modeled in detail, which expands the resource effectiveness portion of the fundamental structure used by Roberts. Richardson and Pugh distinguish between tasks-done-correctly and tasks-requiring-rework. Project cycle time and the amount of rework are significantly affected by the error rate in product development and the discovery rate of development errors. Based on their model, Richardson and Pugh illustrate the effects of different assumptions and policies on project cycle time and identify some impractical policies which were obtained on invalid assumptions of rework or project scope.

Abdel-Hamid (1984) built a model of software development to better understand the root causes of cost overruns, late deliveries, and user dissatisfaction. In this model, software production is influenced by human resource management, planning, and controlling. Schedule pressure increases the predicated amount of work force which is necessary to complete the project on schedule. Schedule pressure also influences the productivity of work force, the error generation rate, and the resource allocation in different development activities.

The impacts of project manager motivations on project performance are investigated by Jessen (1990) with a model based upon resources strategy, targets, and rework. This model focused on the roles of balancing feedback loops in projects. It improves our understanding of the motivational structures in projects.

"Gate functions" were introduced by Homer et al. (1993) to describe the constraints on the progress of a development stage imposed by both the progress of the preceding stages and the work within the stage. In this model, graphical table functions are used to describe these precedence relationships in more detail than possible with the Critical Path or PERT methods (Ford, 1995). For example, the work availability of a downstream stage can be affected by the progress of the upstream stage throughout the duration of the downstream phase. However, using CPM or PERT, the information transfer only occurs at the start or finish of a development stage. Homer et al.'s model uses both available work and resources to constrain progress. We adopted this structure in the SD model to describe information prerequisites.

Ford et al. (1993) did a case study in an electronic entertainment equipment manufacturer and investigated the interface between two product development groups. The relationships among coordination, schedule and project quality are focused in this model. The authors explicitly modeled rework due to development errors and operational iteration. They distinguished between required and voluntary iteration in product development. Based on this model, they studied the influence of schedule pressure on decisions about iteration for quality. The time-quality tradeoff was also discussed in this model.

Whilst the above studies have covered the three domains (monitoring and control, rework and human resources) as identified by Rodrigues and Bowers (1996), it is Ford and Sterman (1998) who tie these together with a process structure. Ford and Sterman (1998) developed a product development model which includes all four performance drivers: process structure, resources, targets and scope. They have demonstrated the importance of integrating process structure with resource, scope, and targets in dynamic models of development projects. More specifically, Ford and Sterman (1998) explicitly describes the interdependency of the tasks within a phase and the links among multiple project phases.

Chi (2001) proposed that the segregation of knowledge is the endogenous factor which results in an inherent need for iteration. They developed a System Dynamics model to investigate the impacts of such knowledge segregation on the performance of product development projects. This proposed framework helps management to manage the knowledge within NPD projects.

The existing System Dynamics literature has a rich history of modeling development projects. All these models contribute to the description and documentation of the tight linkage between development resources, resource management, and project performance. However, current literature can not explicitly simulate the rework generation process in concurrent product development. This limits the application of SD models in product development projects since CE has already been applied in most development projects. The SD model developed in Chapter 5 addresses it by explicitly modeling the interaction between overlapped development stages. It will help management derive appropriate development policies and thus improve project performance.

2.3.6 Summary of the Models for Product Development

Five different methods to address the development sequence and strategies have been suggested in the literature. CPM and PERT are the methods developed for sequential process and thus they are not applicable for concurrent product development. DSM is a descriptive method which is used in our thesis to represent the general relationships of the development stages in a mobile phone development company in Chapter 5. DSM is not suitable for studying overlapping and communication policies directly, since it can not simulate the interaction between development stages. The continuous nature of NPD process also makes DES not suitable. Therefore, in this thesis, only DSM method, analytical modeling method, and SD approach are applied. DSM is used to describe the general process of product development; two analytical models and a SD model are developed to study the optimal/appropriate development policies for different concurrent processes.

2.4 A Framework to Study Concurrent Processes

Although many models have been developed to study new product development, current models cannot capture the rework generation process in concurrent processes. Therefore, management still lacks of tools to derive appropriate overlapping and communication strategies for concurrent processes.

However the current literature does give some useful concepts for modeling concurrent processes:

1) Krishnan et al. (1997) proposed that “evolution” and “sensitivity” are the properties which determine optimal overlapping.

2) The information evolution process for product development projects is usually available and it is one of the key factors affecting downstream rework (Krishnan et al., 1997; Roemer et al., 2000).
3) Terwiesch and Loch (1999) and many other researchers (e.g. Carrascosa et al., 1998) make it clear that the downstream sensitivity is linearly related to downstream progress.

These concepts are supported by empirical studies (e.g. Eisenhardt and Tabrizi, 1995; Helms, 2004), and have had a strong influence on the literature on concurrent engineering. They are the start point of this thesis.

2.5 Summary of Literature Evaluation

This chapter documents the shift from sequential product development processes to dynamic development processes because of the trends of concurrent development. It identifies several fundamental features of concurrent development processes such as overlapping of development stages, cross-functional team, and rework due to upstream changes.

Traditional project management models based on the Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT) assume the process is in a static fashion with activity duration estimates and precedence relationships describing the network of development activities. Consequently, these models can not deal with concurrent product development.

Although some researchers have built models to study the relationship between the features of concurrent processes and NPD performance, most of these models simply assume that the rework probability is directly estimable. However, from our case study and previous empirical literature, it is almost impossible to estimate these parameters directly, which leads to the fact that although there are several models built for analyzing concurrent processes, most of them cannot be applied directly in practice (Jun et al., 2005). This research proposes three models to describe concurrent development processes with different types of information dependencies. Using these models, we can derive appropriate development policies based on the data available for most development projects such as historical data of modifications, and dependence between development stages. The applicability and usefulness of these models are illustrated and validated with the case studies in three companies developing consumer electronics products.

CHAPTER 3

MANAGING CONCURRENT DEVELOPMENT PROCESSES WITH LOW COMMUNICATION COST

This chapter presents an analytical model for managing concurrent and sequentially dependent development processes in which communication cost is low. Firstly, the relevant literature is reviewed and the research gap is illustrated. Then, we investigate the downstream progress in detail, which is essential to derive the optimal overlapping policies. We find that downstream progress increases over time when upstream evolution is fast or linear, but it is indefinite when upstream evolution is slow. After that, we present the optimal overlapping policies taking into account the complexity of downstream progress. The impact of different project properties (such as dependency between development stages, rework rate, and opportunity cost of time) on overlapping policies is discussed. Finally, we derive the optimal functional interaction strategies under the condition that optimal overlapping is followed. The methodology is illustrated with a case study at a handset design company.

3.1 Introduction

The increasing importance of rapid product development has given rise to a large body of literature dedicated to development cycle time reduction (Krishnan and Ulrich, 2001; Carrillo and Franza, 2006). Overlapping and functional interaction are two of the most popular approaches which are considered necessary to accelerate product development process and reduce project uncertainty (e.g. Wheelwright and Clark, 1992; Ganapathy and Goh, 1997; Smith and Reinertsen, 1998; Joglekar and Ford, 2005). Overlapping refers to the product development processes where the downstream stage starts prior to the completion of the upstream stage. Functional interaction is defined as the involvement of downstream engineers in upstream development.

Despite the popularity of overlapping and functional interaction, empirical studies show that the gain from these approaches must be weighed against the additional time and cost for rework and communication (Cordero, 1991; Bhuiyan et al., 2006; Eisenhardt and Tabrizi, 1995; Von Corswant and Tunälv, 2002). Overlapping allows the downstream stage to start earlier at the cost of additional rework arising from subsequent modifications of the upstream stage (Krishnan et al., 1997; Helms, 2002). For example, mould fabrication can start before the detailed design is finalized so that development cycle time may be reduced. However, when the product design is changed the mould will have to be revised or re-fabricated, which is costly and time consuming. Overlapping may not be beneficial when the upstream information changes significantly and/or the downstream stage is highly dependent on upstream output. Functional interaction reduces upstream uncertainty and downstream rework at the cost of additional time for communication and cooperation (Loch and Terwiesch 1998). Consequently, analytical investigation of the trade-offs involved is needed. Note that without overlapping, rework also occurs (e.g. product design may be changed because it doesn't fit with product equipment requirements). In this thesis we only focus on the rework which is caused by overlapping.

Some analytical models have been developed to determine the optimal overlapping and functional interaction levels. We sorted them into two groups. In the first group, researchers assume that the total amount of downstream rework for different projects with different overlapping degrees is directly estimable. For example, Roemer et al. (2000) analyzed the time-cost trade-offs for multistage overlapped processes by assuming that the downstream rework can be estimated by project engineers. The interdependencies between overlapping and crashing were studied by Roemer and Ahmadi (2004). Chakravarty (2001) studied the optimal overlapping policies in three overlapping modes assuming the incompatibility among development stages is estimable. While these works have shed light on the analysis of product development process, it is still not clear how the probability of rework can be determined in practice (Jun et al., 2005).

On the contrary, the second group of studies assumes that the rework probability is unavailable in practice but can be derived according to certain project properties, such as upstream evolution, project uncertainty, and dependency between development stages. Krishnan *et al.* (1997) developed a model-based framework to manage the overlapping of two development stages. They showed that "upstream evolution" and "downstream sensitivity" are the two properties determining optimal overlapping strategies. Loch and Terwiesch (1998) adapted the concepts of evolution and sensitivity: "upstream evolution" is defined as the speed at which the preliminary information is modified; "downstream sensitivity" represents the impact of a modification on downstream rework. Based on these concepts, they developed an analytical model and derived the optimal communication strategies for overlapped processes. These studies are supported by empirical studies (e.g. Eisenhardt and

Tabrizi, 1995; Terwiesch and Loch, 1999; Helms, 2004), and have had a strong influence on the literature on concurrent engineering.

The model developed in this chapter follows the work of Krishnan *et al.* (1997) and Loch and Terwiesch (1998) but differs from their research in three important aspects.

Firstly, we model the impact of an upstream change on downstream rework as a function of the downstream progress and the dependency of overlapped stages (Terwiesch and Loch, 1999; Thomke and Bell, 2001; Lin et al., 2008). Downstream progress is the amount of work which is completed but not corrupted by upstream changes. Dependency describes the percentage of the downstream progress which will be affected by a change. Figure 3.1 shows the progress of stage *i*. For the purpose of explanation, we assume that two changes happen at t_1 and t_2 (in reality, changes have a probability of happening at any time). At time t_1, the progress of stage *i* would be 5 if there is no change. If a change takes place at t_1 and the dependency is 20%, the progress will be dropped to 4 and 1 day's rework arise. The same dynamics take place at t_2. From this example we can see that the progress of the downstream stage is dynamically affected by project properties (such as upstream evolution and dependency). Krishnan et al. (1997) simplified the impact of upstream changes on downstream rework by assuming information updating was allowed only when an iteration of the downstream stage was completed (i.e. the downstream progress is 100%). Therefore they can use a single parameter, "sensitivity", to represent the impact of upstream modifications on downstream rework. For the simplicity of their model, Loch & Terwiesch (1998) assume that the impact of engineering changes is linearly related to downstream time, i.e. they implicitly assume that downstream progress equals downstream time. This assumption does not guarantee the fact that the downstream progress can not be negative at any time. For example, in Figure 3.1, 10 modifications arrive at time t_2. It is clear that up to 10 days of downstream work can be affected by upstream modifications, since the downstream progress can not be negative. However, based on their assumption, a total of 20 days of rework arises if the dependency is 20% (i.e. the downstream progress becomes -10 days at t_2). Unlike their research, we investigate the downstream progress in detail. Such analysis is essential to derive the optimal overlapping and functional interaction strategies (Carrascosa et al., 1998).

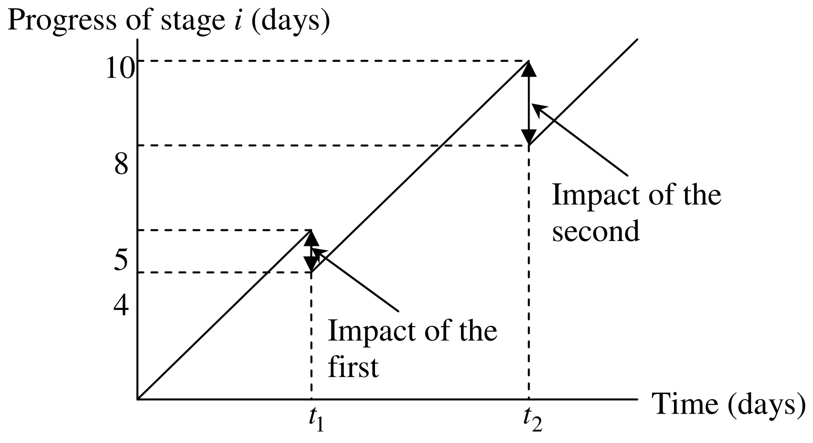


Figure 3.1 The progress of a downstream stage

Secondly, our model is more general with respect to the evolution patterns studied. Eppinger et al. (1994) pointed out that, in a typical design project, most engineering changes arise somewhere in the middle of the project which implies that the evolution function may be S-shaped. Krishnan et al. (1997) defined two types of upstream information evolution: “fast evolution” and “slow evolution”. If the modification rate decreases with time, the evolution is said to be fast. In contrast, the modification rate increases over time in slow evolution. Loch and Terwiesch (1998) studied the communication and overlapping policies for the projects in which the modification rate is linearly related to development time. While making specific assumption of the upstream evolution can help to derive management insights, it also limits the application of the model. Our model is developed based on a general assumption of the upstream evolution: the modification rate is non-negative, which ensures our model to be a good abstraction of practice.

Thirdly, project performance is measured by time and cost in this model. There are three traditional measures of project performance: time, cost, and quality. For most derivative projects the target quality is often well defined. Hence, the objective for management is to achieve the required quality with less time and cost. However, most of the extant literature, such as Krishnan *et al.* (1997) and Loch and Terwiesch (1998), takes development cycle time as the sole measure for project performance (Gerwin and Barrowman, 2002). Unlike their research, we explicitly address the trade-off between cycle time and cost. Furthermore, the relationship between project properties (e.g. uncertainty, dependency, and evolution) and project performance is explored. (As we know, the time-cost trade-off is also studied by some other researchers, such as Roemer et al. (2000) and Chakravarty (2001). However all of these studies are developed based on the assumption that the total amount of downstream rework is directly estimable and thus the effect of uncertainty, upstream evolution, and dependency on development cycle time and cost is not explicitly explored.)

This chapter is organized as follows. Section 3.2 introduces an analytic model which captures the relationship between upstream evolution, dependency of development stages, overlapping and functional interaction policies, and project performance.

Section 3.3 discusses the progress of downstream stage and the earliest downstream start time in detail. In section 3.4, we present the optimal overlapping and functional interaction policies, taking into account the complexity of downstream progress. Then two closely related problems are discussed. In section 3.6, the methodology is illustrated with a case study at a handset design company. Conclusions are summarized in section 3.7.

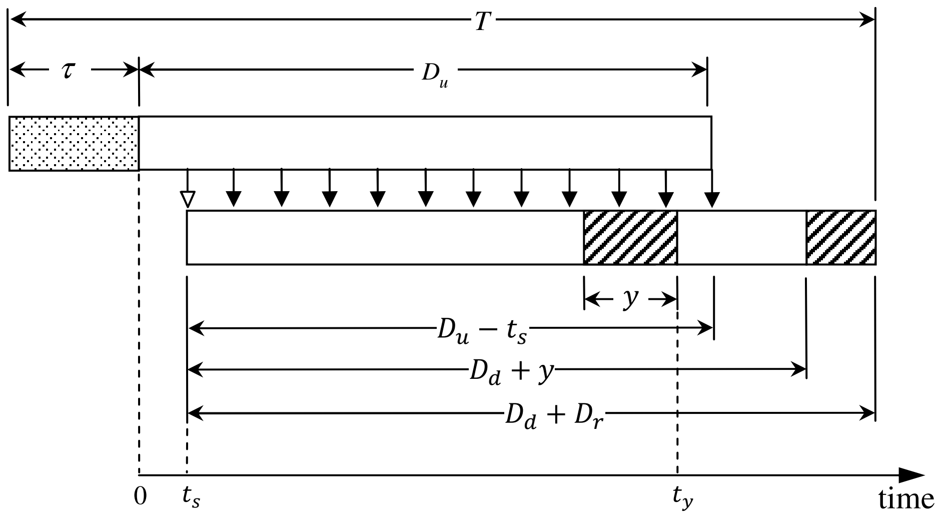
3.2 Model Formulation

In this section we first formulate the tradeoffs of cost and development cycle time. Then, based on a review of relevant literature (e.g. Krishnan et al., 1997; Loch and Terwiesch, 1998; Carrascosa et al., 1998) we introduce a general mathematical model of upstream evolution and its impact on downstream rework. The objective function, constraint equations, model parameters, and decision variables are summarized at the end of this section.

3.2.1 Overview of the Model

The theoretical basis of our model is the information-based view of product development (Clark and Fujimoto, 1991), in which individual development stages are viewed as information-processing units that receive information from upstream stages and transform it into new information to be passed on to downstream stages.

Figure 3.2 shows the information transformation between two overlapped development stages. Unlike the traditional sequential process, in overlapped product development, a downstream stage can start on preliminary information. Without loss of generality, we assume preliminary information of the upstream is available at time zero and it is continuously modified until the end of the upstream stage. D_u and D_d denote the durations of initial development of upstream and downstream stages respectively, where the duration of initial development is the expected time it takes to perform a stage without overlapping. The downstream stage is dependent on the upstream stage and cannot be finished earlier. Consequently, the earliest start time of the downstream stage t_e satisfies $t_e \leq D_u$. The exact point of t_e is discussed in Section 3.3 and can be derived on Proposition 3.3. The decision variable t_s, which is the start time of downstream stage, ranges between t_e and D_u. A smaller t_s corresponds to higher overlapping degree.



——▷	Preliminary information	——►	Modified information
0	The time when preliminary information is available		**Decision variables**
D_u	Upstream development duration	y	Time spent on rework in the overlapped period
D_d	Downstream development duration	t_s	Start time of downstream stage
D_r	Downstream rework duration	t_y	Finish time of y
T	Development cycle time	τ	Functional interaction duration

Figure 3.2 Overlapped product development process

In Figure 3.2, functional interaction and upstream development are sequential because they are usually sequentially done by one team. It is consistent with previous studies (e.g., Loch and Terwiesch, 1998). Functional interaction can also be applied in downstream and it will affect the overlapping between downstream stage and the later development stages. In this model, we only study the overlapping between two development stages. Therefore, functional interaction in downstream is not studied.

Without considering the drawbacks of overlapping, the development cycle time of the two stages can be reduced to $t_s + D_d$. However, because downstream stage starts with preliminary information, upstream changes may lead to rework at the downstream stage. The downstream rework duration is denoted by $D_r(t_s, \tau)$. Functional interaction reduces upstream uncertainty and downstream rework but increase interaction time τ (Ha and Porteus, 1995; Loch and Terwiesch, 1998). Taking into account of the rework duration and functional interaction time, the product development time for concurrent approach is:

$$T_c = \tau + t_s + D_d + D_r(t_s, \tau) \qquad (3.1)$$

As we know development cycle time for traditional sequential process is $T_s = D_u + D_d$. Thus, the reduced cycle time due to overlapping and functional interaction is:

$$T_s - T_c = D_u - \tau - t_s - D_r(t_s, \tau) \qquad (3.2)$$

Defining c_t as the opportunity cost of time, the gain from reducing development cycle time is $c_t(D_u - \tau - t_s - D_r(t_s, \tau))$. Let c_τ and c_r be the unit cost of functional interaction and rework respectively. The additional cost for overlapping and functional interaction is $c_\tau \tau + c_r D_r(t_s, \tau)$. The overall gain from these approaches can be formulated by summing the negative and positive effects of overlapping and functional interaction, i.e.

$$Max: G = c_t(D_u - \tau - t_s - D_r(t_s, \tau)) - c_\tau \tau - c_r D_r(t_s, \tau) \tag{3.3}$$

3.2.2 Upstream Information Evolution

In product development, modifications are inevitable because of the high uncertainty of development projects. We define upstream evolution as the arriving rate of upstream changes. These changes affect the preliminary information based on which downstream stage has begun work. These changes follow a nonhomogeneous Poisson process with rate $\mu_\tau(t)$ (see Loch & Terwiesch (1998), Carrascosa et al. (1998), and Jun et al. (2005) for the justification of the Poisson assumption). Consistent with Krishnan et al. (1997) and Loch & Terwiesch (1998), the evolution is said to be fast if the modification rate $\mu_\tau(t)$ decreases with t. In this case, the preliminary information gets close to its final form rapidly. Slow evolution indicates that $\mu_\tau(t)$ increases with t. The evolution is linear, if the modifications are generated as a homogeneous Poisson process.

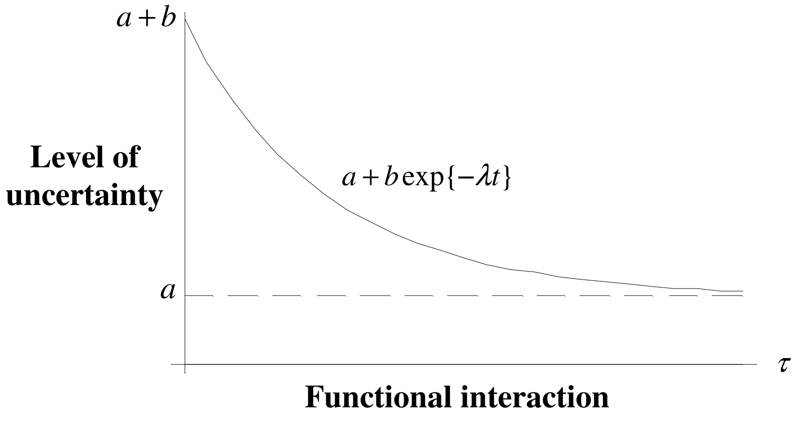


Figure 3.3 Impact of functional interaction on uncertainty

Clausing (1994) identified two types of uncertainties. One is the result of the fact that it is difficult to make the design concept very clear at the beginning. The other is the modifications caused by problems regarding manufacturability and maintainability of the product. Previous studies (e.g. Clausing, 1994; Adler, 1995; Bhuiyan et al., 2004) have shown that the second type of uncertainty can be reduced through functional interaction, and the uncertainty reduction usually exhibits diminishing returns (see Figure 3.3). Consistent with these empirical results, the relationship between modification rate and functional interaction is modeled as:

$$\mu_\tau(t) = \frac{a + b\exp\{-\lambda\tau\}}{a+b}\mu_0(t) \tag{3.4}$$

$a+b$ represents the overall level of uncertainty in the absence of functional interaction. The uncertainty which can be reduced through functional interaction is captured by b. Parameter λ represents the cross-functional teams' capability to reduce upstream uncertainty. $\mu_0(t)$ denotes the modification rate if there is no functional interaction ($\tau = 0$).

3.2.3 Downstream Rework

Downstream rework arises when the modifications of upstream stage corrupt the relevant tasks in the downstream stage. In other words, these tasks need to be reworked because they were based on incorrect information. Since there are numerous tasks in a project, it is infeasible to tract tasks individually. Consistent with previous literature (e.g. Chakravarty, 2001), we describe workload of development stage in terms of duration. That is the workload equals the duration of initial development. For example, we can say there is 10 days' work. In this model, the workloads of upstream and downstream are D_u and D_d respectively. Consequently the initial development rate of downstream, which is the regular speed of downstream development without overlapping, is equal to one. Let r denote the downstream rework rate and $r \geq 1$ because of the learning effect. As shown in Figure 3.2, $D_u - t_s$ is the overlapped period of two consecutive development stages. y and $D_u - t_s - y$ denote the time spent on rework and initial development in the overlapped period respectively. Therefore, the total amount of tasks completed in the overlapped period equals $D_u - t_s - y + ry$. $P(D_u)$ is the downstream progress at time D_u, i.e. the amount of work which is completed on preliminary information but not affected by upstream modifications. The total amount of rework and rework duration are:

$$R(t_s, \tau) = D_u - t_s + (r-1)y - P(D_u) \tag{3.5}$$

$$D_r(t_s, \tau) = \frac{R(t_s, \tau)}{r} \tag{3.6}$$

where y is constrained by the total amount of tasks can be reworked at time t_y.

$$y \leq \frac{t_y + (r-1)y - P(t_y)}{r}. \tag{3.7}$$

To derive the rework duration $D_r(t_s, \tau)$, it is essential to determine the expression of downstream progress. Downstream progress rate is composed of two key elements: development rate which is one for initial development and r for rework, and corruption rate which is the effect of upstream modifications on the progress of the downstream stage. Corruption rate is determined by modification rate, $\mu_\tau(t)$, dependency between development stages, k, and the progress of downstream stage, $P(t)$ (Terwiesch and Loch, 1999; Carrascosa et al., 1998). More completed downstream work will be corrupted when modification rate and dependency are high. The progress of downstream stage accounts for the reason why additional rework arises in overlapped product development. For traditional sequential process, modifications are found and resolved before the downstream stage starts, i.e. the changes are made when the downstream progress is zero. However, in overlapped process, the downstream stage starts on preliminary information in order to reduce development cycle time. Rework arises when modifications are made after the downstream stage starts (Krishnan et al., 1997; Lin et al., 2008). The downstream progress rate is formulated as follows:

$$\frac{\partial P(t)}{\partial t} = 1 - k\mu_\tau(t)P(t) \text{ for } t < t_y - y \text{ or } t_y < t < D_u \tag{3.8}$$

$$\frac{\partial P(t)}{\partial t} = r - k\mu_\tau(t)P(t) \text{ for } t_y - y \le t \le t_y \tag{3.9}$$

Equation (3.8) represents the progress rate of initial development. Equation (3.9) denotes the progress of rework. These Equations show the dynamic relationship between progress rate and downstream progress. Based on these equations we can derive $P(t)$, the downstream progress at time t. Let $t = D_u$, we can derive $P(D_u)$. Then using equation 3.5, we can derive the total amount of rework. In section 3.4, the downstream progress will be investigated in greater detail.

3.2.4 Summary

Based on the elements discussed above, we formulate the optimization problem as follows:

$$\max : G = c_t(D_u - \tau - t_s - \frac{R(t_s,\tau)}{r}) - c_\tau \tau - \frac{c_r R(t_s,\tau)}{r}; \tag{3.10}$$

subject to: Equations (3.4), (3.5), (3.7), (3.8),(3.9);

$$t_e \le t_s \le D_u; \tag{3.11}$$

$$\tau \ge 0; \tag{3.12}$$

Table 3.1 Model parameters and decision variables

Parameters	**Definition**
D_u	Upstream development duration
D_d	Downstream development duration
y	Time spent on initial development in the overlapped period
$\mu_\tau(t)$	Rate of upstream modifications
a	Modifications irreducible through functional interaction
b	Modifications reducible through functional interaction
λ	Functional interaction capability
c_t	Opportunity cost of time (the cost of increasing one unit of time-to-market)
c_τ	Functional interaction cost
c_r	Rework cost
k	Dependency between development stages (the percentage of downstream progress which will be affected by a modification)
r	Rework rate of downstream stage
Decision Variables	**Definition**
y	Time spent on rework in the overlapped period
t_s	Start time of downstream stage
t_y	Finish time of y
τ	Functional interaction duration

All variables are summarized in Table 3.1. Our model focuses on the effects of overlapping and functional interaction on cost and cycle time. It is employed to answer the following managerial questions:

(1) What is the relationship of downstream progress with time, upstream uncertainty, dependency of development stages, and other project properties?

(2) How can we maximize project performance through overlapping?

(3) What is the relationship between project properties (e.g. modification rate, dependency, rework rate, and rework cost) and overlapping strategy?

(4) How can we maximize project performance by applying overlapping and functional interaction simultaneously?

3.3 Downstream Progress and Earliest Start Time

To understand the impact of an upstream modification on downstream rework, the downstream progress should be investigated first. Assume that the functional interaction duration τ and the start time of downstream stage t_s are given. In this section, we first discuss the priority ordering of initial development and rework. Then, we derive the expression of downstream progress $P(t)$, the total amount of downstream rework $R(t_s,\tau)$, and the earliest downstream start time. The results are summarized in Propositions 3.1, 3.2, and 3.3.

Proposition 3.1

For downstream development, initial development is prior to rework.

Proof. To enhance readability of the thesis, all proofs of this chapter are shown in Appendix A.

In overlapped product development, tasks can be continuously corrupted by upstream modifications. Therefore at certain time, engineers can do the design tasks that have yet to be completed or the design tasks that have been completed but are corrupted by upstream modifications. Proposition 3.1 proves that the traditional working sequence (rework follows initial development) (Black and Repenning, 2001) is optimal when rework rate is not lower than initial development. An empirical support of this result is provided by Repenning (2001). Furthermore, if rework is prior to initial development, rework and initial development will be completely interwoven. Frequent changes from initial development to rework and from rework to initial development can significantly increase setup time, which primarily comprises time for procuring specialized tooling for build operations, as well as time and cost penalties associate with an engineer's learning curve (Chakravarty, 2001; Meixell et al. 2006). Moreover, for some tasks, frequent redoing is not allowable. For example, frequent change of hard mold will damage it and thus causing a tremendous increase of cost. Therefore, in practice, rework usually begins after the initial tasks are finished (e.g. Black and Repenning, 2001; Lin et al., 2008). Note, for many projects, a small percentage of rework, which constraints the initial development, is done before finishing the initial development. For modeling simplicity, we ignore the trivial effect of it on project performance. Proposition 3.1 is the base for the following results. Moreover, it can be directly applied for guiding resource allocation among development activities.

Proposition 3.2

(*a*) *Downstream progress at time* $\bar{t}$ *is*

$$\begin{cases} P(\bar{t}) = \exp\{-k\int_{t_s}^{\bar{t}} \mu_\tau(t)dt\}\int_{t_s}^{\bar{t}} \exp\{k\int_{t_s}^{t} \mu_\tau(x)dx\}dt & \text{if } \bar{t} \le \min(t_s + D_d, D_u) \\ P(\bar{t}) = \exp\{-k\int_{t_s}^{\bar{t}} \mu_\tau(t)dt\}\int_{t_s}^{D_d+t_s} \exp\{k\int_{t_s}^{t} \mu_\tau(x)dx\}dt & \\ \qquad + r\cdot\exp\{-k\int_{t_s}^{\bar{t}} \mu_\tau(t)dt\}\int_{D_d+t_s}^{\bar{t}} \exp\{k\int_{t_s}^{t} \mu_\tau(x)dx\}dt & \text{if } t_s + D_d \le \bar{t} \le D_u \end{cases}$$

(*b*) *The total amount of downstream rework is*

$$\begin{cases} R(t_s,\tau) = D_u - t_s - \int_{t_s}^{D_u} \exp\{-k\int_{t}^{D_u} \mu_\tau(x)dx\}dt & \text{if } t_s \ge D_u - D_d \\ R(t_s,\tau) = D_d - \int_{t_s}^{D_d+t_s} \exp\{-k\int_{t}^{D_u} \mu_\tau(x)dx\}dt & \\ \qquad + r[D_u - t_s - D_d - \int_{D_d+t_s}^{D_u} \exp\{-k\int_{t}^{D_u} \mu_\tau(x)dx\}dt] & \text{if } t_s \le D_u - D_d \end{cases}$$

(*c*) *The progress of downstream stage increases over time when $\mu_\tau(t)$ is nonincreasing with t, but it is indefinite when $\mu_\tau(t)$ increases with t.*

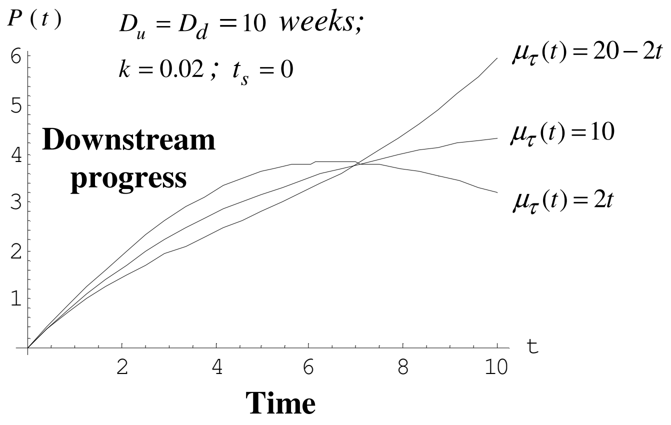


Figure 3.4 Downstream progress: numerical example

Proposition 3.2(a) expresses the downstream progress at time t . Only initial development should be done in the overlapped period when $t_s \geq D_u - D_d$, since initial development is prior to rework. When $t_s \leq D_u - D_d$, the time spent on rework in the overlapped period is $D_u - D_d - t_s$. By these concepts, the downstream progress at time t is derived. Then, by equation (3.5) and Proposition 3.2(a), the total amount of rework is derived in Proposition 3.2(b). Proposition 3.2(c) describes the change of downstream progress over time. Downstream progress increases monotonously over time when the upstream evolution is fast or linear. However, for slow evolution, the corruption rate may dominate at the end of the project which leads to the decrease of downstream progress. It is further illustrated with a numerical example in Figure 3.4, which shows that the change of downstream progress over time is complex. Detailed investigation of it is essential to derive optimal overlapping policies.

Proposition 3.3

Define the earliest start time as the time which ensures development teams work continuously. The earliest start time of downstream stage is zero if $D_u \le D_d$ *or*

$$r\int_{D_d}^{D_u} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt + \int_0^{D_d} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt \le D_d$$

Otherwise t_e *must satisfy the following equation:*

$$r\int_{D_d+t_s}^{D_u} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt + \int_{t_s}^{D_d+t_s} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt = D_d$$

Furthermore t_e *can be easily found through a simple binary search, because the left-hand side of the equation is strictly decreasing with* t_s*.*

Proposition 3.3 clearly defines the earliest downstream start time. This provides the base for analyzing optimal overlapping policies. In the next section, we try to find the exact downstream start time which maximizes project performance.

3.4 Analysis of the Optimal Policies

3.4.1 Optimal Overlapping

Assume the functional interaction duration τ is exogenously determined. With Propositions 3.2 and 3.3 we can now determine the optimal overlapping degree.

Proposition 3.4

(*a*) *The objective function G is concave with respect to* t_s*, i.e., there is a unique optimum* t_s *in the interval* $[t_e, D_u]$.

(*b*) *The optimal start time of downstream stage, denoted by* t_s^**, equals to* t_e *if*

$t_e \ge D_u - D_d$ and

$$-c_t + \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\}] \le 0 \tag{3.13}$$

or

$t_e \le D_u - D_d$ and

$$c_r - \frac{(r-1)(c_t + c_r)}{r}\exp\{-k\int_{D_d+t_e}^{D_u} \mu_\tau(t)dt\} - \frac{c_t + c_r}{r}\exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt \le 0 \tag{3.14}$$

(*c*) *The optimal start time of downstream stage should be* t_e, *when* $r \ge 1 + \frac{c_r}{c_t}$. *This conclusion is independent of the properties of uncertainty, dependency, and upstream evolution.*

(*d*) *If* $t_e \ge D_u - D_d$ and $-c_t + \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\}] > 0$

The optimal start time can be derived from the following equation:

$$\int_{t_s^*}^{D_u} \mu_\tau(t)dt = -\frac{1}{k} \cdot \ln(1 - \frac{r \cdot c_t}{c_t + c_r}) \tag{3.15}$$

If $t_e \le D_u - D_d$ *and*

$$c_r - \frac{(r-1)(c_t + c_r)}{r}\exp\{-k\int_{D_d+t_e}^{D_u} \mu_\tau(t)dt\} - \frac{c_t + c_r}{r}\exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\} > 0$$

Through a simple binary search, the optimal start time can be derived from the following equation

$$c_r - \frac{(r-1)(c_t + c_r)}{r}\exp\{-k\int_{D_d+t_s}^{D_u} \mu_\tau(t)dt\} - \frac{c_t + c_r}{r}\exp\{-k\int_{t_s}^{D_u} \mu_\tau(t)dt\} = 0 \tag{3.16}$$

Proposition 3.4(b) defines the necessary and sufficient condition for complete overlapping (starting the downstream at time t_e). Roemer et al. (2000) proved that complete overlapping is optimal for time-to-market problems when $r = 1$. We generalize their conclusion in Proposition 3.4(c). This result can be easily used by management to check the applicability of complete overlapping, since it does not relay on any information about evolution and dependency and reduces the number of parameters that must be estimated to three.

As we discussed above, overlapping may significantly increase downstream rework and thus complete overlapping may not be the best solution for certain development projects. Proposition 3.4(d) determines the optimal downstream start time for the projects for which complete overlapping is unsuitable. As shown in Figure 3.5, if the conditions in Proposition 3.4(d) are satisfied, project performance increases from D_u to t_s^* but it decreases from t_s^* to t_e. This is because as overlapping degree increases, the negative effect of more downstream rework begins to outweigh the positive impact of starting the downstream stage earlier. The optimal solution can be derived from equation (3.15) or (3.16).

Proposition 3.5

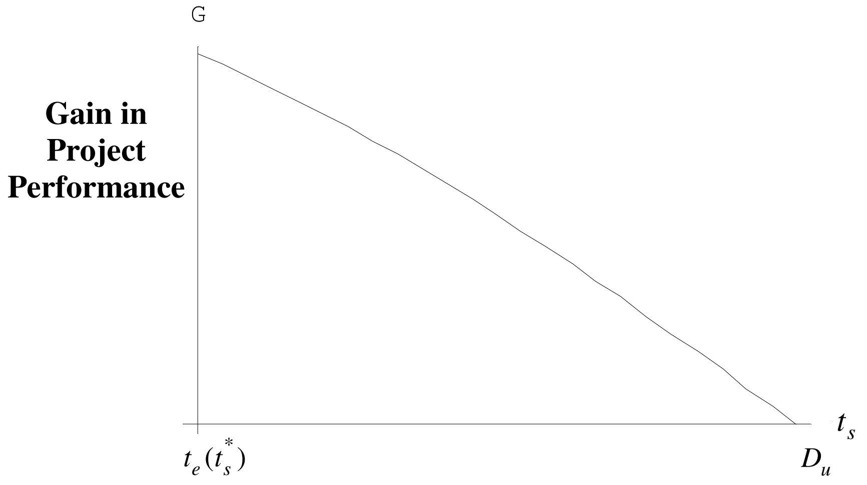
(*a*) *Higher dependency, k, or modification rate, $\mu_\tau(t)$, decreases the optimal overlapping degree. Furthermore faster evolution increases the optimal overlapping degree.*

(*b*) *Higher rework rate r increases the optimal overlapping degree.*

(*c*) *The start time of the downstream stage is earlier when the opportunity cost of time, c_t, increases or the rework cost, c_r, decreases.*

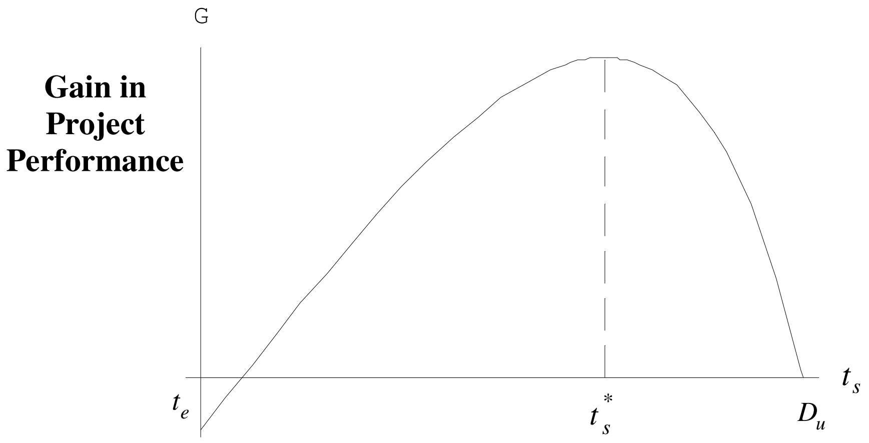
(*d*) *The optimal overlapping policy is independent of the downstream development duration D_d, if D_d changes within $[D_u, +\infty)$.*

$$-c_t + \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\}] \le 0 \; and \; t_e \ge D_u - D_d$$



Start time of downstream

$$-c_t + \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\}] > 0 \; and \; t_e \ge D_u - D_d$$



Start time of downstream

Figure 3.5 Optimal start time of downstream stage

Proposition 3.5(a) describes the impact of upstream evolution and dependency of development stages on optimal overlapping. It is consistent with previous research (e.g. Krishnan et al., 1997; Bhuiyan et al., 2004) which concludes that projects should start on preliminary information when sensitivity is low and evolution is fast. However sequential process is better for projects with high sensitivity and slow evolution.

Proposition 3.5(b) explains the reason why overlapping strategies do not work well for distributed and cross-company development projects. Rework caused by engineering changes usually needs close cooperation between functional teams. Divergent cultural values, geographical distance, and goal incongruity make the rework process much more time consuming in distributed environment than it in collocated environment. Thus, project delay becomes a common problem for distributed projects. In order to avoid delay, complete overlapping may be applied. However, our analytical result shows that lower rework rate decreases the optimal overlapping degree. When rework rate is low, increasing overlapping degree may increase development cost significantly and thus lower project performance. The efficient way to increase project performance for distributed product development is to increase rework rate by clearly defining responsibilities and benefits, building trust between functional teams and so on (O'Sullivan, 2003). Accelerating rework rate will not only reduce rework time but also increase the optimal overlapping degree which leads to further reduction of development cycle time.

Minderhoud and Fraser (2005) showed that although sequential process is still favorable for some development projects, most projects are shifting to concurrent process especially for consumer electronics products. Many other empirical studies (e.g. Smith and Reinertsen, 1998; Cohen et al., 1996) recognized that time-to-market is becoming more and more important for today's development projects. Proposition 3.5(c) builds the relationship between these phenomena. The increasing importance of time-to-market makes concurrent engineering much more favorable for most product development projects.

Proposition 3.5(d) indicates that a company does not need to adjust its overlapping policies when downstream development duration changes within $[D_u, +\infty)$. The change of development cycle time T is the same as the increasing or decreasing of D_d.

Corollary 3.1

(*a*) *For time-driven projects,* $c_t >> c_r$, *complete overlapping is optimal as long as rework is not more difficult than initial development.*

(*b*) *For cost-driven projects,* $c_r >> c_t$, *sequential process is optimal.*

Corollary 3.1 describes the optimal overlapping policies for time-driven and cost-driven problems. It gives the condition of when complete overlapping is optimal for time-to-market problem and explains why sequential process is widely used for cost driven projects.

3.4.2 Overlapping and Functional Interaction

Overlapping with or without functional interaction may lead to very different impacts on project performance. Without functional interaction, the reduction of project cycle time must be accompanied by additional cost. However, overlapping with functional interaction may reduce cost and cycle time simultaneously, which is supported by previous studies (e.g. Pisano, 1997; Ward et al., 1995; Bhuiyan et al., 2004). This is because efficient functional interaction can significantly reduce project uncertainty;

thus modification cost, rework duration, and rework cost will be reduced accordingly. Therefore choosing suitable overlapping and functional interaction policies, project cycle time and cost can be reduced simultaneously for the projects with low downstream sensitivity, fast upstream evolution, and efficient functional interaction. This situation is further illustrated with a numerical example shown in Figure 3.6. The shaded area shows the feasible policies which can reduce project cycle time and cost simultaneously. However, as discussed above, functional interaction also incurs cost. The question then is: how to optimize project performance by applying functional interaction and overlapping simultaneously? In this section we analyze optimal functional interaction strategies under the condition that optimal overlapping discussed above is followed.

Proposition 3.6

(*a*) *The optimal functional interaction duration locates in*

$$0 \le \tau^* \le \max\{0, \frac{1}{\lambda}\ln[\frac{kb\lambda(c_t + c_r)(D_u - t_e)}{c_t + c_\tau}]\} \tag{3.17}$$

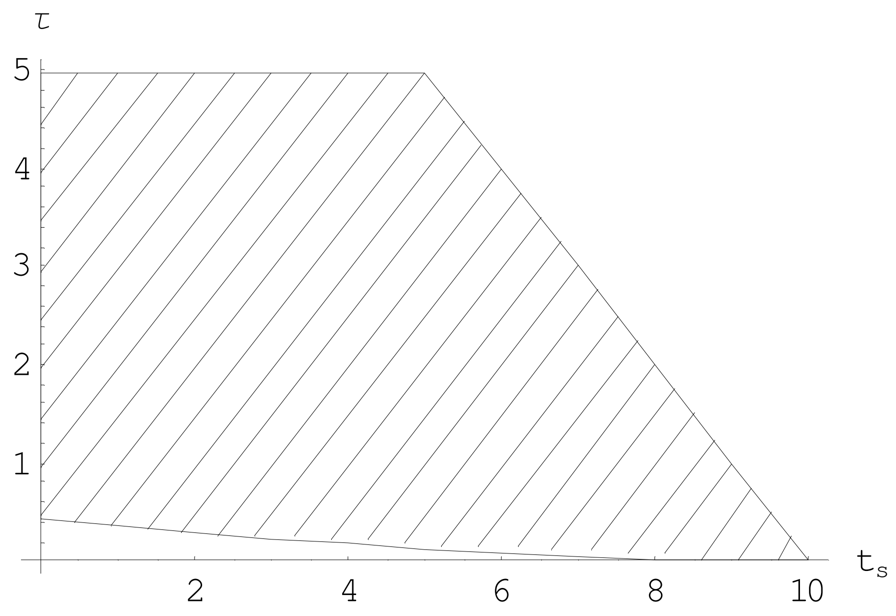
Furthermore $\tau^* = 0$ *if* $\frac{kb\lambda(c_t + c_r)(D_u - t_e)}{r} - (c_t + c_\tau) \le 0$ (3.18)

(*b*) If $1 - \frac{kb}{a+b}\int_{t_e}^{D_u} \mu_0(t)dt \ge 0$ (3.19)

G is concave with respect to τ. *Thus,* $\tau^* = 0$ *when* $\left.\frac{\partial G}{\partial \tau}\right|_{\tau=0} \le 0$.

(*c*) *If* $\lambda\tau_{\max}$ *is small,* G *is convex with respect to* τ. *Therefore*

$$G^* = \max(G|_{\tau_0, t_s^*}, G|_{\tau_{\max}, t_s^*}) \tag{3.20}$$



$D_u = D_d = 10$ weeks; $r = 1$; $k = 0.01$; $\theta = 0$; $c_r = 1$;

$c_\tau = 2$; $c_m = 0.1$; $\lambda = 1$; $a = 0$; $b = 100$ modifications

Figure 3.6 Reducing time and cost simultaneously

Proposition 3.6 shows that G can be convex or concave for certain cases, but overall it is indefinite. This result is consistent with that from Bhuiyan et al. (2004). Then where does this complexity come from? As described in section 3.2, the effect of functional interaction decreases with time. However every extra reduction of modifications will have a greater benefit on downstream rework than any previous reduction of the same magnitude. Take an extreme example. Suppose several modifications occur continuously at the time when two consecutive stages are almost finished. Every upstream modification will corrupt 50% of downstream work. Therefore 50%, 75%, and 87.5% of downstream work will be corrupted by one, two, and three modifications respectively. Reducing modifications from three to two is relatively easy but this only reduces 12.5% rework. On the contrary, reducing modifications from one to zero is difficult but it can decrease 50% rework.

Because of the complex relationship between functional interaction and project performance, optimal solutions are presented only for the projects satisfying certain conditions. Proposition 3.6(a) shows that no functional interaction should take place if project performance strictly decreases over τ . Proposition 3.6(b) shows the case where the reducing effect of functional interaction dominates the process which makes G concave with respect to τ. It indicates that either no functional interaction or the solution of the first order condition is optimal. Proposition 3.6(c) shows the case where the increasing importance of reducing a modification dominates the process. It leads to the convexity of G with respect to τ. Consequently the optimal solution must lie on a border.

For a project not satisfying the above conditions we can derive the solution based on the generalized conclusion shown in Proposition 3.6(a), i.e., the optimal functional interaction duration can not be bigger than a certain value. This is because the effect of functional interaction must be negative when τ is big enough. Further functional interaction will reduce uncertainty very little but increase functional interaction time significantly. Based on the range shown in Proposition 3.6(a) and the optimal overlapping policies discussed in the previous section we can obtain the graph about the relationship between functional interaction duration and project performance and thus identify the optimal solution. Figure 6 shows an example of a time-to-market problem. In this case, the graph is first convex and then changes to concave. The optimal functional interaction duration is about 5.68 weeks.

$D_u = D_d = 10$ *weeks;* $k = 0.1$; $r = 1$; $c_\tau = c_r = 0$; $c_t = 1$;
$\mu_\tau(t) = 10\exp\{-0.5\tau\}$; $t_s^* = 0$; $0 \le \tau^* \le 2\ln 50$

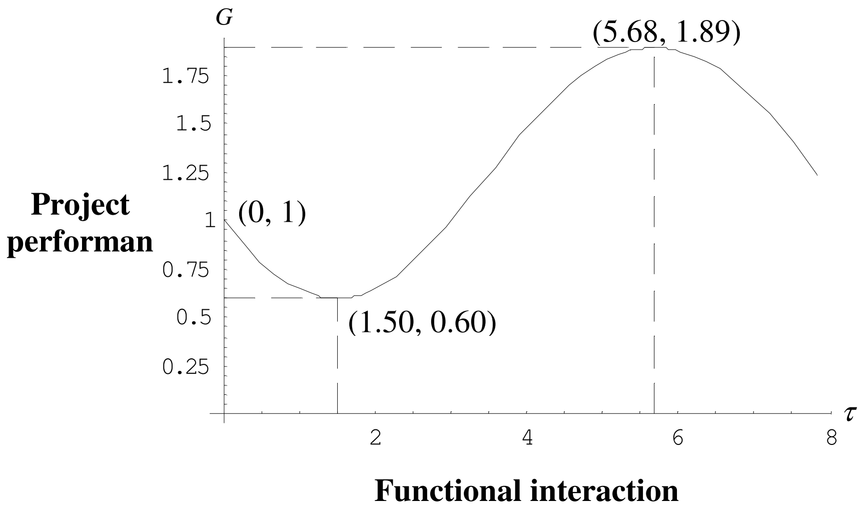


Figure 3.7 Functional interaction and project performance

The complex relationship between project performance and functional interaction implies that incremental improvement of functional interaction may be inapplicable. It may lead to local optimum. As shown in Figure 3.7, functional interaction lowers project performance at the beginning and thus $\tau = 0$ is a local optimum. The development cycle time based on local optimum is 0.89 week's longer than the result with global optimal solution.

Functional interaction has been found to increase (Eisenhardt and Tabrizi, 1995; Von Corswant and Tunälv, 2002), decrease (Bhuiyan et al., 2004; Wagner and Hoegl, 2006), or to have no significant effect (Datar et al., 1997) on the development cycle time. Proposition 3.6 explains these conflicting findings. Functional interaction is not a panacea for managing new product development. It is beneficial only if we can choose suitable functional interaction levels according to project properties such as evolution, dependency, rework rate, and functional interaction capacity.

3.5 Problem Variations

In the previous sections, we have developed a model for maximizing project profit, which we refer to as the *profit maximizing problem*. However, for some projects, Pareto-optimal overlapping strategies should be considered. These are strategies that generate a given cycle time at minimum cost, or conversely, strategies whose cycle times are shortest for a given budget, which we will refer to as the *target timing problem* and the *budget constraint problem*, respectively. Both of these problems can be solved by the similar approach as the profit maximizing problem.

Budget Constraint Problem

In this version of the problem, we need to determine the minimum cycle time under a budget constraint. We can formulate the problem as follows:

$$\min: T = D_d + \tau + t_s + \frac{R(t_s,\tau)}{r}$$

subject to: $C = c_\tau \tau + \dfrac{c_r R(t_s,\tau)}{r} \le B$

where B represents the budget for the additional costs incurred due to overlapping. By Proposition 3.2, it is easy to prove that the first derivative of the objective function with respect to t_s is positive $\forall t_s \in [t_e, D_u]$. That is, in order to reduce project cycle time, we need to start the downstream as early as possible. The first derivative of $c_\tau \tau + \dfrac{c_r R(t_s,\tau)}{r}$ is negative $\forall t_s \in [t_e, D_u]$. That is, the earliest downstream start time occurs at the point where $C = B$. Therefore the optimal start time t_s^* can be derived from the following equation:

$$c_\tau \tau + \frac{c_r R(t_s^*, \tau)}{r} = B$$

Target Timing Problem

For the second problem, we assume that the time to market is given, and the problem is to determine the optimal degree of overlapping such that the product will be launched at a specific target time with minimum development cost. Typical situations where this type of question arises are when new products are introduced during trade shows or when a competitor's launch date has to be preceded. This problem may be characterized as the "dual" of the previous problem. Target timing has commonly been used as the strategy of new product introduction. Given the time to market as a target, the objective is to minimize product development cost, which can be formulated as follows:

$$\min : C = c_\tau \tau + \frac{c_r R(t_s, \tau)}{r}$$

subject to: $T = D_d + \tau + t_s + \dfrac{R(t_s, \tau)}{r} \le T_t$

where T_t represents the target timing of the project. We have shown that C is decreasing with t_s and T is increasing with t_s. Therefore, for target timing problem, the cost can be minimized when $T = T_t$. The optimal start time t_s^* can be derived from the following equation:

$$D_d + \tau + t_s + \frac{R(t_s^*, \tau)}{r} = T_t$$

3.6 Model Application

Our model was applied to improve the mobile phone development process at a leading handset application software and handset solution provider in China. This company employs approximately 2,600 professionals, of which 90% are engineers. As of July 2006, the company has developed more than 100 handset designs for international brand owners such as NEC, Kyocera, and Mitsubishi.

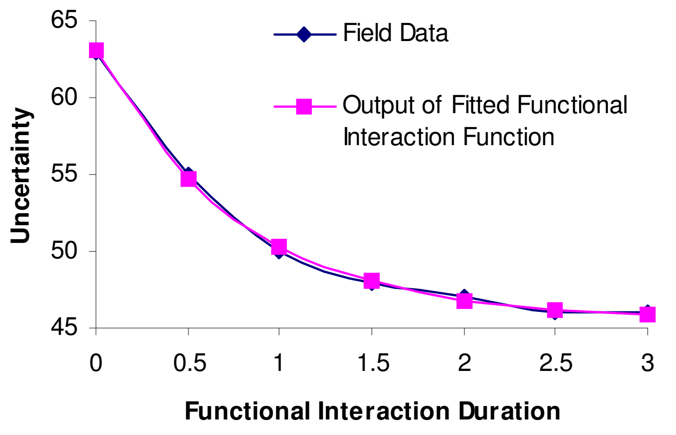
As is common in consumer electronics industry, shorter product life cycles impel the company to develop new products at an increasingly rapid pace. Hence, all the major precepts of concurrent engineering (including overlapping and functional interaction) are practiced in the company. However, these policies were intuitively determined by project managers, rather than on analytical grounds, tending to yield inefficient strategies. We aim to improve the development process by providing analytical solutions about overlapping and functional interaction. Derivative projects with mature architecture, which account for about 70% of the projects completed during the 2004-2006 period, were studied. We focused specifically on the detail design (upstream) and mold fabrication (downstream) since they are within the critical path of mobile phone development and are the primary drivers of development cycle time. Mold fabrication is done by suppliers with whom the company has developed close partnership over the years. The information transformation between these companies, the updating of 3D and 2D drawings, is relatively inexpensive.

Most of the parameters were directly derived from the five mobile phone development projects completed in the first half of 2006. To improve the reliability of the collected information, these data were checked and confirmed by the engineers who were familiar with these projects. For example, the information evolution duration was extremely long for project "VX" because of the customer's suspension of the project. Consequently the evolution duration for this project was derived by deducing the abnormal delay. We now describe our data collection efforts at the company in detail.

We first collected the data about functional interaction and upstream evolution. Functional interaction represents the involvement of tooling engineers (from supplier) during the detail design stage. We observed that functional interaction duration was short in most projects (e.g. half day or one day), resulting in insufficient data for us to derive the relationship between design uncertainty and functional interaction duration. Hence, we asked the experienced project engineers to estimate the uncertainty level when longer functional interaction is applied and then conducted a regression analysis by minimizing SSE (sum of squared residuals) to determine the functional interaction function.

Upstream evolution is the nonhomogeneous Poisson process of engineering changes. After finishing the preliminary design, the resulting design drafts are continuously modified. The modification usually lasts for 12 working days. The number and time of design modifications are well documented in the company and thus the evolution function can be derived from field data. The outputs of regression analysis of these two functions are summarized in Figure 3.8.

$$a+b\exp\{-\lambda\tau\}=45.50+17.57\exp\{-1.29\tau\}$$



$$\mu_1(t)=11.93\exp\{-0.23t\}$$

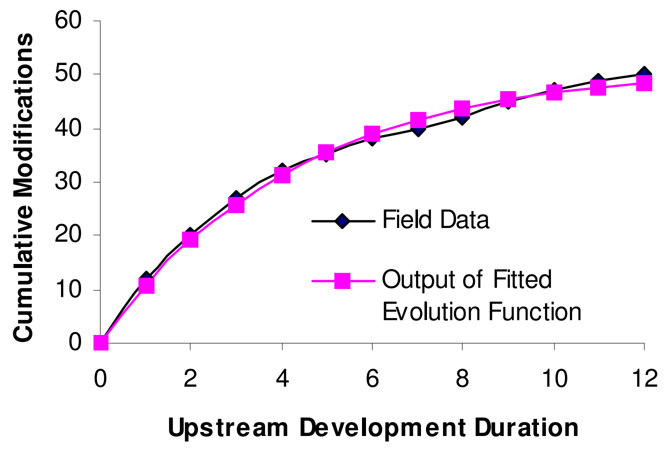


Figure 3.8 Evolution and functional interaction functions

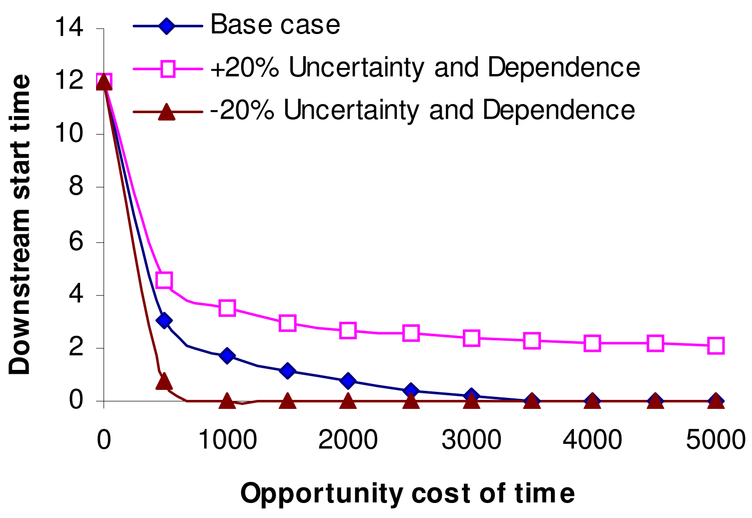
After that, we operationalized the estimation of the dependency parameter k . As previous literature (Terwiesch and Loch, 1999; Roemer and Ahmadi, 2004; Carrascosa et al., 1998) and our field study show, the impact of an upstream modification on downstream rework is the result of the dependency of overlapped stages multiplied by the progress of the downstream stage. Vice versa, if the total amount of rework, the number of modifications, and the progress when each modification occurred are available, we can derive the dependency directly from historical data. Mathematically:

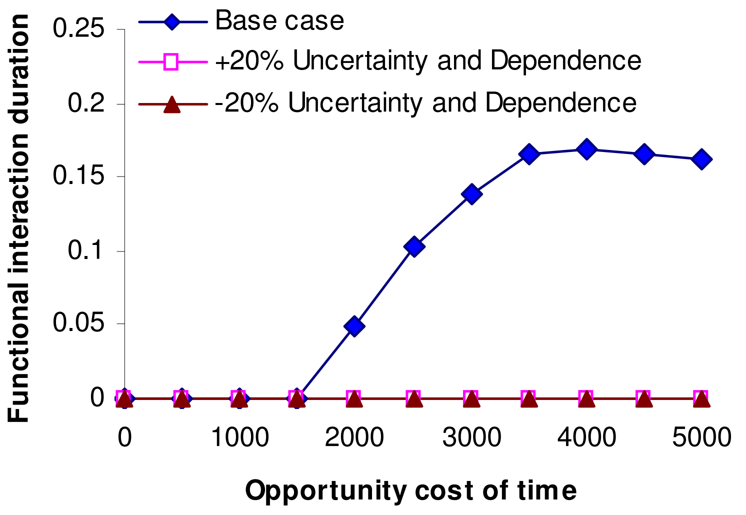
$$k = R / \sum_{i=1}^{N} P_i \tag{3.21}$$

where N denotes the overall number of modifications occurred in the detail design stage; P_i is the progress of the downstream stage when the i^{th} modification occurred; R is the amount of downstream work corrupted by these modifications. In our data set, the value of dependency k is 1.9%.

The estimates of the development duration and rework rate were much simpler because both of them can be directly derived from previous projects (Ford and Sterman, 1998; Repenning, 2001). The costs of functional interaction and rework are largely dominated by the engineering hours spent on the activities. Therefore these costs were simply determined by the average hourly wage of the engineers and the related engineering hours. However, the opportunity cost of time changed significantly for different projects (It was less than \$5000 in our data set).

After deriving these parameters, we applied the methodology developed in this chapter and supplied management with the functional interaction and overlapping strategies for the projects with different opportunity cost of time. The effect of inaccurate estimates of uncertainty and dependency was evaluated in Figure 3.9.





$\mu_\tau(t) = (10.78 + 4.16\exp\{-1.29\tau\})\exp\{-0.23t\}$; $D_u = 12$;
$D_d = 25$; $k = 1.9\%$; $r = 1$; $c_\tau = 200\$$; $c_r = 350\$$

Figure 3.9 Optimal policies for the projects with different opportunity cost

Although the results may not be very accurate because of estimation errors of the parameters, they are good enough to guide the management of similar development projects in the company. Previously, low overlapping degree and short term functional interaction were applied for all the projects in the company. Figure 3.9 shows that the applied functional interaction policy is consistent with our analytical results. The designers, usually with 5 or more similar project experiences, know very well of the downstream stage; thus the benefit of functional interaction is small. Note, as Figure 3.9 shows, the relationship between uncertainty and functional interaction is nonlinear. The increasing of uncertainty will increase the number of modifications reducible through functional interaction, but it will also reduce the overlapping degree and thus reduce the impact of each modification on downstream rework. The optimal functional interaction level may decrease when both uncertainty and dependency are increased.

Previous overlapping policy applied in the company shows the dilemma faced by project managers. Short product life cycles impel engineers to reduce cycle time by overlapping development stages but the limited development budget tells them that overlapping should be avoided since it is costly. Previous overlapping policy made by intuition is far from optimal. Our study clearly shows when the benefit of overlapping overwhelms the cost of rework and when complete overlapping should be applied (Figure 3.9). High level of overlapping is more favorable when opportunity cost of time is big ($c_t > 1000$), regardless of the change of other parameters (such as uncertainty and dependency).

3.7 Discussion and Conclusion

Literature (e.g. Terwiesch and Loch, 1999; Roemer and Ahmadi, 2004) shows that companies are still determining overlapping and functional interaction on an ad hoc basis, yielding inefficient development policies. There is a need to quantitatively analyze development strategies on project properties. The model presented in this chapter helps to identify optimal functional interaction and overlapping policies to improve project performance. The interactions between overlapping and functional interaction as well as the trade-offs between cost and cycle time are explicitly studied. We use a general assumption of the upstream evolution, which ensures the model to be a good abstraction of practice. Moreover, the impact of upstream modifications on downstream rework is explicitly studied, which ensures the accuracy and reliability of the conclusions derived.

This research yields three main results. Firstly, our analytical results show that the change of downstream progress over time is quite complex. It may be increasing or decreasing for the projects with different uncertainty, dependency, and evolution features. The assumption of increasing downstream progress is only applicable for the projects with fast or linear evolution. For future models of overlapped process, it is necessary to address the downstream progress in detail since it is one of the key factors affecting overlapping policies.

Secondly, we derive the optimal overlapping policies for the projects with different properties. Two conditions for complete overlapping are given in Proposition 3.4. One is based on detailed information of evolution and dependency. It defines the necessary and sufficient condition for complete overlapping. The other is the sufficient condition for complete overlapping which only relies on the information about rework rate, opportunity cost of time, and rework cost. Proposition 3.5 and Corollary 3.1 show the relationship between project properties and overlapping policies. Higher uncertainty and dependency reduce the optimal overlapping which agrees with previous literature (e.g. Eisenhardt and Tabrizi, 1995; Loch and Terwiesch, 1998). The downstream rework rate, opportunity cost of time, and rework cost also affect overlapping degree. Higher rework rate and opportunity cost of time make overlapping more active. However rework cost is negatively related to the overlapping degree and thus sequential process is more attractive for cost-driven projects.

Thirdly, we derive the optimal functional interaction strategies when overlapping follows the optimal pattern. The relationship between project performance and functional interaction is complex. It may not simply increase or decrease with time. Incremental improvement of functional interaction may not work for consecutive projects, which increases the need of formal models for decision making.

There are several limitations for the application of our results. Firstly, we assume that the information transformation to the downstream stage is relatively inexpensive because of the advancement of information technology and the close cooperation between functional teams. Previous research (e.g. Krishnan et al., 1997; Helms, 2004; Roemer and Ahmadi, 2004) supports this argument. However some projects may have information transformation difficulty. In the next chapter, we discuss the optimal communication and overlapping policies for these projects. Secondly the model in this chapter requires extensive knowledge of evolution and dependency. For incremental innovations which account for the majority of product development projects (Whitney, 1990; Wang and Yan, 2005), these data are typically available (Krishnan et al., 1997; Roemer et al., 2000; Helms, 2002), as demonstrated in our in-depth case study at the handset design company. For completely new projects this information is often not accessible and optimal solutions suggested here may not be achievable. Finally, product quality and product originality, which are not concerned in this paper, may play important roles in the success of radical projects. Consequently, if managers intend to apply our model to radical projects, it is important to make sure that: (1) cost and time are the most important factors for these projects; (2) the data about evolution and dependency are available or estimable.

Several aspects of the model presented in this chapter merit further examination. Firstly, we only focus on time-cost trade-off in this chapter. The trade-off between time and quality or cost and quality deserves further attention. Secondly, extension of the model to multiple development stages should be further investigated. Thirdly, we may extend our model by including the detection of upstream errors by the downstream stage. Lin et al. (2008) studied it with a System Dynamics model (the model shown in Chapter 5). Because of the limitation of System Dynamics approach, no optimal solution is derived in that paper. Further investigation with an analytical model may derive better solutions and useful management insights.

CHAPTER 4

MANAGING CONCURRENT DEVELOPMENT PROCESSES WITH HIGH COMMUNICATION COST

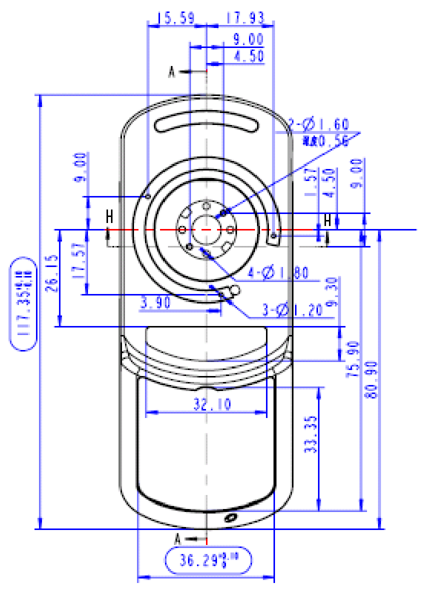
An analytical model for managing sequentially dependent stages with low communication cost is described in Chapter 3. However, the results and methodology derived in that chapter cannot be applied for the projects with high communication cost, since communication policy will significantly influence project performance when communication takes time or cost. In this chapter we first introduce an algorithm to derive the optimal overlapping and communication strategies for the projects with high communication cost. Then the projects with linear upstream evolution are discussed and a more efficient algorithm is proposed. Furthermore, some guidelines for structuring concurrent processes are provided. We applied this methodology to study the development projects in a large consumer electronics company.

4.1 Introduction

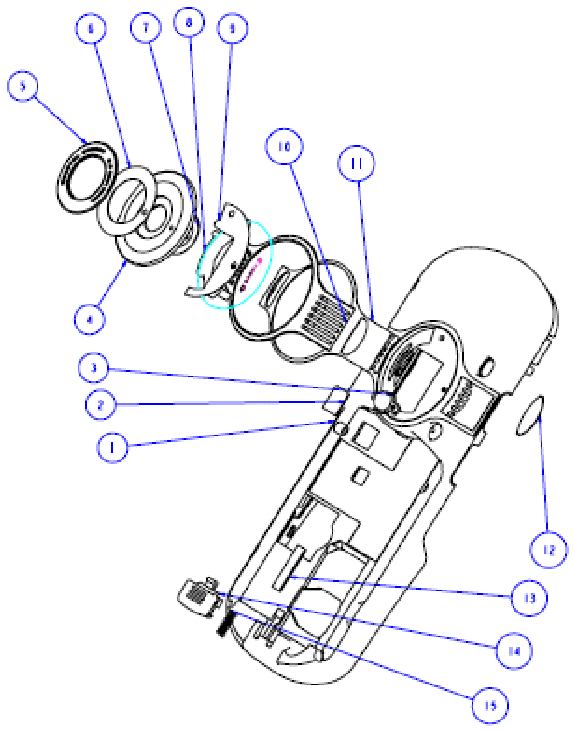
Shorter product life cycles in many industries impel firms to develop new products at an increasingly rapid pace. Overlapping development stages, combined with frequent information exchange, is commonly regarded as a core technique for faster product development (Clark and Fujimoto, 1991; Wheelwright and Clark, 1992; Smith and Reinertsen, 1998; Joglekar and Ford, 2005; Minderhoud and Fraser, 2005).

For projects with low communication cost, studies have shown that frequent information exchange is optimal. However, for projects with high communication cost, the trade-offs need to be studied. Frequent information exchange between the development teams reduces the negative effect of overlapping but increases communication time and development cost (Haberle et al., 2000). To improve project performance, overlapping and communication policies should be coordinated and the time-cost trade-offs involved in concurrent product development should be investigated.

Here we give a simple example to further illustrate the problem studied in this chapter. In mobile phone development, mold fabrication (such as the slots shown in Figure 4.1) is done according to the outputs of detail design (such as the dimensions shown in Figure 4.1). Initial design usually takes two and a half weeks, and then the design will be continuously modified until the end of the fifth week. In order to reduce development cycle time, mold fabrication often starts before all of the design errors have been identified and resolved. Then, what is the optimal start time of mold fabrication? If overlapping is applied, should we arrange a series of meetings in order to frequently inform the downstream team of the latest engineering changes? How many meetings should be arranged and when should the meetings take place? This chapter presents an analytical model addressing these questions which are commonly faced by development engineers.



Detail design



Components

Figure 4.1 Mobile phone development

This chapter is organized as follows. The next section reviews the relevant literature on concurrent engineering. In section 4.3, we provide an analytical model which captures the relationship between project properties, overlapping degree, communication policy, and project profit. Section 4.4 introduces an algorithm for optimizing overlapping and communication strategies. Then, the projects with linear upstream evolution are studied and some guidelines for structuring concurrent processes are provided. In section 4.5, we illustrate the methodology with a case study at an electronics company. Conclusions are summarized in the last section.

4.2 Related Literature

The interaction between overlapping and communication strategies was first emphasized by Clark and Fujimoto (1991). Their studies of the world automobile industry showed that, for effective operation of overlapping, problem solving must be integrated with frequent information exchange. This concept is useful to initiate organizational changes and has had a strong influence on the research and practice in concurrent engineering. However, to fully operationalize the concept, we need to better understand the interaction between overlapping and communication strategies, and derive the appropriate development policies depending on project characteristics.

An interesting body of mathematical models has been developed to address the trade-offs involved in concurrent product development. We sorted them into two groups. In the first group, researchers assume that the information exchange between development teams is costless (e.g. Krishnan et al., 1997; Roemer et al., 2000; Chakravarty, 2001) and thus it can take place immediately after an engineering change occurs. According to this assumption, the concurrency problem is simplified into an overlapping problem which is irrelevant to the communication strategies. There are several notable studies in this group. Krishnan *et al.* (1997) developed a model to determine the number of downstream iterations so as to minimize the development cycle time. They showed that "upstream evolution" and "downstream sensitivity" are the two properties determining the optimal overlapping strategies. Their model was applied in the development of a door panel of a car. Roemer et al. (2000) studied the time-cost trade-offs in multistage overlapped processes. An efficient algorithm was proposed to determine an appropriate overlapping strategy. Chakravarty (2001) sorted overlapping into three major modes: *interrupt build overlapping*, *continuous build* *overlapping*, and *preempt build overlapping*. The optimality conditions for each mode are presented and the relationship between different overlapping modes is discussed. Roemer and Ahmadi (2004) studied the interaction of two common tools for reducing project cycle time: overlapping of development stages and crashing of development times. This line of research greatly advances the analysis of concurrent product development. However, the results and methodology derived cannot be applied for projects with high communication cost, because, for these projects, communication policy interacts with overlapping strategy and substantially affects project performance.

It is well known that information exchange between development teams usually takes time and cost (Loch and Terwiesch, 1998). Helms (2002) reported a typical information exchange procedure in a chemical company. If design changes are needed, the upstream team needs to submit a Change Notice to the downstream members. A meeting is then arranged where the engineers from the two teams discuss the reasons for the changes, the cost to process the changes, and so on. This will finally results in the release of a new version of drawings and documents. Because the information exchange process is time consuming, the engineers in the company tend to have fewer meetings and information exchanges in order to concentrate on their development work. However, the managers asked for more frequent information exchanges in order to reduce downstream rework. The questions then are what is the optimal timing and frequency of information exchange, and how communication policy interacts with overlapping degree.

The second group of studies takes into account the cost of communication and tries to address these questions. Ha and Porteus (1995) studied the benefit of early detection of upstream flaws through overlapping and frequent communication. If information exchange is too frequent, then communication time and cost would increase significantly. However, infrequent information exchange would delay the identification of the design flaws and increase the corresponding rework of the upstream stage. While their research emphasizes the downstream team's ability to discover the faults in the upstream stage, our study focuses on the influence of the upstream modifications on the downstream rework, or "integrated problem solving" (Clark and Fujimoto 1991).

Our work is most closely related to that of Loch and Terwiesch (1998). In their model, the overlapping and communication policies are determined by modification rate, downstream sensitivity, and communication cost. Modification rate represents the arrival rate of upstream changes, corresponding to the concept of "upstream evolution" in Krishnan et al. (1997) work. Downstream sensitivity is defined as the impact of a modification on downstream rework. Communication cost denotes the fixed set-up time for information exchange. Based on these concepts, the authors built an analytical model to minimize project cycle time. We improved upon this work in three important aspects:

Firstly, the impact of downstream progress on rework is explicitly addressed in our model. It is well known that the impact of an upstream modification on downstream rework is closely related to the progress of the downstream stage, the amount of work that is completed and not corrupted by upstream changes (e.g. Terwiesch and Loch, 1999; Carrascosa et al., 1998; Thomke and Bell, 2001, Roemer and Ahmadi, 2004; Lin et al., 2008). However the question of how to model the dynamic change of downstream progress is not well addressed by previous studies. Loch & Terwiesch (1998) implicitly assume that the downstream progress equals the downstream development time t. However, in concurrent processes, the downstream progress at time t should be less than t due to the existence of uncertainty and corruption (Carrascosa et al., 1998). Furthermore, if the project uncertainty and dependency are high, their assumption may lead to negative downstream progress which contradicts with the fact that the downstream process should be always greater or equal to zero. For example, assume the dependency of two development stages is 0.5. Then, by Loch and Terwiesch's assumption, each modification at the 10th day will corrupt 5 day's downstream work. The downstream progress at the 10th day becomes negative if more than two modifications occur at that day. Therefore, it is valuable to reinvestigate the overlapping and communication policies by describing the progress of downstream stage in detail.

Secondly, we extend Loch and Terwiesch's (1998) research by relaxing their assumption of modification rate. Literature shows that the modification rate may be linear, nonlinear, concave, convex, or S-shaped with respect to development time (Eppinger et al. 1994, Krishnan et al. 1997, Chakravarty 2001). However, Loch and Terwiesch (1998) only studied projects where the modification rate is linearly related to development time. In our research, we examine projects with non-linear modification rate.

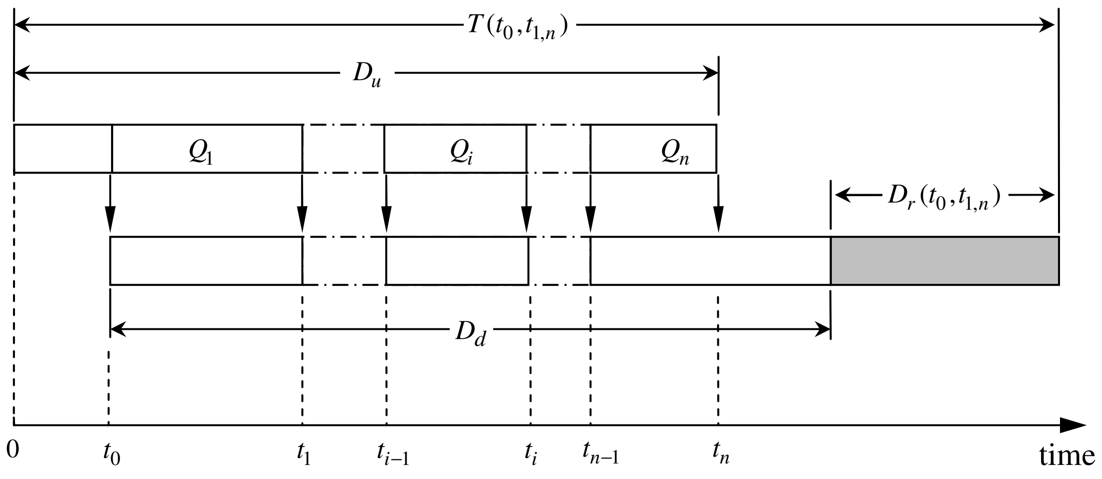
Thirdly, this work explicitly addresses the trade-off between increased cost and reduced project cycle time due to concurrent execution. Project performance reflects integration of three intermediate objectives: time, cost, and quality. For most derivative projects, the target quality is well defined. Therefore, the objective for management is to achieve the required quality with less time and cost (Gerwin and Barrowman 2002). However, as with most of the studies in this area, Loch and Terwiesch used development cycle time as the sole measure for project performance. As a result, they disregarded the fact that hidden costs are usually accompanied with the practices that reduce time-to-market (Smith and Reinertsen 1998). We explicitly address development cost in our model which is mainly the labor cost of information exchange and rework. We measure project performance in profit which is defined as gain from early launch of the product minus additional project cost.

4.3 Model Formulation

In this section we first formulate the time-cost trade-off involved in concurrent product development. Then we introduce a general mathematical model of the relationships between project properties, development policies, and project performance. The objective function, constraint equations, model parameters, and decision variables are summarized at the end of this section.

4.3.1 Overview of the Model

Our model examines the projects with a “nominal sequential” (Krishnan et al., 1997) structure, i.e. the principal information exchange between consecutive design stages is unidirectional: from upstream stage to downstream stage. For each time of communication, both the upstream and downstream engineers must drop everything they are doing and commit themselves entirely to set a meeting and discuss the latest changes for downstream incorporation (Ha and Porteus, 1995; Loch and Terwiesch, 1998; Wang and Yan, 2005).



→	Information exchange	$T(t_0,t_{1,n})$	Development cycle time
0	The time when preliminary information is available		**Decision variables**
D_u	Upstream development duration	t_0	Start time of downstream stage
D_d	Downstream development duration	Q_i	The time interval between the i^{th} and $(i-1)^{th}$ information exchange
$D_r(t_0,t_{1,n})$	Downstream rework duration, where $t_{1,n}$ denotes the information exchange policy, i.e. $(t_1,\cdots,t_i,\cdots,t_n)$.	t_i	The time of the i^{th} information exchange

Figure 4.2 Overlapped process with multiple information exchanges

Consider the concurrent execution of two consecutive development stages (Figure 4.2). We call the first stage upstream and the second downstream. D_u and D_d represent the durations of initial development of upstream and downstream developments respectively, where the duration of initial development is the expected time it takes to perform a stage without overlapping. Without loss of generality, we assume preliminary information of the upstream is available at time zero. t_e denotes the earliest start time of the downstream stage, which ensures that the upstream is finished earlier than the downstream. Then, the start time of the downstream stage t_0 should be

$$t_e \le t_0 \le D_u ; \tag{4.1}$$

$$t_e = \max\{0, D_u - D_d\} . \tag{4.2}$$

During the overlapped process, there may be several "bursts" of information transfer from the upstream stage to the downstream stage. Frequent information exchange allows the downstream engineers to be aware of the latest upstream change, and thus reduces the amount of rework.

A typical overlapped process with multiple information exchanges is shown in Figure 4.2. The information exchange policy can be represented in the form ($Q_1, Q_2, \cdots, Q_i, \cdots, Q_n$), where Q_i is the interval between the $(i-1)^{th}$ and i^{th} information exchange. The time for the i^{th} information exchange is denoted as t_i , so that $t_i = t_0 + \sum_{j=1}^{i} Q_j$. Therefore, the information exchange policy can also be represented in the form ($t_1, t_2, \cdots, t_i, \cdots, t_n$). Note that, in this model, the i^{th} information exchange denotes the i^{th} meeting since the start of downstream development.

If sequential process is applied, the development cycle time should be $D_u + D_d + \beta$, where β represents the setup time for information exchange which is closely related to Ha and Porteus (1995) and Loch and Terwiesch's (1998) definitions of setup/penalty time and communication cost respectively. For concurrent process, the development cycle time can be written as

$$T(t_0, t_{1,n}) = t_0 + D_d + (n+1)\beta + D_r(t_0, t_{1,n}) . \tag{4.3}$$

where $t_0 + D_d$ represents the time for regular development; n denotes the number of information exchanges after the start of downstream development; $D_r(t_0, t_{1,n})$ is the duration for rework; $t_{1,n}$ denotes the information exchange policy, i.e. $(t_1, t_2, \cdots, t_i, \cdots, t_n)$.

We assume the opportunity cost of time c_t is constant as, for most development projects, the opportunity cost of time does not change significantly in short run. That is, although the opportunity cost of time may be different in years, but it is almost constant in several weeks. Furthermore, in practice, the penalty for delay and the premium for early completion are usually proportional to time. Therefore, for concurrent development, the gain from reducing development cycle time can be written as

$$c_t[D_u + D_d + \beta - T(t_0, t_{1,n})]. \tag{4.4}$$

We next consider the costs associated with rework and communication. Let c_β be the cost for information exchange (cost per meeting) and c_r denote the cost for one unit time of rework. The additional cost for overlapped development is

$$C(t_0, t_{1,n}) = n \cdot c_\beta + c_r \cdot D_r(t_0, t_{1,n}). \tag{4.5}$$

Thus, profit maximization objective can be represented as

$$\max : G = c_t[D_u + D_d + \beta - T(t_0, t_{1,n})] - C(t_0, t_{1,n}). \tag{4.6}$$

4.3.2 Downstream Rework

At the heart of overlapping problem lies the downstream rework duration $D_r(t_0, t_{1,n})$ which impacts project cycle time and increases development cost. The downstream rework duration is determined by the evolution of upstream modifications, the dependency between development stages, and the downstream progress (Krishnan et al. 1997, Carrascosa et al. 1998, Loch and Terwiesch 1998, Lin et al. 2008). Previous studies (e.g. Eppinger et al., 1994; Krishnan et al., 1997; Chakravarty, 2001) show that upstream evolution can be concave, convex, or S-shaped. Therefore, we model the arrival of upstream modifications (upstream evolution) as a nonhomogeneous Poisson process with non-negative rate $\mu(t)$. This assumption is quite general and includes all the evolution patterns identified by previous studies.

Previous research (e.g. Carrascosa et al., 1998; Loch and Terwiesch, 1998; Lin et al., 2008) has made it clear that, for most development projects, the impact of a modification on downstream rework is in proportion to the downstream progress $P(t)$. Let k_1 be the percentage of downstream progress which will be affected by a modification. Then, if a modification arises at time t the progress will be changed into $P(t)\cdot(1-k_1)$. If the second modification occurs at the same time, then the progress will be further reduced into $P(t)\cdot(1-k_1)^2$. For modeling simplicity, we define k as the dependency parameter and it equals $-\ln(1-k_1)$. Then the progress for one and two modifications can be represented as $P(t)\cdot\exp\{-k\}$ and $P(t)\cdot\exp\{-2k\}$ respectively. Figure 4.3 graphically shows the change of downstream progress over time. At time t_1 the downstream group is informed about the modifications identified from t_s to t_1. To simplify our model, we approximate the downstream progress by calculating it on the mean of the modifications. Consequently, at time t_1, the progress changes from Q_1 to $Q_1\exp\{-k\int_{t_0}^{t_1}\mu(t)dt\}$. At time t_2 the progress changes from $Q_1\exp\{-k\int_{t_0}^{t_1}\mu(t)dt\}+Q_2$ into

$[Q_1 \exp\{-k\int_{t_0}^{t_1}\mu(t)dt\}+Q_2]\cdot\exp\{-k\int_{t_1}^{t_2}\mu(t)dt\}$, and so on. It is clear that the progress rate of the downstream stage is dynamically affected by upstream modifications, dependency of overlapped stages, overlapping strategies, and communication policies. The expected rework for a concurrent process can be written as

$$D_r(t_0,t_{1,n}) =$$

$$\frac{1}{r}(\sum_{i=1}^{n}Q_i-((Q_1\exp\{-k\int_{t_0}^{t_1}\mu(t)dt\}+Q_2)\exp\{-k\int_{t_1}^{t_2}\mu(t)dt\}+\cdots+Q_n)\exp\{-k\int_{t_{n-1}}^{t_n}\mu(t)dt\}),$$

which can be simplified to

$$D_r(t_0,t_{1,n})=\frac{1}{r}\sum_{i=1}^{n}Q_i(1-\exp\{-k\int_{t_{i-1}}^{D_u}\mu(t)dt\}). \tag{4.7}$$

where parameter r denotes the rework rate.

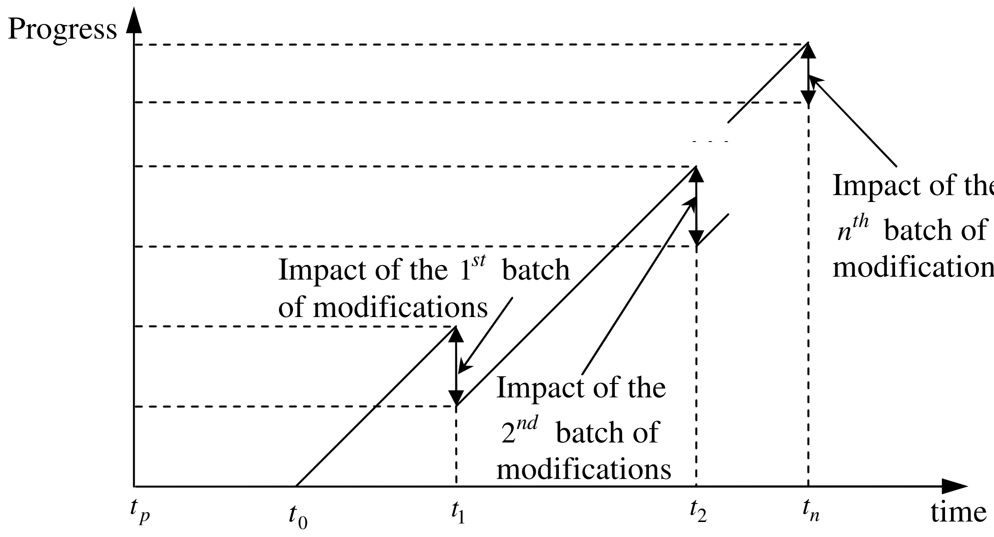

Progress
Impact of the 1^{st} batch of modifications
Impact of the 2^{nd} batch of modifications
Impact of the n^{th} batch of modifications
t_p
t_0
t_1
t_2
t_n
time

Figure 4.3 Progress of downstream stage

To incorporate all the upstream modifications, the last meeting should be arranged at time D_u (i.e. $t_n = D_u$). k_1 should be less than one, since the downstream progress is non-negative. Mathematically

$$\sum_{i=1}^{n} Q_i = D_u - t_0 , \tag{4.8}$$

$$0 < k_1 < 1 . \tag{4.9}$$

4.3.3 Summary

Table 4.1 Inputs and decision variables

Parameters	**Definition**
D_u	Upstream development duration
D_d	Downstream development duration
$\mu(t)$	Rate of upstream modifications
k_1	The percentage of downstream progress which will be affected by a modification
k	Dependency between development stages
r	Rework rate of downstream stage
β	Communication duration (per meeting)
c_β	Communication cost (per meeting)
c_t	Opportunity cost of time (the cost of increasing one unit of time-to-market)
c_r	Rework cost (the cost for one unit time of rework)
Decision Variables	**Definition**
t_0	Start time of downstream stage
n	The number of meetings
Q_i	The time interval between the $(i-1)^{th}$ and i^{th} information exchange
t_i	The time of the i^{th} information exchange

We summarize all the variables in Table 1 and state the optimization problem as follows

$$\max : G = c_t(D_u - t_0 - n\beta) - nc_\beta - \frac{c_t + c_r}{r}\sum_{i=1}^{n} Q_i[1 - \exp\{-k\int_{t_{i-1}}^{D_u} \mu(t)dt\}]\,; \qquad (4.10)$$

subject to: Equations (4.1), (4.2), (4.8), (4.9);

$$k = -\ln(1 - k_1)\,; \qquad (4.11)$$

$$t_i = t_0 + \sum\nolimits_{j=1}^{i} Q_j\,. \qquad (4.12)$$

The decision variables of this model are t_0, n, and Q_i ($1 \le i \le n$): n and Q_i determine the communication policy; t_0 represents the overlapping policy (smaller t_0 corresponds to higher overlapping degree).

4.4 Analysis of Overlapping and Communication Policies

For concurrent process, project performance is determined by overlapping and communication strategies. In this section, we first derive the optimal communication policy assuming t_0 is given. Then we address the optimal overlapping degree and introduce an algorithm to improve project performance by adjusting overlapping and communication strategies simultaneously.

4.4.1 Information Exchange Policy

Assuming the downstream start time is fixed, we seek a contingent information exchange strategy that maximizes project profit. The results are summarized in Proposition 4.1.

Proposition 4.1

(*a*) *Given n and* t_0*, all the stationary points can be determined using the following equations*

$$Q_i = \frac{1-\exp\{-k\int_{t_{i-2}}^{t_{i-1}} \mu(t)dt\}}{k\mu(t_{i-1})} \qquad 2 \le i \le n, \tag{4.13}$$

$$\sum_{i=1}^{n} Q_i = D_u - t_0 .$$

By comparing the stationary points, we can identify the optimal information exchange policy. Inspection of (4.13) shows that Q_i ($2 \le i \le n$) *is determined by* Q_1*. Therefore, the current problem can be simplified into a problem of identifying the optimal* Q_1.

(*b*) *Given* t_0*, the optimal communication times* n^* *must satisfy the following equation*

$$n^* < 1 + \frac{c_t(D_u - t_0)}{c_t\beta + c_\beta} \tag{4.14}$$

Proof. To enhance readability of the thesis, all proofs of this chapter are shown in Appendix B.

By Proposition 4.1(a), the optimal communication policy can be derived by one dimensional search when t_0 and n are predetermined (Q_1 changes from 0 to $D_u - t_0$). The range of n^* is given in proposition 4.1(b). $c_t(D_u - t_0)/(c_t\beta + c_\beta)$ is small since we only study the development projects with high communication cost (for the projects with low communication cost, a model has been developed in Chapter 3). Therefore, the optimal communication policy can be derived through several iterations of one dimensional search when t_0 is predetermined. These deliberations give rise to the following algorithm.

Algorithm 1

Initiation:

Step 0. Setting $n=1$ and $Q_1 = D_u - t_0$, compute $G^*(t_0,n)$. Let $n^*=1$, $Q_1^* = D_u - t_0$, $G^*(t_0) = G^*(t_0,n)$, and $n=2$. Here $G^*(t_0,n)$ represents the optimal performance when t_0 and n are given. $G^*(t_0)$ is the optimal performance when t_0 is predetermined.

Iteration:

Step 1. If $n \geq 1 + c_t(D_u - t_0)/(c_t\beta + c_\beta)$, stop.

Step 2. Search Q_1 in the range $0 < Q_1 < D_u - t_0$ and identify all the values which satisfy $\sum_{i=1}^{n} Q_i = D_u - t_0$. By the method of Lagrange multipliers, these are local optima.

Step 3. Compare the local optima and identify the Q_1 which yields the maximum profit $G^*(t_0,n)$.

Step 4. If $G^*(t_0,n) > G^*(t_0)$, then $n^* = n$, $Q_1^* = Q_1$, and $G^*(t_0) = G^*(t_0,n)$.

Step 5. $n = n+1$. Go to *Step 1.*

If $n=1$, Q_1 must be $D_u - t_0$ and $G^*(t_0,n)$ can be derived directly. In the initialization of the algorithm, we first let $n^*=1$, $Q_1^* = D_u - t_0$, and $G^*(t_0) = G^*(t_0,n)$. Then, in step 1 and 2, we change the value of n and check the optimal project performance when t_0 and n are given. Step 3 compares $G^*(t_0,n)$ with $G^*(t_0)$. If $G^*(t_0,n)$ is better, the communication policy will be changed accordingly. The optimal communication policy can be identified after all the possible values of n are examined.

In Algorithm 1, we first try to identify all the stationary points when n and t_0 are given. Then, by comparing the stationary points, we can identify the optimal information exchange policy when n and t_0 are given. By trying all the possible values of n, we can identify the optimal information exchange policy when overlapping degree is predetermined.

4.4.2 Optimizing the Concurrency

We have derived an algorithm to determine the optimal information exchange policy when t_0 is given. Then, how to identify the optimal overlapping degree? This problem can be solved with the analytical results presented in this section.

Proposition 4.2

Assume t^*_{1,n^*} (i.e. $t^*_1, t^*_2, \cdots, t^*_i, \cdots, t^*_{n^*}$) is the optimal information exchange policy when t_0 is given. Then, if the downstream starts at t^*_i the optimal information exchange policy should be $t^*_{i+1}, \cdots, t^*_{n^*}$.

Proposition 4.3

Assume the optimal communication policy is $\hat{t}_{1,\hat{n}}$ (i.e. $\hat{t}_1, \hat{t}_2, \cdots, \hat{t}_i, \cdots, \hat{t}_{\hat{n}}$) when complete overlapping is applied (i.e. $t_0 = t_e$).

(a) If $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_{\hat{n}-1}}^{D_u} \mu(x)dx\}] > 0$, then there is a smallest $\hat{t}_i$ satisfying $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_i}^{D_u} \mu(x)dx\}] > 0$ and the optimal downstream start time locates in $[\hat{t}_i, \hat{t}_{i+1}]$. Furthermore, the optimal performance is either $G(\hat{t}_i, \hat{t}_{i+1,\hat{n}})$ or $G(\hat{t}_{i+1}, \hat{t}_{i+2,\hat{n}})$, if

$t_0 \in [t_e, \hat{t}_i] \cup [\hat{t}_{i+1}, D_u]$. *$G(\hat{t}_i, \hat{t}_{i+1,\hat{n}})$ is the project performance when the downstream starts at $\hat{t}_i$ and the communication policy is $\hat{t}_{i+1,\hat{n}}$.*

(b) If $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_{\hat{n}-1}}^{D_u} \mu(x)dx\}] < 0$, then $t_0^* = D_u$ (i.e. sequential process is optimal).

(c) If $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_i}^{D_u} \mu(x)dx\}] = 0$, then $t_0^* = \hat{t}_{i+1}$ and $G^* = G(\hat{t}_{i+1}, \hat{t}_{i+2,\hat{n}})$.

Proposition 4.2 shows that if the optimal communication policy is known when t_0 is given, the optimal communication policy and project performance are also clear when the downstream starts at t_i^*. Proposition 4.3 gives the overlapping and communication policies in different situations. Note that Proposition 4.3(a) only gives a range of the downstream start time. However, it should be accurate enough since the optimal point is often not required for most project management problems.

The steps of solving the concurrency problem are that: first let $t_0 = t_e$ and derive the optimal communication policy by Algorithm 1; then, by Proposition 4.2 and Proposition 4.3, the optimal or appropriate communication strategy, overlapping policy, and project performance can be derived; finally, if $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_{\hat{n}-1}}^{D_u} \mu(x)dx\}] > 0$ and an accurate solution is required, then we need to examine and compare the solutions when the downstream starts in $[\hat{t}_i, \hat{t}_{i+1}]$, which can be done using Algorithm 1.

In certain industries, project performance is mainly determined by development cycle time (Carrillo and Franza 2006). Time-to-market problem is a special case of the concurrency problem. We assume $r \geq 1$, which is valid for many projects because of the existence of learning effect (Browning and Eppinger 2002). Some additional insights are derived.

Proposition 4.4

Assume $r \geq 1$.

(*a*) *The optimal development cycle time for the time-to-market problem satisfies*

$$T(t_e, \hat{t}_{1,\hat{n}}) - \beta \leq T^* \leq T(t_e, \hat{t}_{1,\hat{n}}) . \tag{4.15}$$

(*b*) *The optimal start time of the downstream stage satisfies*

$$t_e \leq t_0^* \leq \min(\hat{t}_1, t_e + \varphi) , \tag{4.16}$$

where $\varphi = \beta / \exp\{-k \int_{t_e}^{D_u} \mu(x) dx\}$. *If* β *is small,* $t_0^* \approx t_e$. *In other words, complete overlapping should be applied when* β *is small.*

From Proposition 4.4(b) we can see that complete overlapping is optimal when β is small. This result is consistent with the conclusions derived by previous models for projects with low communication cost (e.g. Roemer et al. 1999). However, our result also shows that the downstream start time should be delayed if communication takes time. It indicates that complete overlapping is not necessary a global optimum for time-to-market problem, although it is recommended by many researchers. Proposition 4.4(a) shows the optimal project cycle time for the time-to-market problem.

4.4.3 The Linear Evolution Case

In this section we list the management insights obtained for the projects with linear upstream evolution (i.e. the modifications are generated as a homogeneous Poisson process) and present a computationally efficient solution procedure which determines the optimal policies through a binary search. The results are nevertheless important since most evolution curves can be approximated by piecewise linear functions, which corresponds to a situation where the decision makers only estimate the average modification rates in different periods (Loch and Terwiesch 1998, Roemer and Ahmadi 2004). Thus, the insights and algorithm presented here also constitute the basis for solving the general case.

Proposition 4.5

(*a*) *Given* n *and* t_0, *the optimal information exchange policy can be determined by the following equations*

$$Q_i = \frac{1-\exp\{-k\mu Q_{i-1}\}}{k\mu} \qquad 2 \le i \le n. \tag{4.17}$$

$$\sum_{i=1}^{n} Q_i = D_u - t_0 .$$

Furthermore, $\sum_{i=1}^{n} Q_i$ *strictly increases with* Q_1. *Therefore, the optimal* Q_1 *can be identified through a binary search.*

(*b*) $Q_i^* > Q_{i+1}^*$, *i.e. management should increase the communication frequency over time.*

(*c*) n^* *is non-increasing with* t_0.

(*d*) *Given* t_0, *the profit function* $G^*(t_0,n)$ *is concave with respect to* n.

Proposition 4.5(a) shows that $\sum_{i=1}^{n} Q_i$ strictly increases with Q_1. Therefore, given n and t_0, we can easily find the optimal Q_1 through a binary search. The idea behind the binary search procedure is that whether $\sum_{i=1}^{n} Q_i$ is greater or less than $D_u - t_s$ at a trail solution definitely indicates whether Q_1 should be decreased or increased. Proposition 4.5(d) shows that the object function is concave with respect to n. That is, if project profit increases with n, then $n^* > n$. Conversely, if project profit decreases with n, then $n^* \le n$. Therefore, given t_0, the optimal communication policy can be derived by a two-dimensional binary search. These deliberations give rise to the following algorithm.

Algorithm 2 (for the linear evolution case)

Notation

$\underline{n}$ = current lower bound of n^*, $\bar{n}$ = current upper bound of n^*, $\underline{Q}_1$ = current lower bound of Q_1^*, $\bar{Q}_1$ = current upper bound of Q_1^*.

Initiation:

Step 0. Let $\underline{n} = 1$, $\bar{n} = \left\lceil 1 + \frac{c_t(D_u - t_0)}{c_t\beta + c_\beta} \right\rceil$, $\underline{Q}_1 = 0$, and $\bar{Q}_1 = D_u - t_0$.

Iteration 1:

Step 1. Let $n = \lceil (\bar{n} - \underline{n})/2 \rceil - 1$ and $\tilde{n} = \lceil (\bar{n} - \underline{n})/2 \rceil$.

Step 2. **Iteration 2:**

Step 2.1. Let $Q_1 = (\underline{Q}_1 + \overline{Q}_1)/2$. Computer $\sum_{i=1}^{n} Q_i$.

Step 2.2. If $\sum_{i=1}^{n} Q_i < D_u - t_0$, then $\underline{Q}_1 = Q_1$, go to *Step 2.1*; if $\sum_{i=1}^{n} Q_i > D_u - t_0$, then $\overline{Q}_1 = Q_1$, go to *Step 2.1*. Else, compute $G^*(t_0, n)$; let $\underline{Q}_1 = 0$, $\overline{Q}_1 = D_u - t_0$.

Step 2.3. Let $\tilde{Q}_1 = (\underline{Q}_1 + \overline{Q}_1)/2$. Compute $\sum_{i=1}^{\tilde{n}} \tilde{Q}_i$.

Step 2.4. If $\sum_{i=1}^{\tilde{n}} \tilde{Q}_i < D_u - t_0$, then $\underline{Q}_1 = \tilde{Q}_1$, go to *Step 2.3*. If $\sum_{i=1}^{\tilde{n}} \tilde{Q}_i > D_u - t_0$, then $\overline{Q}_1 = \tilde{Q}_1$, go to *Step 2.3*. Else, compute $G^*(t_0, \tilde{n})$; let $\underline{Q}_1 = 0$, $\overline{Q}_1 = D_u - t_0$.

Step 3. If $\overline{n} - \underline{n} \le 2$ and $G^*(t_0, n) \ge G^*(t_0, \tilde{n})$, then $n^* = n$, $Q_1^* = Q_1$, $G^*(t_0) = G^*(t_0, n)$, stop; if $\overline{n} - \underline{n} \le 2$ and $G^*(t_0, n) < G^*(t_0, \tilde{n})$, then $n^* = \tilde{n}$, $Q_1^* = \tilde{Q}_1$, $G^*(t_0) = G^*(t_0, \tilde{n})$, stop. Else, if $G^*(t_0, n) < G^*(t_0, \tilde{n})$, then $\underline{n} = n$, go to *Step 1*; if $G^*(t_0, n) \ge G^*(t_0, \tilde{n})$, then $\overline{n} = \tilde{n}$, go to *Step 1*.

Because the binary search for n can be performed in $\log(1 + \frac{c_t(D_u - t_0)}{c_t\beta + c_\beta})$ steps and, for each n, the optimal Q_1 can be identified in $\log(D_u - t_0)$ steps, the overall computational effort of the algorithm is $O(\log(1 + \frac{c_t(D_u - t_0)}{c_t\beta + c_\beta}) \cdot \log(D_u - t_0))$.

Propositions 4.4(b), 4.4(c), and 4.4(d) are the guidelines for structuring concurrent processes. Proposition 4.4(b) indicates that the communication frequency should be increased over time. Proposition 4.4(c) describes the relationship between the number of meetings and the degree of overlapping. More meetings should be arranged when the overlapping degree is high. Proposition 4.4(d) shows that, given t_0, the object function is concave with respect to n. This is important from a managerial perspective. Consider an organization attempting to improve information exchange policy across similar projects. Since $G^*(t_0, n)$ is concave, the organization can rely on marginal improvement: continuously increase n until it decreases project performance.

4.5 Model Application

In this section, we illustrate the model with its application in a consumer electronics company which is one of the world's leading white goods home appliance manufacturers. It manufactures home appliances in over 15,100 different specifications under 96 categories and has approximately 50,000 employees throughout the world. Its product categories range from refrigerators, refrigerating cabinets, air conditioners, washing machines, televisions, mobile phones, home theatre systems, computers, water heaters, DVD players and integrated furniture. The global revenue of the company for 2005 was more than 12 billion US dollars.

As is common in consumer electronics industry, concurrent engineering is applied in the case study company. Our model was applied to improve the refrigerator development process. There are six types of refrigerator development projects in the company, one of which is analyzed in this section to illustrate the methodology proposed. The data set shown in this chapter was estimated together with the experienced engineers according to the records of the previous projects (the documents of design issues and project schedule). In most companies, these data are available (Krishnan et al. 1997, Roemer et al. 2000, Lin et al. 2008) which allows us to derive the development policies on reliable inputs. Actually, this is one of the most important reasons that why the concepts of upstream evolution (the evolution of upstream modification) and downstream sensitivity, which were proposed by Krishnan et al. 1997 and extended by Loch and Terwiesch (1998), have had a strong influence on the literature of concurrent engineering.

Our focus is on the Detail Design (upstream) and Tooling Fabrication (downstream) since they are the key drivers of development time and cost and are usually concurrently executed. After completing the preliminary detail design, the drawings are reviewed by experienced engineers and two rounds of prototype testing are arranged. The design drafts are continuous modified in the testing process. As shown in Figure 4.4, the modification process lasts 48 working days (working 6 days per week). For the previous projects, the downstream starts on the 22nd day when the testing of the first prototype is almost finished. After that, no meeting is arranged until the drawings are finalized. The estimated setup time and cost for information exchange are 1 day and 2000 dollars respectively.

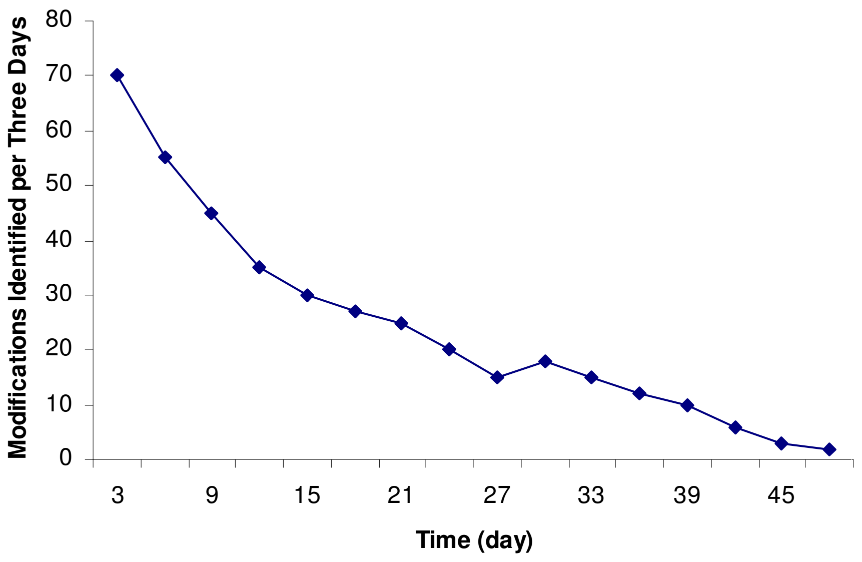


Figure 4.4 Modification process

According to Figure 4.4, we derive the function of cumulated design modifications using regression analysis by minimizing SSE (sum of squared residuals). The outputs of regression analysis are summarized in Table 4.2 and plotted in Figure 4.5. We see that the proposed function fits well to the real data set.

Table 4.2 Assessing model fit to data

	DF	Sum of Squares	Mean Square	R-Square
Model	2	1.46009E+06	7.30045E+05	0.999116
Error	15	200.659	13.37726667	
Uncorrected Total	17	1.46029E+06		
Corrected Total	16	226926		

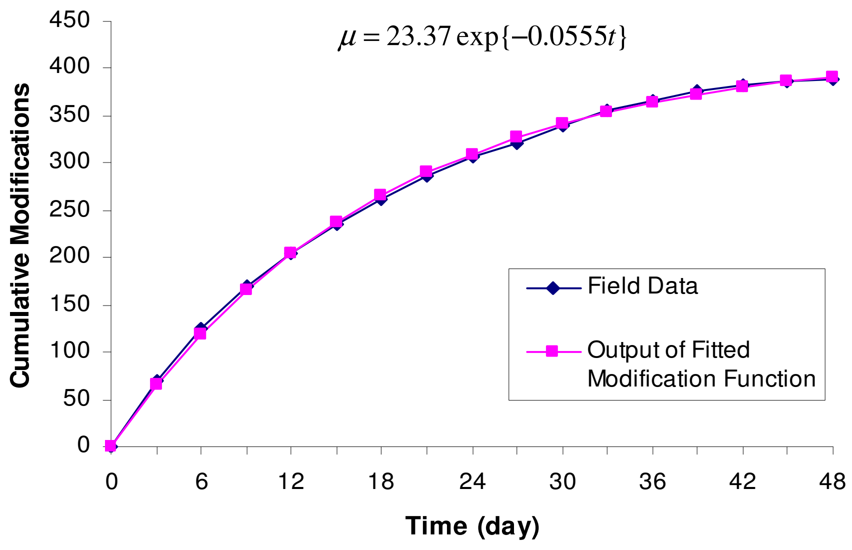


Figure 4.5 Cumulated design modifications

According to the documents of design issues and project schedule, in average, about 100 modifications were transferred to the downstream at the beginning of the 49th day and corrupted 30% of downstream work. The downstream progress at that time is 27 days and the downstream development duration is 44 days. By the equations derived in Section 3, the impact factor k equals 0.65%. The other parameters, which are commonly used for project modeling (Ford and Sterman 1998, Repenning 2001), were derived from historical records or estimated by experienced engineers.

By these parameters, we investigated the optimal communication policy for complete overlapping using Algorithm 1. The results are: $\hat{n} = 5$, $\hat{t}_{1,\hat{n}} = 13.25, 21.26, 29.28, 37.96, 48$, and $G^*(t_0) = 107.99$. Then, by Proposition 2a, we derive that $t_0^* \in [13.25, 21.26]$, $G(\hat{t}_1, \hat{t}_{2,\hat{n}}) = 162.73$, and $G(\hat{t}_2, \hat{t}_{3,\hat{n}}) = 179.03$. Therefore, an appropriate development policy is that: $t_0 = 21$; the communication occurs at the 30th, 39th, and 49th day (assuming the solution must be integer). If an accurate solution is required, we need to compare the project performance when the downstream starts in $[13.25, 21.26]$. Figure 4.6 shows the change of project performance with t_0, by which we derive $G^* = G(\hat{t}_2, \hat{t}_{3,\hat{n}})$.

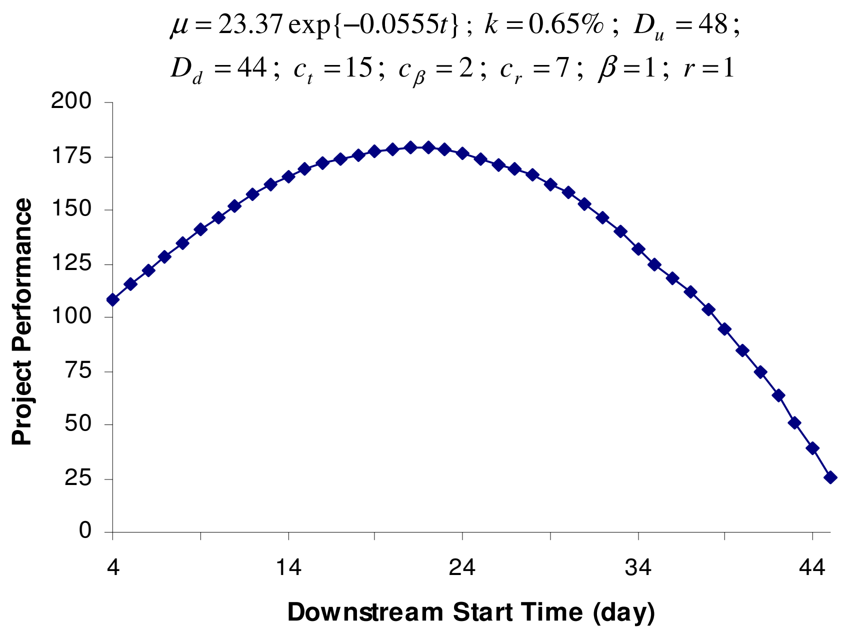


Figure 4.6 The effect of overlapping policy on project performance

Comparing the analytical results with the development policies applied in the company, we see that the previous communication strategy is far from optimal. If two more meetings were arranged on the 30th and 39th day, the gain from CE can be increased from $87,000 to $179,030.

The sensitivity of development policies on setup time and cost is also studied. As shown in Table 4.3, n^* decreases with c_β and β and t_0^* increases with c_β and β. However, n^* and t_0^* change discontinuously. For example, as shown in Table 4.3, the development policies are not affected when the communication cost and time changes from 150% to 175%. This is important from a managerial perspective. Consider an organization attempting to continuously decrease its setup time and cost for information exchange by applying new technologies. Since n^* and t_0^* change discontinuously, the organization can only adjust its development policies when the setup time and cost reach a critical value.

Table 4.3 The impact of communication time and cost on development policies

Setup Time and Cost	n^*	t_0^*	t^*_{1,n^*}	G^*
25%	7	17	20.94, 24.90, 28.97, 33.22, 37.74, 42.62, 48	234.24
50%	5	19	24.15, 29.43, 35.03, 41.14, 48	210.23
75%	4	19	6.48, 6.75, 7.36, 48	192.69
100%	3	21	29.10, 37.86, 48	179.03
125%	3	21	29.10, 37.86, 48	166.28
150%	2	24	35.06, 48	155.77
175%	2	24	35.06, 48	147.27
200%	2	24	35.06, 48	138.77
225%	2	24	35.06, 48	130.27
250%	2	24	35.06, 48	121.77
275%	2	24	35.06, 48	113.27
300%	1	30	48	106.71

4.6 Discussion and Conclusion

Literature (e.g. Clark and Fujimoto 1991, Loch and Terwiesch 1998, Helms 2004) shows that overlapping and communication policies are interrelated. Overlapping may degrade project performance if appropriate communication policy is not followed. We present an analytical model that optimally schedule overlapping and communication policies. The dynamic relationship between downstream progress and project characteristics is explicitly studied, which ensures the accuracy and applicability of the insights and algorithms derived. Furthermore, the availability of the input data and the applicability of the model are validated by a case study in a consumer electronics company.

This research yields four main results. Firstly, by Proposition 4.1, Algorithm 1 is proposed which can help us determine the optimal communication policy when overlapping degree is predetermined. Then, by the insights given in Propositions 4.2 and 4.3, an appropriate overlapping degree and the corresponding communication policy are determined. After that, the time-to-market problem is discussed in Proposition 4.3. Previous research (e.g. Roemer et al. 1999) argues that complete overlapping is optimal for time-to-market problem. However, our result shows that the downstream start time should be delayed if communication takes time. The conflict exists because previous studies usually ignore communication time and cost. Our result takes the effect of communication time into account and thus is more general. Finally, Proposition 4.4 presents several important management insights for the linear evolution case. The linear evolution case is studied in detail because: (1) all the evolution processes can be approximated by piecewise linear functions; (2) the algorithm for the linear evolution case is much more efficient; (3) the insights can guide management to determine communication policy without modeling.

Nonetheless, we would like to point out the limitations to the approach presented here. Firstly, our model can only be built and analyzed when historical data is available. For derivative projects which account for the majority of product development projects (Whitney 1990, Wang and Yan 2005), these data are typically available (Krishnan et al. 1996, Roemer et al. 2000, Helms 2002). However, for new projects, we can only build a model based on estimated data which may lead to inaccurate results. Secondly, although the model is built on previous literature and illustrated with a case study done in consumer electronics industry, it is important to test and improve it through real applications in other industries since the development processes may be different. Thirdly, our model is deterministic and therefore does not directly address risk, while many facets of risk can be addressed by running sensitivity analyses on the input data. Finally, the downstream progress is derived on the mean value of upstream modifications. It is an approximation of the mean value of downstream progress.

The model discussed in Chapter 4 will reduce to the model in Chapter 3 when communication cost is zero. As we know, communication cost can never be zero in practice. Therefore, theoretically, all the projects can be studied using the model discussed in Chapter 4. However, as communication cost becomes small, it is computationally difficult to derive appropriate development policies using this model. Furthermore, many simple and useful insights in Chapter 3 cannot be derived with the model in Chapter 4. Researchers usually ignore the communication cost when it is insignificant comparing to the project cycle time and total development cost (e.g. Krishnan et al. 1997).

CHAPTER 5

A SYSTEM DYNAMICS MODEL OF OVERLAPPED ITERATIVE PROCESSES

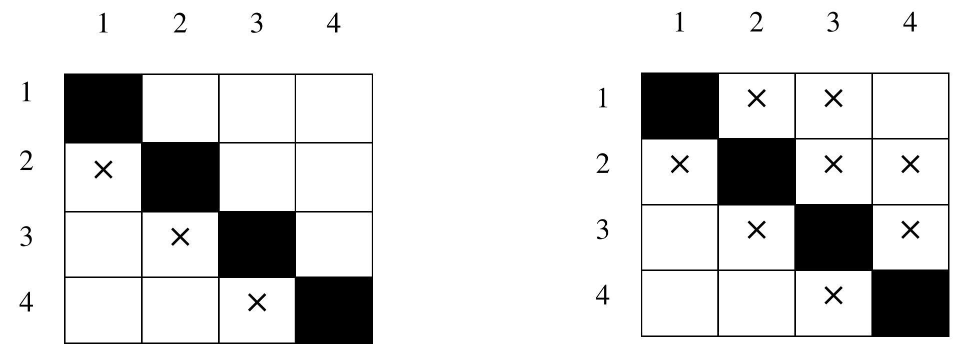
In Chapter 3 and Chapter 4, concurrent exaction of sequentially dependent stages is studied. However, for some projects, the development stages are interdependent. How to estimate the cycle time of these product development projects? What is the effect of activity properties and development policies on project performance? This chapter presents a new product development (NPD) process model, termed Dynamic Development Process Model (DDPM), for managing overlapped iterative product development. The model was validated with data from a mobile phone development project. It was employed to identify appropriate policies for the overlapped iterative projects in the case study company. These identified policies were implemented in the company and led to marked improvement in project performance, thus demonstrating the viability of the model.

5.1 Introduction

Traditional project management models, such as Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT) (Moder et al., 1983; Badiru, 1993; Golenko-Ginzburg and Gonik, 1996), describe development processes which are relatively stable, with activity duration estimates and precedence relationships representing the network of development activities. However, these models ignore iterations or implicitly incorporate iterations into duration estimates (Ford and Sterman, 2003a, 2003b), limiting the capability of these scheduling techniques in modeling NPD processes.

Therefore, some models other than CPM/PERT have been developed to study iterative product development processes. Design structure matrix (DSM) was developed to describe durations and rework probabilities of development activities (Eppinger et al., 1994; Steward, 1981). Several other researchers (e.g. Ahmadi and Wang, 1999; Belhe and Kusiak, 1996) have developed extensions by considering the dynamics of rework probability and activity duration. Smith and Eppinger (1997a, 1997b) developed analytic models of sequential and parallel design iterations and addressed the effect of iterations among project phases on project cycle time with Design Structure Matrix and reward Markov chain. Browning and Eppinger (2002) developed the first DSM-based simulation model which analyzed NPD iterations in a generalized project network. After that Cho and Eppinger (2005) developed the second-generation DSM-based simulation model which accounts for resource constraints. Cooper (1980, 1993a, 1993b, 1993c) and several other researchers (Ford and Sterman, 1998; Repenning, 2001; Richardson and Pugh III, 1981) built System Dynamics (SD) models to understand the continuous evolution of NPD projects. While these models have advanced our understanding on the dynamics of iterative NPD projects, they do not take into account the overlapping nature of development activities, which is a common practice to reduce project cycle time (Lawson and Karandikar, 1994; Roemer and Ahmadi, 2004).

Previous empirical studies showed that overlapping can reduce project cycle time at the cost of additional development effort (Smith and Reinertsen, 1995; Helms, 2004) and the effect of overlapping is closely related to the uncertainty of development projects (Eisenhardt and Tabrizi, 1995; Terwiesch and Loch, 1999). Based on these studies some models have been developed to investigate overlapped processes in which two development stages are sequentially dependent. Krishnan *et al.* (1997) developed a framework of concurrent development processes with sequential dependence to determine the optimal number and timing of information transfer. They showed that "upstream information evolution" and "downstream sensitivity" are the two properties affecting optimal overlapping strategies. Loch and Terwiesch (1998) adapted the concepts of evolution and sensitivity: "upstream information evolution" is defined as the continuous design modification process; "downstream sensitivity" represents the impact of a modification on downstream rework. Based on these concepts, they developed an analytical model and derived the optimal communication strategies for overlapped processes with sequential dependence. Roemer et al. (2000) analyzed the time-cost tradeoffs in multistage product development. Chakravarty (2001) studied the trade-offs between the overlapping risk and the project time saved. Some special cases were analyzed to establish useful insights for overlapping development activities. Unlike previous research we developed a model for overlapped iterative product development, where downstream activities may discover upstream errors and give feedback to the corresponding activities (Figure 5.1). The extension from sequentially dependent to interdependent process makes it possible to simulate and study the effect of overlapping for complex development projects.



(a) Sequentially dependent stages

(b) Interdependent stages

Figure 5.1 DSM representation of sequential dependence and interdependence

To model and analyze overlapped iterative product development we developed the Dynamic Development Process Model (DDPM) using System Dynamics (SD) methodology. Discrete event simulation model and continuous time model (system dynamics) are two methods commonly used to simulate NPD processes. The former assumes that product development process is composed of a finite set of activities and information flow only exists at the beginning or at the end of an activity. In contrast, the SD approach to project management treats the process of each phase as continuous work flow. It is consistent with the assumption in the overlapping models (e.g. Loch and Terwiesch, 1998; Roemer et al., 2000; Roemer and Ahmadi, 2004). Through building the relationship between work flow and information flow, we simulate the continuous upstream information evolution and its effect on downstream rework using SD approach.

The rest of this chapter is organized as follows. In section 5.2, we develop the concepts of *Rework Due to Development Errors* and *Rework Due to Corruption*. According to our field study and literature review (e.g. Joglekar et al., 2001; Krishnan et al., 1997) of NPD process, these are the types of rework existing in overlapped iterative product development. In the next section, we use these concepts to construct the dynamic development process model, followed by the validation of DDPM in section 5.4. The effect of corruption on project performance is discussed in section 5.5. Then policies for the overlapped iterative projects in a development company are analyzed in section 5.6. The successful application of the proposed new policies further validated the model. Conclusions are summarized in section 5.7.

5.2 Rework due to development errors and corruption

We follow the information-based view of product development (Clark and Fujimoto, 1991) in which individual development activities are the information-processing units that receive information from their preceding activities and transform it into new information to be passed on to subsequent activities. The information changes between activities are embedded in the tasks carried out. Each activity of the product development process is related to the development tasks such as customer specifications at concept development phase, detailed engineering drawings at detail design phase, and part dimensions at pilot production phase. The ultimate objective is to ensure these tasks are carried out correctly, at low cost and in short time. We describe and simulate the rework process in the form of *Rework Due to Development Errors* and *Rework Due to Corruption*.

5.2.1 Rework Due to Development Errors

Product development, even for derivative products, is a process with much uncertainty. Consequently, many tasks are incorrectly done in the completion and rework processes. These tasks are termed as *Development Errors* (*DE_s*). *Rework Due to Development Errors* refers to rework or rectification of *DE_s* which are identified through review and testing activities.

We illustrate the rework process using a stock and flow structure (Figure 5.2). Stocks represent the accumulation of tasks and flows represent the rates of development activities (Sterman, 2004). Tasks initially reside in the *Tasks Remaining* (*T_r*) stock. As the project begins and progresses, tasks correctly done flow into the *Tasks-done-correctly* (*T_c*) stock while tasks containing errors or defects add to the *Development Errors* stock. *Development Errors* may be identified by a testing activity and flow into the *Tasks to be Reworked* stock. Therefore the total number of *Development Errors* decrease as some of them are correctly reworked. Because rework quality is usually not perfect, tasks which are incorrectly reworked flow back into the *Development Errors* stock. Some of the reworked tasks in the *Development Errors* stock may need to flow into this rework cycle one or more times. When rework quality is low, this vicious rework cycle dominates the development process. According to Cooper (1993a), “A quality of 0.20 will require five cycles of work and cost (four full rework cycles) to ‘get it right’”.

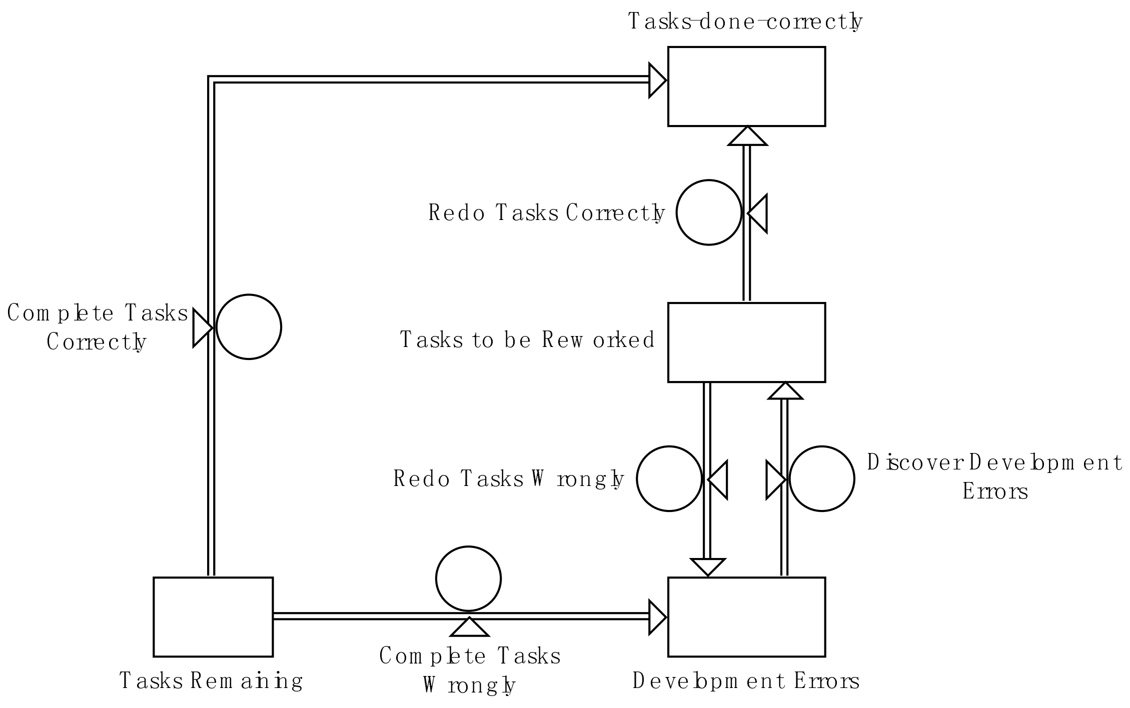


Figure 5.2 Rework due to development errors

5.2.2 Rework Due to Corruption

Rework Due to Corruption refers to rework or rectification when the change of tasks in an upstream phase corrupts the relevant tasks in the downstream phases, whether the downstream tasks are done correctly or not. In other words, some tasks need to be reworked because they start on incorrect information from upstream phases. We termed this phenomenon as *Corruption*. In this model, the n^{th} $(n \geq 2)$ order rework, i.e. the downstream corruption caused by upstream corruption, is also studied.

Tasks corrupted are dependent on the reworked tasks of the upstream phase, the *Dependency* of the development phases, and the progress of the downstream phase (*Tasks Done*). The reworked tasks of the upstream phase are positively related to *Rework Due to Corruption*. More changed tasks inevitably mean that more tasks may be reworked in the downstream phases (Krishnan et al., 1997). *Dependency* represents the relationship between the tasks corrupted in the downstream phase and the fraction of the tasks changed in the upstream phase. It is also positively related to *Rework Due to Corruption*. *Tasks Done* accounts for the reason why more rework is needed in overlapped NPD process than the rework in sequential process. For traditional sequential product development process most of the *Development Errors* can be found and resolved before the downstream activities start (at that time *Tasks Done* of downstream phase is equal to zero and no *Corruption* arise). For example, in a fully sequential process, pilot production only starts after detail design has been completed and most of the quality problems have been resolved. However, in practice, pilot production usually starts before the upstream activities have been completed in order to reduce project cycle time. Therefore, in today's overlapped NPD process, *Corruption* accounts for a large portion of rework and affects product development performance seriously (Krishnan et al., 1997).

The stock and flow structure of *Rework Due to Corruption* is shown in Figure 5.3. Certain percentage of downstream tasks is completed based on wrong information from *Development Errors* of upstream phase. These tasks, together with other tasks, reside in the *Tasks-done-correctly* stock and the *Development Errors* stock. Corruption occurs when *DEs* of an upstream phase are identified. The tasks associated with *DEs* of upstream phases leave the *Tasks-done-correctly* stock and the *Development Errors* stock, and then flow into the *Tasks to be Reworked* stock.

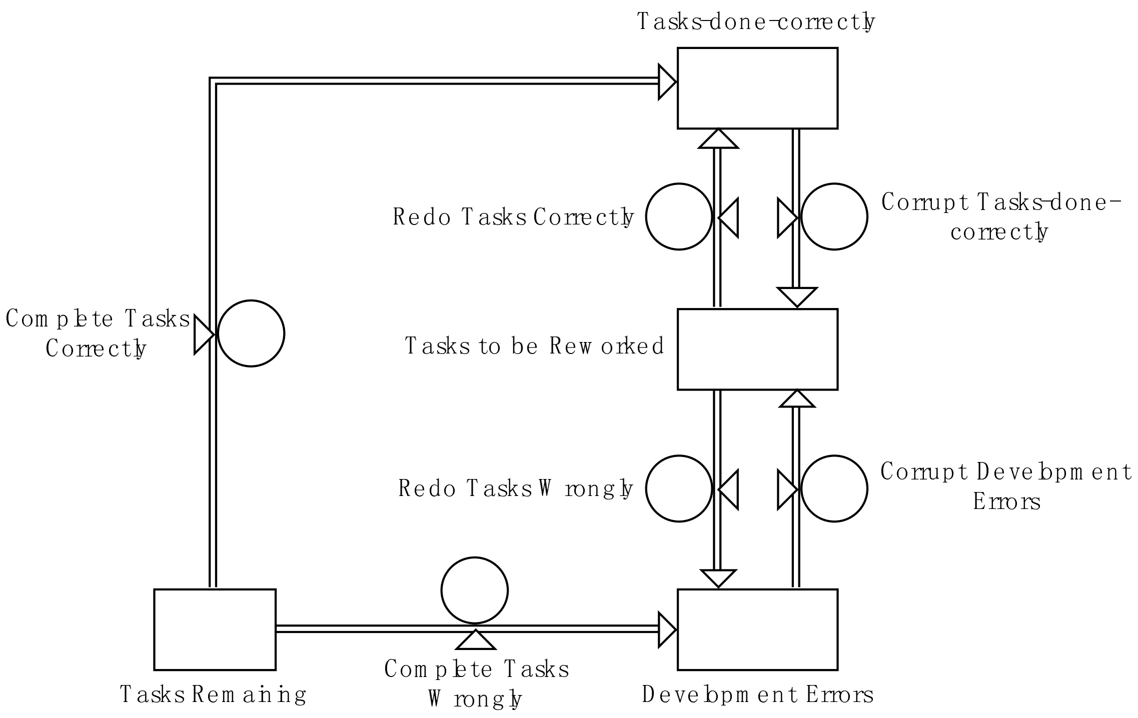


Figure 5.3 Rework due to corruption

5.2.3 Example

Here we give a simple example to further illustrate the rework process. As shown in Figure 5.4, *Slot A* and *Slot B* are determined by four dimensions. These dimensions are derived in *Phase 1*, and then the slots are fabricated in *Phase 2*. The detailed development process can be described as follows:

1) In the beginning, all of the dimensions reside in *Tasks Remaining 1* and the slots reside in *Tasks Remaining 2*.

2) Assuming the development activity in phase 1 is not perfect, *Dimension 3* flows into the *Development Errors 1* stock. The other dimensions flow into the *Tasks-done-correctly 1* stock. The slots are kept in *Tasks Remaining 2*.

3) *Phase 2* starts before the development error in *Phase 1* is identified and revised. Assuming the development activity in *Phase 2* is perfect, *Slot A* and *Slot B* are exactly

fabricated according to the dimensions and flow into the *Tasks-done-correctly 2* stock. The states of the dimensions are not changed.

4) After that the error of *Dimension 3* is identified and revised. It is represented as *Rework Due to Development Errors* in our model.

5) Since *Slot B* is determined by *Dimension 3,* it needs to be revised accordingly. We term this type of rework as *Rework Due to Corruption*.

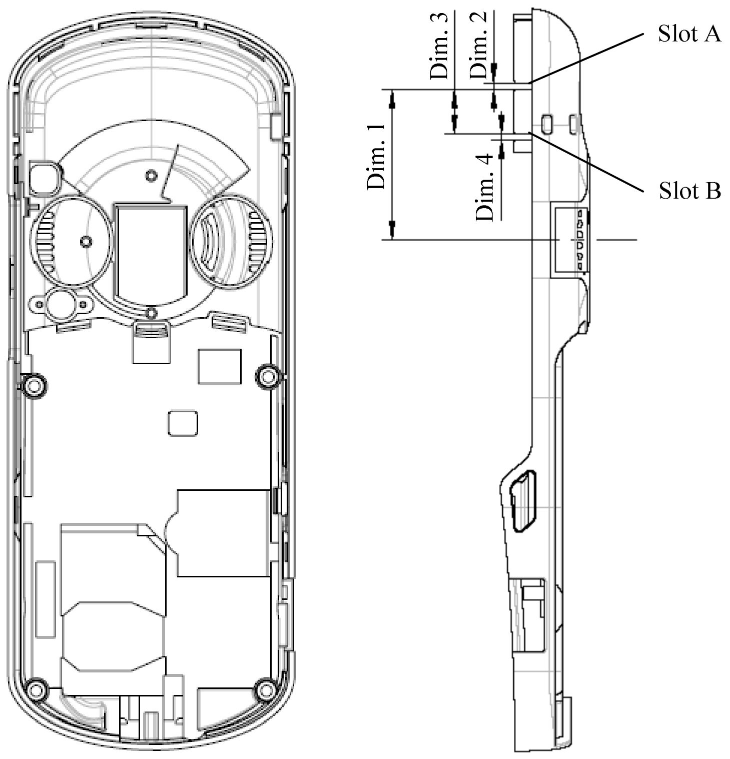


Figure 5.4 Base rear of a mobile phone

5.3 Dynamic Development Process Model

5.3.1 Stocks and Flows

We combine *Rework Due to Development Errors* and *Rework Due to Corruption* in one stock and flow structure (Figure 5.5).

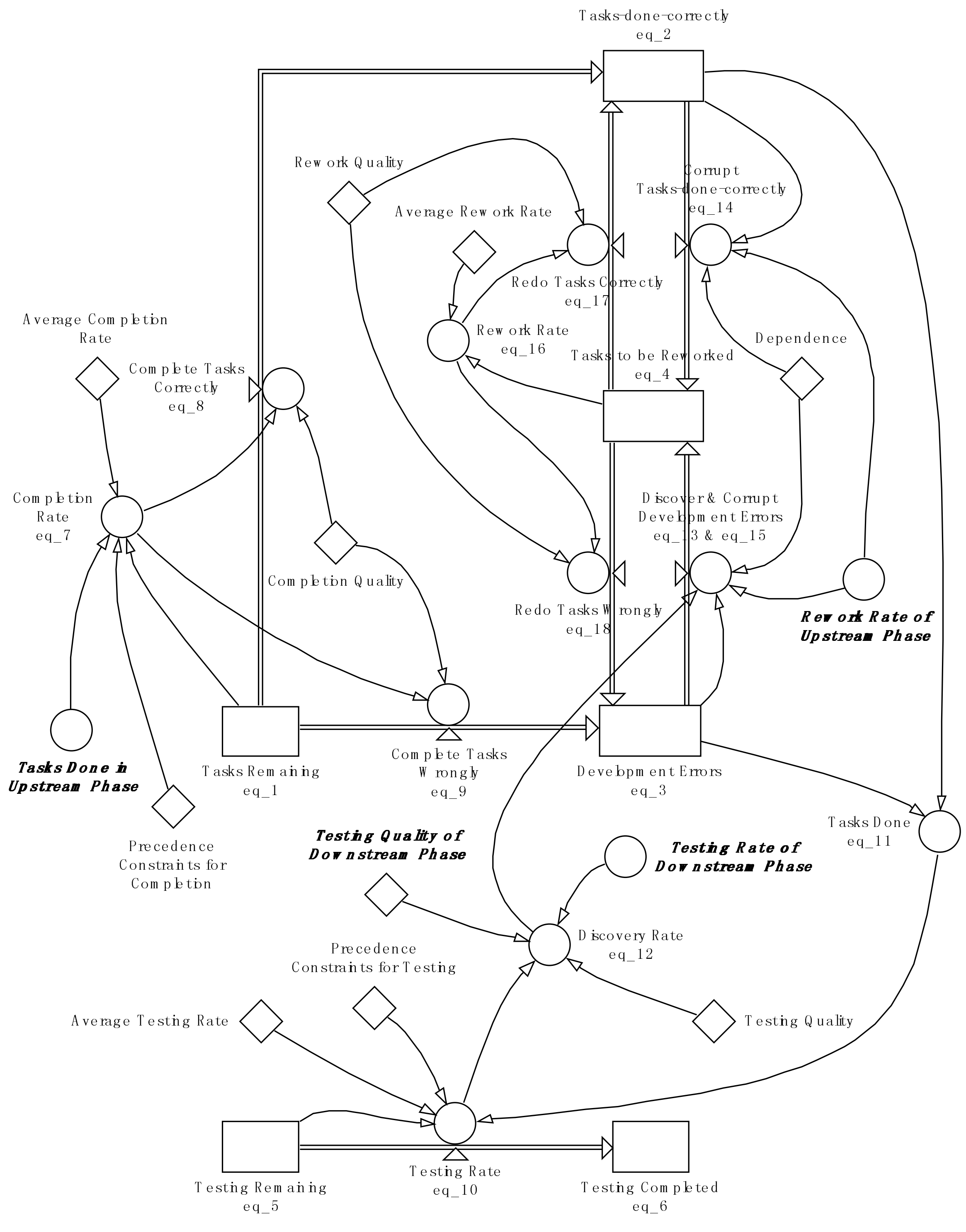


Figure 5.5 Dynamic development process model (DDPM)

Four stocks and six flows are used to represent the completion and rework processes. The stocks (*Tasks Remaining* (T_r), *Tasks-done-correctly* (T_c), *Development Errors*

(DE_s), and *Tasks to be Reworked* (T_{tr})) represent the sizes of the backlogs of tasks. The sizes of the stocks change due to the flows related to development activities. In Figure 5.5, *Complete Tasks Correctly* (c_c) and *Complete Tasks Wrongly* (c_w) represent the completion activity; *Testing Rate* (g_{re}) and *Discover Development Errors* (d_e) represent the testing activity; *Redo Tasks Correctly* (r_c) and *Redo Tasks Wrongly* (r_w) represent the rework activity; *Corrupt Tasks-done-correctly* (k_c) and *Corrupt Development Errors* (k_e) represent the *Corruption* caused by upstream rework. Therefore the processes described in section 5.2 can be represented by the following differential equations (at the start of a project, $T_r(0)$=100%, and $T_c(0)$, $DE_s(0)$, and $T_{tr}(0)$ all equal zero):

$$(d/dt)(T_r) = - c_c - c_w \tag{5.1}$$

$$(d/dt)(T_c) = c_c + r_c - k_c \tag{5.2}$$

$$(d/dt)(DE_s) = c_w + r_w - (d_e + k_e) \tag{5.3}$$

$$(d/dt)(T_{tr}) = (d_e + k_e) + k_c - r_c - r_w \tag{5.4}$$

Testing process is represented by two stocks and one flow (Figure 5.5). Tested tasks leave the *Testing Remaining* (G_r) stock, pass through the *Testing Rate* (g_{re}), and then accumulate in the *Testing Completed* (G_c) stock (at the start of a project $G_r(0)$=100% and $G_c(0)$=0). Mathematically:

$$(d/dt)(G_r) = -g_{re} \tag{5.5}$$

$$(d/dt)(G_c) = g_{re} \tag{5.6}$$

We formally model the flows related to completion, testing, corruption, and rework with the equations in the rest of this section. The input parameters needed to build the model are listed in Table 5.1 and shown as diamonds in Figure 5.5. The italic font shown in Figure 5.5 represents the parameters of upstream and downstream phases. Tasks and *Development Errors* are in percentage values in our model. For example at the start of a project all the development tasks reside in the *Tasks Remaining* stock, so that $T_r(0)$ is 100%.

Table 5.1 Model parameters and performance measures

Parameters	**Definition**
Precedence Constraints	The condition to start an activity
Average Activity Rate	The average rate of completing a development activity
Completion/Rework Quality	The percentage of tasks correctly done
Testing Quality	The percentage of *Development Errors* identified in the testing process
Dependency	The percentage of downstream tasks will be affected by one percentage of upstream changes.
Measures of Project Performance	**Definition**
Project Quality	The percentage of *Development Errors* remained when the overall project is completed.
Cycle time	The duration from the start to the end of a project
Development Effort	The tasks completed and reworked from the start to the end of a project

5.3.2 Completion

Three development activities drive the flows of tasks in NPD process: completion, rework and testing. The progress rate for each of three development activities is the lesser of the average development rate and the rate allowed by tasks available. Therefore the *Completion Rate* (c_{re}) is the minimum of the *Average Completion Rate* (A_c), and the number of *Tasks Remaining* (T_r) divided by the time step (τ) of the simulation model when *Tasks Done in Upstream Phase* (T_{du}) is greater or equal to *Precedence Constraints for Completion* (P_c). Otherwise *Completion Rate* equals zero. According to *Completion Quality* (C_q), *Completion Rate* is decomposed into *Complete Tasks Correctly* and *Complete Tasks Wrongly.* These conditions can be represented by the following equations:

$$c_{re} = \text{IF } (T_{du} \geq P_c) \text{ THEN } (Min\ (A_c, T_r/\tau)) \text{ ELSE } (0) \tag{5.7}$$

$$c_c = c_{re} * C_q \tag{5.8}$$

$$c_w = c_{re} * (1 - C_q) \tag{5.9}$$

5.3.3 Testing

Similar to completion rate, *Testing Rate* is equal to the minimum of the *Average Testing Rate* (A_g) and the number of *Testing Remaining* (G_r) divided by the time step if *Tasks Done* (T_d) is greater or equal to *Precedence Constraints for Testing* (P_g). Otherwise it is zero. Rework arises when the *Development Errors* are found by a testing activity. As it is typical that we cannot find all the *Development Errors* through a single round of testing, *DEs* are likely to be discovered by subsequent testing activities. We model *Discovery Rate* (d_{re}) as the sum of the product of *Testing Quality* (G_q) and *Testing Rate* from the current testing activity (denoted by *m*) to the last testing activity of the project (denoted by *n*). *Discover Development Errors* (d_e) is the result of *Discovery Rate* multiplied by *Development Errors*. Mathematically:

$$g_{re} = \text{IF } (T_d \geq P_g) \text{ THEN } (\text{Min } (A_g, G_r/\tau)) \text{ ELSE } (0) \tag{5.10}$$

$$T_d = T_c + DE_s \tag{5.11}$$

$$d_{re} = \sum_{i=m}^{n} G_{q_i} * g_{re_i} \tag{5.12}$$

$$d_e = d_{re} * DE_s \tag{5.13}$$

Figure 5.6 Parameters of dynamic development process model

Parameters	**Definition**
T_r	Tasks remaining
T_d	Tasks done
T_{du}	Tasks done in upstream phase
T_c	Tasks-done-correctly
T_{tr}	Tasks to be reworked
DE_s	Development errors
G_r	Testing remaining
G_c	Testing completed
A_c	Average completion rate
A_g	Average testing rate
A_r	Average rework rate
P_c	Precedence constraints for completion
P_g	Precedence constraints for testing
C_q	Completion quality
G_q	Testing quality
R_q	Rework quality
c_{re}	Completion rate
c_c	Completion tasks correctly
c_w	Completion tasks wrongly
g_{re}	Testing rate
d_{re}	Discovery rate
d_e	Discover development errors
K	Dependence between development phases
k_c	Corrupt tasks-done-correctly
k_e	Corrupt development errors
r_{re}	Rework rate
r_{ru}	Rework rate of upstream phase
r_c	Redo tasks correctly
r_w	Redo tasks wrongly
τ	Time step

5.3.4 Corruption

When *Development Errors* of an upstream phase are found and corrected after starting the downstream activities, some tasks are corrupted. The amount of the tasks corrupted is the product of upstream modifications, *Dependency* (k) between development phases, and *Tasks Done* in the downstream phase (Loch and Terwiesch, 1998; Carrascosa et al., 1998; Roemer and Ahmadi 2004). In DDPM, *Rework Rate of Upstream Phase* (r_{ru}) corresponds to upstream modifications. *Tasks Done* in downstream phase is the sum of *Development Errors (DEs)* and *Tasks-done-correctly* (T_c). Consequently, *Corrupt Tasks-done-correctly* is the product of *Rework Rate of Upstream Phase, Dependency,* and *Tasks-done-correctly*. Similarly *Corrupt Development Errors* is the product of *Rework Rate of Upstream Phase, Dependency,* and *Development Errors*. Note that corruption only happens when *Tasks-done-correctly* and *Development Errors* are not equal to zero. These relationships can be presented as follows:

$k_c = r_{ru} * k * T_c$ (5.14)

$k_e = r_{ru} * k * DE_s$ (5.15)

5.3.5 Rework

Rework Rate (r_{re}) is formulated similarly to *Completion Rate*. It is the lesser of the *Average Rework Rate* (A_r), and the number of *Tasks to be Reworked* (T_{tr}) divided by the time step. *Rework Rate* is composed of *Redo Tasks Correctly* and *Redo Tasks Wrongly*. *Redo Tasks Correctly* is the product of *Rework Rate* and *Rework Quality* (R_q). *Redo Tasks Wrongly* is the rate of generating wrong tasks in the rework process. These relationships are represented by the following equations:

$r_{re} = \text{Min}\ (A_r, T_{tr}/\tau)$ (5.16)

$r_c = r_{re} * R_q$ (5.17)

$$r_w = r_{re} * (1 - R_q) \tag{5.18}$$

5.4 Validation of the model

5.4.1 Base Case

The company where the case study was conducted is a design company in Shanghai, China. The company designs mobile phones according to market and technology trends and sells the design to manufacturers, or according to customer requirements when approached by a specific customer. In order to reduce time-to-market, overlapped iterative development process is implemented in the company: downstream phase starts before the tasks in upstream phase are frozen; upstream development errors are continuously rectified according to the feedback information from downstream phases.

The project of a derivative product, which is developed based on a relatively mature architecture, is used to validate DDPM. The project started in September 2003 and was completed in May 2004. As shown in Figure 5.7 and 5.8, the development process of the project involves three development phases and each phase is composed of completion, rework, and testing activities:

1) *Concept Development*: Based on the requirements provided by the customer, the design company studied the feasibility of the product concept, refined the requirements, and defined the main features and specifications of the product. This phase is composed of four development activities: completion activity of concept development, rework activity of concept development, 3D model review (R_{C1} in Figure 5.7 and Figure 5.8), and dummy sample review (R_{C2}).

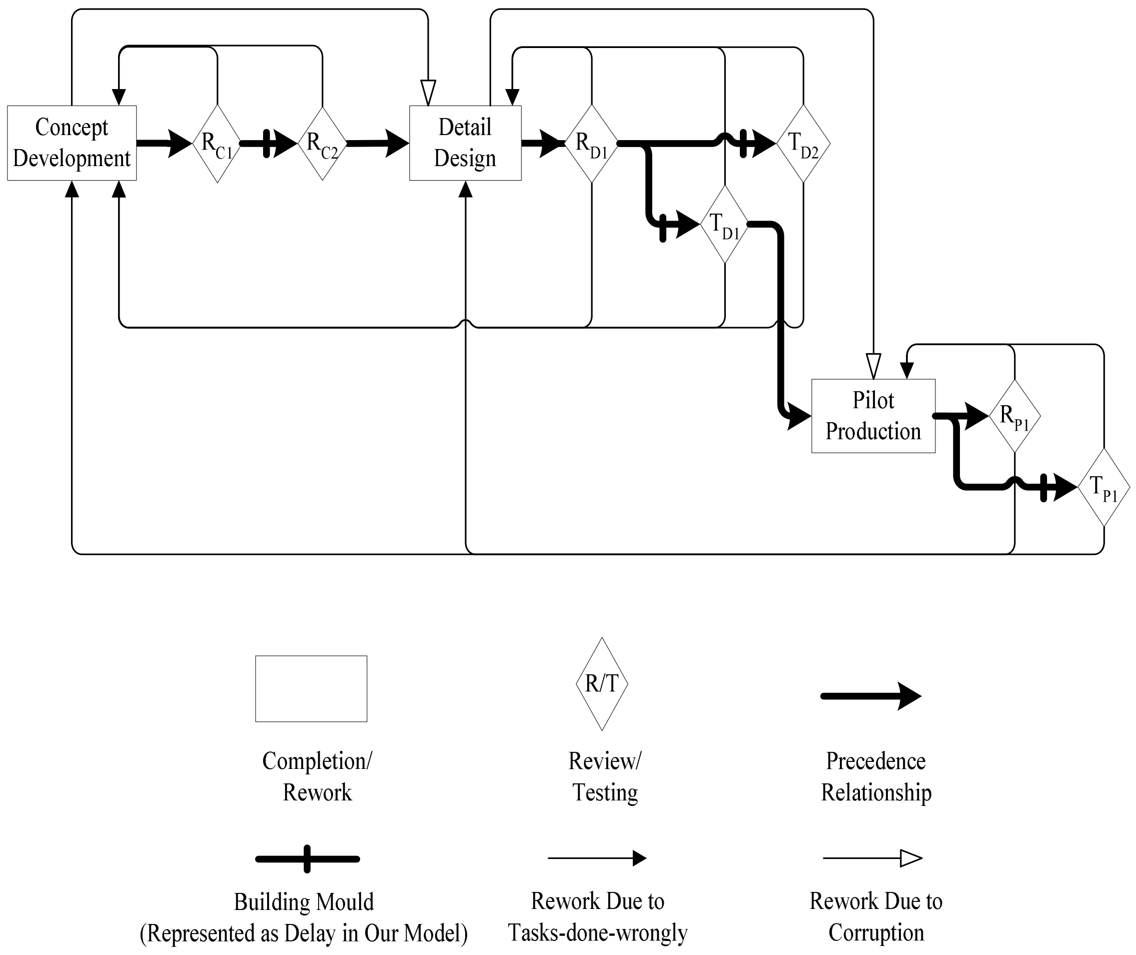


Figure 5.7 Development process of a mobile phone

2) *Detail Design:* This phase constitutes the detail design of mechanical and electronic components. After the first prototype was completed, engineers reviewed its mechanical and electronic performance to ensure compliance with initial requirements (R_{D1}), followed by detailed testing (T_{D1}). In parallel to T_{D1}, the company began making more prototypes to further test the mechanical and electronic performances of the product (T_{D2}).
3) *Pilot Production:* Pilot Production is the stage where the product design is realized as a physical product in a manufacturing plant with further testing implemented to improve the quality of the product. Half-way through the detail design phase, the

engineers began to fabricate moulds and produce products for testing. Normally, 100-500 sets of mobile phones are produced per batch. After that, the engineers started to review and solve the quality problems found in the production process (R_{P1}). At the same time, quality engineers tested the product quality and provided a report to the designers (T_{P1}). For mobile phone development, several rounds of pilot production are needed to identify potential quality problems.

		1	2	3	4	5	6	7	8	9	10
Completion and rework activity of concept development	1	■	×	×		×	×	×			
R_{C1}	2	×	■								
R_{C2}	3	×		■							
Completion and rework activity of detail design	4	×			■	×	×	×		×	×
R_{D1}	5				×	■					
T_{D1}	6				×		■				
T_{D2}	7				×			■			
Completion and rework activity of pilot production	8				×				■	×	×
R_{P1}	9								×	■	
T_{P1}	10								×		■

Figure 5.8 Information flows in the mobile phone development

5.4.2 Data Collection

In order to validate our model we collected detailed data based on historical records, such as project schedule and the quality problems found and solved over the entire period of the project. These data were double checked together with the engineers familiar with this project. The *Average Completion Rate*, *Average Testing Rate*, *Completion Quality*, and *Precedence Constraints* can be directly accessed from the historical data (Ford and Sterman, 1998; Black and Repenning, 2001). The other parameters are calculated using following equations: *Average Rework Rate* is the result of *Tasks Reworked* divided by *Rework Duration*; *Rework Quality* equals *Tasks Correctly Reworked* divided by *Tasks Reworked*; *Testing Quality* equals *Development Errors Found* divided by *Development Errors Exist* (Cooper, 1993a); *Dependency* equals *Tasks Corrupted* divided by the product of *Upstream Tasks Reworked* and *Tasks Done*. The parameter values of the project are listed in Table 5.2.

5.4.3 Model Testing

Behavior-reproduction tests (Sterman, 2004) are used to validate the model by comparing simulation results with field data for the mobile phone development project. Many tools are available to assess a model's ability to reproduce the behavior of a system. Most common are descriptive statistics to assess the point-by-point fit. Point-by-point metrics compute some measures of the error between a data series and the model output at every point for which data exist and then report the average values.

The most widely reported measure of fit is R^2, the coefficient of determination. R^2 measures the fraction of the variance in the data "explained" by the model. If the model exactly replicates the actual series, $R^2=1$; if the model output is constant, $R^2=0$. R^2 is the square of the correlation coefficient which measures the degree to which two series covary.

Table 5.2 Model inputs for the mobile phone development project

(a) Parameter values of completion & rework activities

Completion & Rework Activities	**Precedence Constraints**	**Rate (per**	**Quality**
Completion Activity of Concept Development	-	1/5	79.73%
Rework Activity of Concept Development	-	1/25	85.71%
Completion Activity of Detail Design	R_{C2} finished	1/14	63.01%
Rework Activity of Detail Design	-	1/13	83.48%
Completion Activity of Pilot Production	T_{D1} finished	1/34	78.34%
Rework Activity of Pilot Production	-	1/35	83.41%

(b) Parameter values of testing activities

Testing Activities	**Precedence Constraints**	**Rate (per day)**	**Testing Quality for Each Phase**		
			Concept Development	**Detail Design**	**Pilot Production**
R_{C1}	Concept development initially completed	2	50.00%	-	-
R_{C2}	R_{C1} finished	2	5.41%	-	-
R_{D1}	Detail design initially completed	4	14.29%	43.28%	-
T_{D1}	R_{D1} finished	1/4	50.00%	62.02%	-
T_{D2}	R_{D1} finished	1/10	28.57%	28.28%	-
R_{P1}	Pilot production initially completed	1	-	41.30%	46.24%
T_{P1}	Pilot production initially completed	1/14	-	31.72%	31.73%

(c) Dependency

Stages	**Dependency**
Dependency between Concept Development and Detail Design	2.13
Dependency between Detail Design and Pilot Production	1.63

The mean absolute error, MAE; mean absolute error as a percentage of the mean, MAE/Mean; and root mean square error, RMSE all provide measures of the average error between the simulated and actual series. MAE weights all errors linearly; RMSE weights large errors much more heavily than small ones. Both measure the error in the same units as the variable itself. MAE/Mean and RMSE/Mean provide dimensionless metrics for the error, which are easier to interpret.

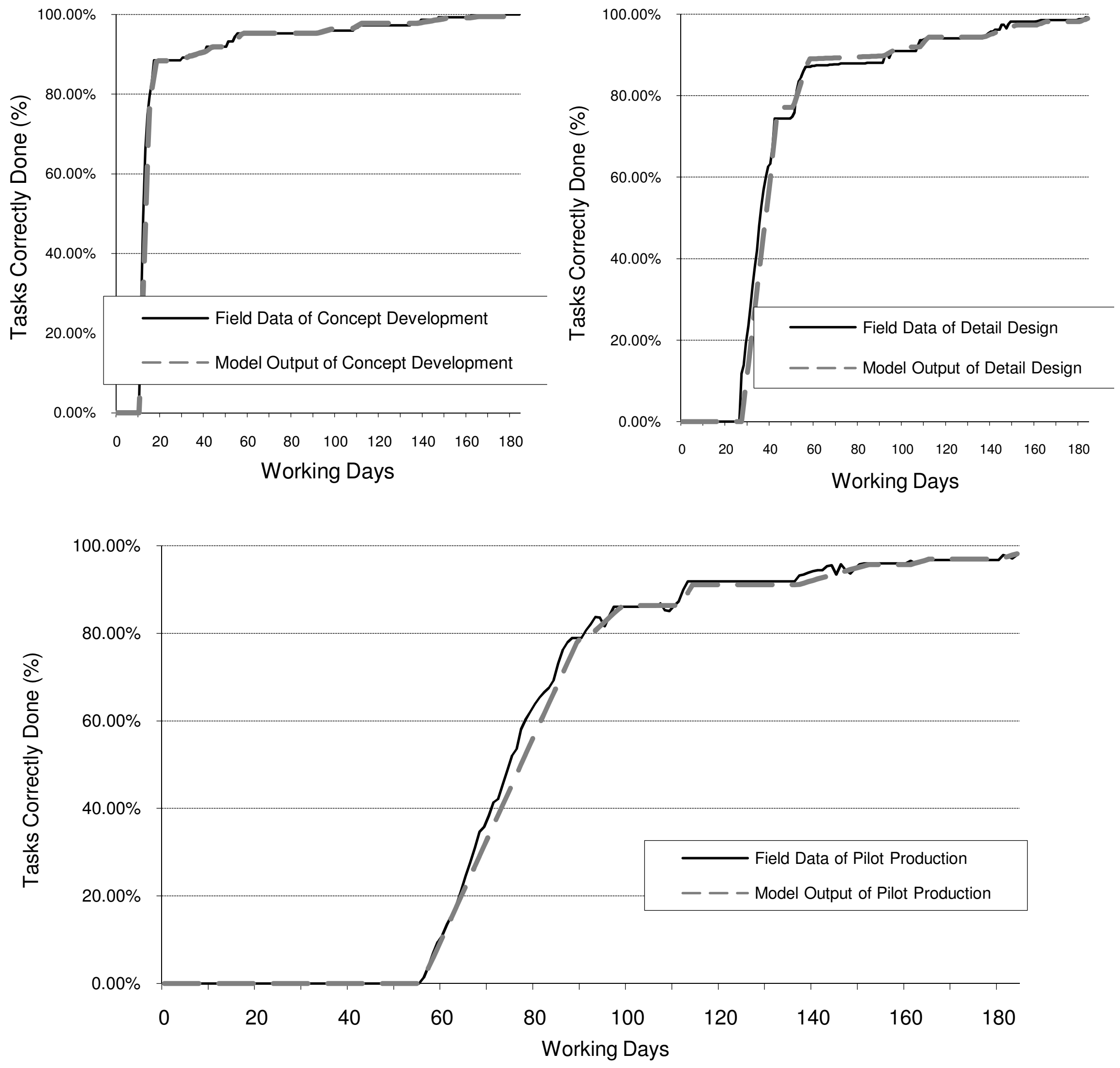


Figure 5.9 Reference mode and simulation results

Field data and model output for the three development phases are shown in Figure 5.9. The behavior patterns of DDPM for the three development phases follow closely the patterns of the field data. The error between a data series and the model output is measured by R^2, MAE/Mean, and RMSE/Mean. As shown in Table 5.3, the errors for three phases are reasonable (R^2>97%, MAE/Mean <3%, and RMSE/Mean<4%).

Table 5.3 Error statistics for assessing model fit to data

					Thiel Inequality Statistics		
Phase	**n**	**R^2 (%)**	**MAE/ Mean**	**RMSE /Mean**	**Bias (%)**	**Unequal Variation (%)**	**Unequal Covariance (%)**
Concept Development	174	97.34%	0.68%	2.39%	2.74%	58.14%	39.12%
Detail Design	158	98.88%	2.03%	3.65%	3.13%	60.26%	36.61%
Pilot Production	129	99.50%	1.97%	3.10%	32.14%	13.55%	54.31%

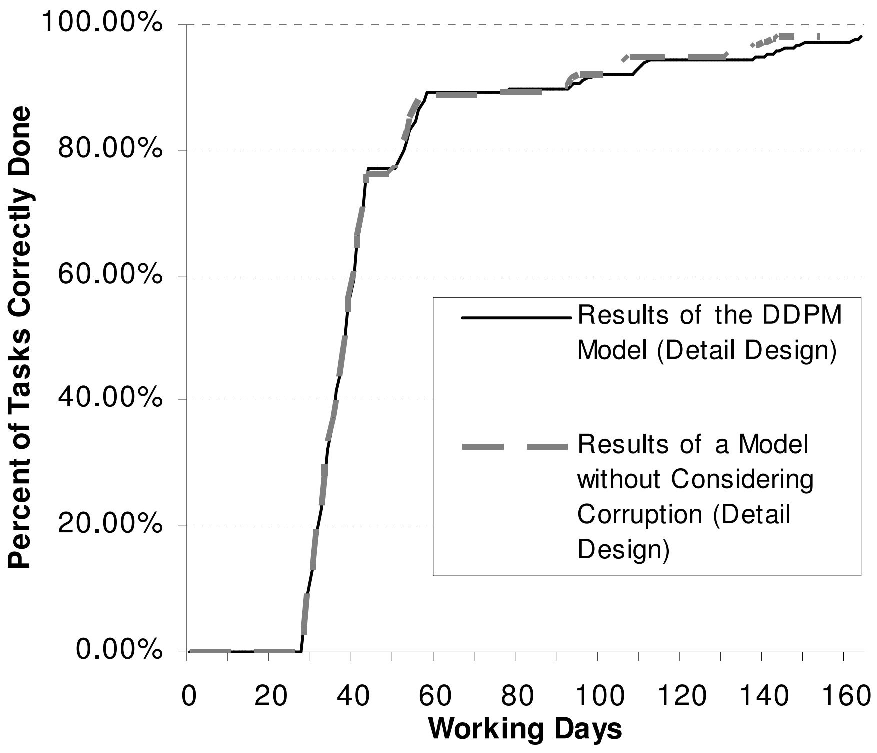
It is important to know the sources of errors as well as the total number of errors. Large errors may be due to a poor model or a large amount of random noise in the data. The total error may be large if a model of behavior in the real system is deliberately excluded as irrelevant to the model purpose. The Theil inequality statistics can help decompose the error into systematic and unsystematic components.

The Theil inequality statistics (Sterman, 1984; Theil, 1966) decompose the mean square error (MSE) into three components: Bias, Unequal Variation, and Unequal Covariation. Bias arises when the model output and data have different means. Unequal variation indicates that the variances of the two series differ. Unequal covariation means the model and data are imperfectly correlated, that is, they differ point by point. Large MAE/RMSE and large bias indicate systematic error and should lead to questions about the assumption of the model.

Partitioning the MSE using the Theil inequality statistics reveals MSE dominated by Unequal Covariance and Unequal Variation. As the errors for our model are small (R^2>97%, MAE/Mean <3%, and RMSE/Mean<4%) and Bias only accounts for a small part of MSE, this model should be accurate enough for us to show the behavior patterns of overlapped iterative development projects and study the effect of different policies on project performance.

5.5 Effect of Corruption on Project Performance

To understand the influence of *Corruption* on project performance we compare the simulation results of our model with a model without considering *Corruption*. Since no *Corruption* exists for the concept development phase, we only studied the results of detail design phase and pilot production phase (Figure 5.10) and analyzed the effect of *Corruption* on the total amount of reworked tasks and project cycle time when the project achieved the required quality with 98% of tasks correctly done. Table 5.4 shows that corruption increased about 50% of rework and 10% of project cycle time on average. This suggests that models ignoring *Rework Due to Corruption* or implicitly incorporating it into *Rework Due to Tasks-done-wrongly* will affect the accuracy of the models, and may lead to wrong managerial implications.



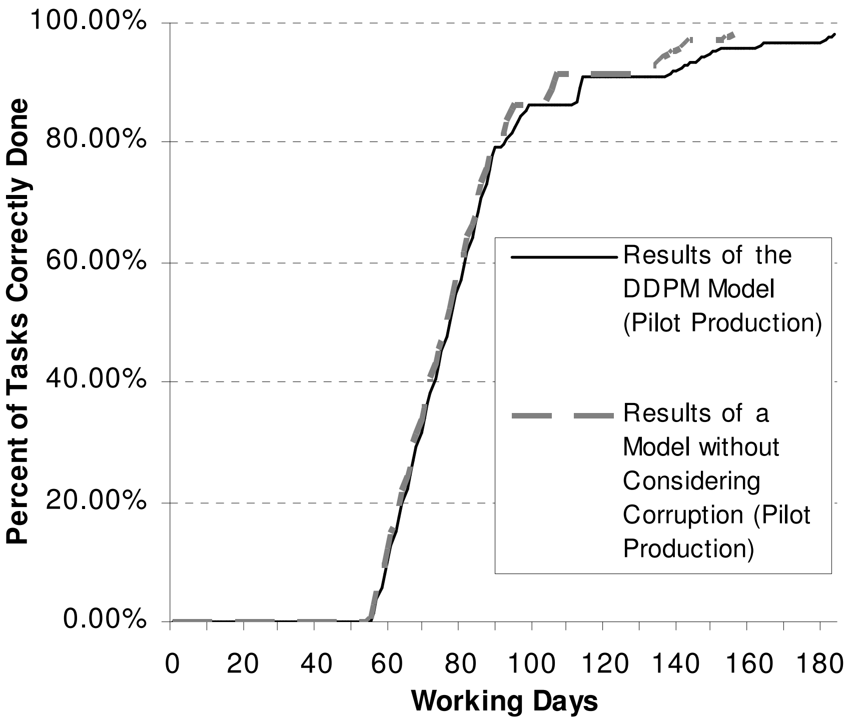


Figure 5.10 Simulating the effect of corruption

Table 5.4 Impacts of corruption on project performance

Phase		**Detail Design**	**Pilot Production**
ReworkedTasks		66.45%	64.31%
Corrupted Tasks		24.47%	38.63%
Corrupted Tasks / Reworked Tasks		36.82%	60.07%
Time Needed to Achieve 98% of Tasks Correctly Done	Without Corruption	153.42 (day)	155.54 (day)
	With Corruption	164.08 (day)	183.54 (day)
	Effect of Corruption on Project Cycle Tim	6.50%	15.14%

5.6 Policy Analysis

In order to check the applicability of our model for policy analysis, we continuously collected and analyzed the data from the projects of the design company. In 2004 we studied the product development process in the company and collected relevant data for the model. Alternative policies were analyzed subsequently. In early 2005 new policies were implemented. The results of the new policies were analyzed in 2006.

We assume that the development projects are completed when they achieve the required quality, with 98% of tasks correctly done (which is the standard currently used in the company), and try to reduce the project cycle time and development effort with different policies for different types of projects. Typically, there are three types of development projects in the company: projects with new architecture and new circuit board (Type 1 project); projects with mature architecture and new circuit board (Type 2 project); and projects with mature architecture and mature circuit board (Type 3 project). Each of these three types of projects has different development qualities, particularly in concept development phase and detail design phase. Generally the development qualities are better for projects with mature architecture and circuit board. The data used in this section were collected on 6 projects completed in 2004 (2 projects for each type). All of the data were validated by experienced engineers from Industrial Design, Mechanic Design, Hardware, Quality Control, and Production departments.

5.6.1 Overlapping between Detail Design and Pilot Production

As shown in Figures 5.7 and 5.8, the initial completion of detail design generates preliminary information for pilot production and then the preliminary information will be modified through testing and rework. Pilot production can start when preliminary information is available. However it will incur *Rework Due to Corruption*. Pilot production can also start after most of the modifications are done. Therefore four alternative overlapping policies are considered: start pilot product after T_{D2} (represented as overlapping policy 1 (O_1) in Figure 5.11); start pilot production after T_{D1} (O_2); start pilot production after R_{D1} (O_3); and start pilot production immediately after detail design (O_4). In 2004, the standard process for the company was to start pilot production after T_{D1}. However, is this level of overlapping suitable for all of the projects in the company? In order to answer it, we tested the influence of different overlapping policies for three types of projects. As can be seen in Figure 5.11, more rework occurs as the degree of overlapping increases. This may explain why overlapped product development does not always work as predicted. There is a trade-off between the time reduced because of overlapping and the cost and time increased due to rework.

Rework changes more significantly in Type 1 project than the change in the other projects, as indicated by the steeper slope of the line. In 2004, O_2 was implemented in the company. This policy is suitable for Type 1 project, because the application of O_3 or O_4 would increase rework seriously. However, O_3 is better for Type 2 and Type 3 projects, because project cycle time can be reduced with little expense of rework. Theoretically, O_4 can also be implemented for projects with mature architecture. However, according to the experience of the engineers from the company, O_4 may increase the risk of damaging the hard mould for pilot production, causing a tremendous increase in cost. For example, the size of handset housing affects a large number of related parts, and a change of it means building new hard moulds. Pilot production should only start after important specifications have been properly reviewed and confirmed. Consequently, the overlapping policy should be adjusted: O_2 can be used for type 1 project, and O_3 can be applied for type 2 and type 3 projects.

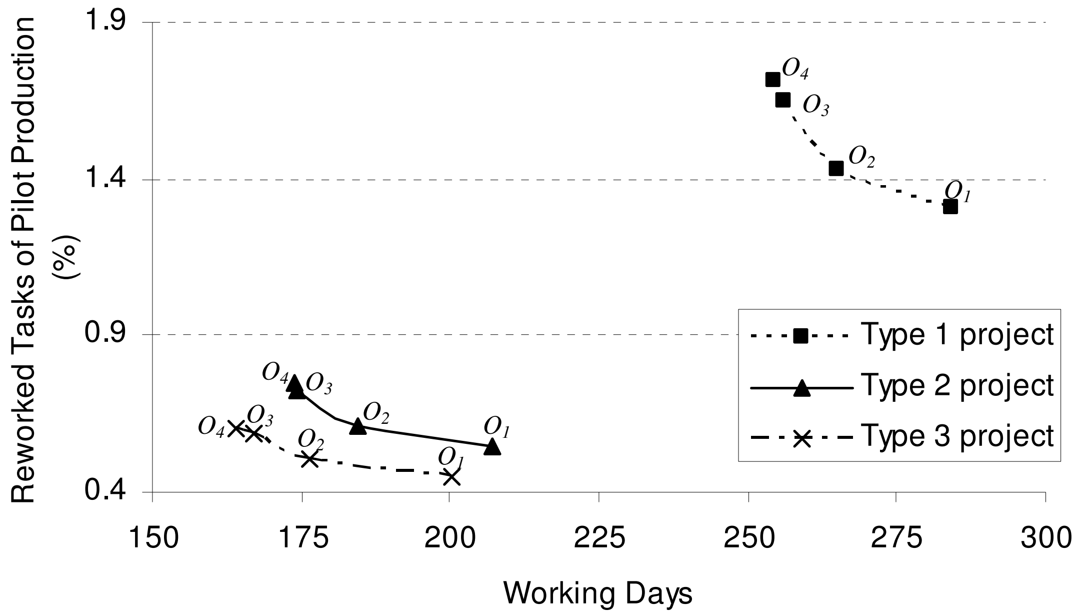


Figure 5.11 Project performance with different levels of overlapping between detail design and pilot production

5.6.2 Overlapping in Pilot Production

Table 5.5 Project performance with different levels of overlapping in pilot production

Performance	**Development Projects**	**Current Policy**	**New Policy**	**Performance Improvement (%)**
Reworked Tasks of Pilot Production (%)	Type 1 Project	1.4274	1.5082	-5.66%
	Type 2 Project	0.6087	0.6102	-0.25%
	Type 3 Project	0.5014	0.5026	-0.24%
Project Cycle Time (working day)	Type 1 Project	265.25	234.75	11.50%
	Type 2 Project	184.5833	175.125	5.12%
	Type 3 Project	176.5833	165.9167	6.04%

For mobile phone development, usually several (2-4) rounds of pilot production are needed. In 2004, the policy for the company was to start the next round of pilot production immediately after R_{P1} if T_{P1} had not begun. For example, in the base case R_{P1} was finished in 6 days before T_{P1} started. Using the old policy, engineers started the second round of pilot production after R_{P1}. Some engineers questioned whether it was worthwhile to do more rounds of pilot production to reduce project cycle time because pilot production is costly (as typically 100-500 sets of mobile phones were produced just for testing). It seems obvious that this policy can reduce project cycle time. However, the simulation results show that this policy not only increases cost but also increases project cycle time (Table 5.5). The policy aims to reduce project cycle time but in reality the opposite happened because quality problems found in T_{P1} could not be corrected in the second round of pilot production. Therefore more rounds of pilot production might be needed to achieve the required project quality. Consequently, the suggested new policy is that the next round of pilot production should start after both R_{P1} and T_{P1}, whether the projects are urgent or not.

5.6.3 Evaluating Activity Criticality

The investment policies can be evaluated by activity criticality, which is the sensitivity of project performance to the investment for improving activity quality and duration. The investment needed and the resulting activity quality and duration could be estimated according to the experience of the managers or by benchmarking. Based on these data we can re-analyze the project performance and compare the investment with the performance improved.

Table 5.6 Project performance with original and improved activity duration

Performance	Development Projects	Current Machine	New Machine	Performance Improvement (%)
Reworked Tasks of Pilot Production (%)	Type 1 Project	1.4274	1.3244	7.22%
	Type 2 Project	0.6087	0.5592	8.13%
	Type 3 Project	0.5014	0.4637	7.52%
Project Cycle Time	Type 1 Project	265.25	259.5833	2.14%
	Type 2 Project	184.5833	180.3333	2.30%
	Type 3 Project	176.5833	173.25	1.89%

When we were doing the case study in the company, the managers were considering introducing a prototyping machine. Prototypes are used to find and solve *Development Errors* of detail design before expensive pilot production starts. The first prototype was outsourced at that time and it took about one week. The rest of the prototypes were done through soft tooling, which was time consuming. The new prototyping machine could reduce by 2 working days for the first prototype, and 21 working days for the other prototypes, but it required a large amount of financial investment. In order to study if it was worthwhile to acquire this new machine, we compared the performance of the projects using current practice with the performance using the new prototyping machine (Table 5.6). It turned out that the reduction of 23 working days with the use of the new machine could only reduce the project cycle time by about 2%. However, it could significantly reduce rework by about 7.6% on average. The reason is that T_{D1} and T_{D2} are not in the critical path of the project. These are the activities to find the quality problems of concept development and detail design. Finding the quality problems earlier can reduce the rework caused by corruption in the project. According to the simulation results, the number of projects done in the company each year, and the cost of each project, we found that the cost saving is greater than the investment involved. We thus suggested the company to invest in the prototyping machine.

5.6.4 Application Result

We simulated the product development process again based on the new policies for overlapping and investment. The result shows that if the new overlapping policies are adopted and the prototyping machine is invested, the company can reduce the project cycle time by about 12% without significantly affecting the percentage of reworked tasks.

All of the policies discussed above were accepted by the company, because we not only analyzed the effect of different policies, but also explained the root causes of the results. Comparing the projects completed before June 2004 to the projects finished in the first half of the year 2005 when the new policies had been implemented, we found that the average project cycle time was reduced by about 30%. Except for the new policies mentioned above, the average rate of pilot production and testing (T_{D1}, T_{D2}, and T_{P1}) was improved because of the introduction of new technologies and the adoption of other development policies. There is no significant difference between the qualities of development activities for the projects developed in 2004 and in 2005. The reason is that the experience of the engineers is not significantly changed from 2004 to 2005. Some experienced engineers left the company and recruits were employed. The average working experience of the engineers was about 5 years. Using the new input data, the model output matched closely with the average project cycle time of the projects finished in 2005. It further validated our model and showed that DDPM is a useful tool for analyzing overlapped iterative product development.

5.7 Conclusion

The ability to expeditiously develop products has been accredited as a critical factor for the survival of many companies (Carrillo and Franza, 2006). Overlapping of development activities is commonly regarded as the most promising strategy to reduce project cycle time. However, overlapping interrelated activities based on preliminary information can be costly (Roemer et al., 2000). Therefore, some researchers have developed models to analyze overlapped and sequentially dependent product development process (e.g. Krishnan et al., 1997; Loch and Terwiesch, 1998; Roemer et al., 2000). The effect of the "upstream information evolution" on the rework of the downstream phase is studied in these models. We extend previous research by developing a SD model for overlapped iterative product development.

The model presented in this chapter has high "face validity" (Law and Kelton, 1991; Smith and Morrow, 1999) because it is developed based on the accepted framework for overlapped product development (e.g. Krishnan et al., 1997; Loch and Terwiesch, 1998; Roemer et al., 2000), in depth case study of the modeled system, and System Dynamics methodology (Black and Repenning, 2001; Cooper, 1980; Ford and Sterman, 1998; Joglekar and Ford, 2005; Williams et al., 1995; Williams et al., 2003) which has been theoretically and practically validated. Furthermore, we validated the model through comparing the simulation results with real data of a project done in a mobile phone development company. The successful application of the new policies further implies the usefulness of our model. Although more detailed examination may be needed, the current level of validity is comparable with models in the related literature (Abdelsalam and Bao, 2006; Browning and Eppinger, 2002; Cho and Eppinger, 2005; Ford and Sterman 1998; Smith and Morrow, 1999).

The contribution of this simulation model is threefold. Firstly, development projects are usually complex and iterative. Although previous research has advanced our understanding of overlapped processes with sequential dependence, there is still a lack of tools to study overlapped iterative processes. Our simulation model provides management a useful tool to predict the effect of different overlapping policies on project performance for overlapped iterative product development. Secondly, the introduction of new technologies and/or new machines can help to reduce the time required to complete various development activities and reduce the probability of making errors. Our model provides a way to evaluate these possibly substantial capital investments by estimating the resulting improvement of project performance. Thirdly, our case study and simulation results show that some insights drawn by previous models for overlapping development stages with sequential dependence are still applicable in overlapped iterative product development: (1) overlapping policies are significantly affected by project uncertainty, thus different overlapping strategies can be applied for the projects with different uncertainty levels; (2) The total amount of downstream rework can be reduced by increasing evolution speed, which is achieved through increasing testing rate in our case.

Several aspects of the model presented in this chapter merit further examination. Firstly, more cases are needed to validate the model. Future research can test the broader application of the DDPM by applying it to other overlapped iterative NPD projects. Such tests will provide a basis for the abstraction of more general dynamic lessons for development process improvement. Secondly, the effective allocation of development resources, such as manpower and equipment, is difficult due to the complexity of overlapped product development process. Therefore, our model may be further developed to analyze resource allocation policies for overlapped iterative NPD projects. Finally, overlapping of development activities incurs additional rework. Spending resource for the rework for one project inevitably causes delay in other projects. Therefore, we may need to extend our model to explore suitable policies for managing multiple projects.

CHAPTER 6

CONCLUSIONS AND FUTURE STUDY

The models for managing concurrent processes with different information dependencies and communication costs have been presented through Chapter 3 to Chapter 5. In this chapter, concluding remarks are given to summarize the main findings and contributions. The limitations of the models and the possible research areas for future studies are also highlighted.

6.1 Introduction

Successful new product development is critical for the survival of many companies. Product development literature often provides open-loop, single-link linear relationships which are usually lists of simplified rules for managing projects. These universal solutions are often not applicable in practice because each firm has a unique set of constraints and requirements, and does not usually carry out the same project twice (Smith and Reinertsen, 1998). For example, in the literature of concurrent engineering, some researchers (e.g. Imai et al., 1985; Takeuchi and Nonaka, 1991; Clark and Fujimoto, 1991; Wheelwright and Clark, 1992) propose that product development processes can be accelerated by concurrent execution of development stages. The others (e.g. Eisenhardt and Tabrizi, 1995; Terwiesch and Loch, 1999) argue that concurrent engineering is not applicable to all product development projects. Although these studies have contributed to theoretical understanding, they are still not applicable in practice. No method is proposed to help management make decisions on how to apply concurrent engineering to a certain project.

Therefore, the focus of this thesis can no longer be whether or not to apply CE but to probe more deeply. This research develops two analytical models and one simulation model for different concurrent processes addressing two questions: (1) how to determine the optimal overlapping degree between consecutive development stages; (2) how to arrange the information exchanges for coordinating the upstream changes. In addition, the analytical model in Chapter 3 also addresses the functional interaction policy, and the simulation model in Chapter 5 discusses the effect of testing and development qualities on project cycle time and development cost.

6.2 Contributions of this Study

This thesis sets out to develop models for managing NPD projects, and to derive management insights to help in structuring concurrent product development processes. It explores the effects of project properties and development policies on NPD performance. Three models for the projects with different information dependencies have been developed to derive appropriate overlapping policies, communication strategies, and functional interaction level. In addition, comparisons are made specifically between sequential and CE processes. The conditions under which one process outperforms the other are investigated. This work has both theoretical and methodological implications.

6.2.1 Contributions to Theory

While some researchers have developed models to analyze concurrent product development process, most of these models are developed based on the assumption that the rework probability or the amount of rework is known. However, the question of how to estimate the probability of rework has not been rigorously addressed by researchers (Roemer and Ahmadi, 2004; Jun et al., 2005). Consequently, these models are not applicable in practice. The models introduced in this thesis are helpful for management to derive appropriate development policies by the data available in practice. This was done based on the concepts of evolution and sensitivity (Krishnan et al. 1997) and on practical studies of concurrent product development processes, following an information processing view of product development processes. The research has produced the following important analytical results, management insights, and algorithms, categorized under the models for different concurrent processes.

For sequentially dependent processes with low communication cost, firstly, the optimal overlapping policies are presented. Then, the impact of different project properties (such as dependency between development stages, rework rate, and opportunity cost of time) on overlapping policies is proposed. Finally, the thesis derives the optimal functional interaction strategies under the condition that optimal overlapping is followed. This model was used to examine the development policies in a handset design company.

For sequentially dependent processes with high communication cost, an algorithm is proposed to derive the optimal overlapping and communication strategies for the general case. Then the linear evolution case is further discussed and a more efficient algorithm for the linear evolution case is presented. The model is applied to improve the refrigerator development process in an electronics company. Moreover, some guidelines for structuring concurrent processes are provided.

For interdependent process, this work presents a new product development (NPD) process model, termed Dynamic Development Process Model (DDPM). Although System Dynamics does not guarantee an optimal solution for a product development project, it can include more factors of reality. Because of this, SD has been widely and successfully applied to study product development projects in practice (e.g. Cooper, 1980; Ford and Sterman, 1998; Black and Repenning, 2001; Joglekar and Ford, 2005). The SD model developed in this thesis extends previous work (e.g. Krishnan et al., 1997; Loch and Terwiesch, 1998; Joglekar et al., 2001) through modeling iterative product development process with multiple phases, considering the rework caused by different causes, and investigating the trade-offs between project cycle time, project quality, and development effort. The model has been validated with data from a mobile phone development project. The DDPM was employed to identify appropriate policies for the overlapped iterative projects in the case study company. These identified policies were implemented in the company and led to marked improvement in project performance, thus demonstrating the viability of the model.

The findings of the models generally agree with empirical results. As such, they seem to provide a reasonable prediction of performance and can therefore be considered to be a good managerial tool to help design a process in terms of overlapping, communication, and functional interaction. The applicability of the models is illustrated with three case studies in consumer electronics companies.

To date, no existing work has produced useful guidelines on how to design and manage concurrent processes in detail. Our study is presumably the first study that has systematically analyzed concurrent processes. Furthermore, different models for the projects with different information dependency and communication cost are developed. Thus, the models developed in this thesis are more able to represent practical situations than previously developed models. The study of concurrent product development is therefore a significant contribution to the existing research.

6.2.2 Contributions to Methodology

Contributions to methodology mainly refer to the use of System Dynamics methodology to study concurrent product development processes. SD has been widely used to model NPD processes in 1980s and 1990s. However, previous models can not explicitly simulate the dynamic properties of concurrent processes. This limits the usability of SD models, as CE has already been applied in most NPD projects (Siemieniuch and Sinclair, 1999). Our DDPM model describes a structure to describe the dynamic relationship between overlapped stages. Together with previous SD model, our model shows that SD can become a powerful tool to study concurrent product development.

The SD methodology is also instrumental in broadening the scope of the study of concurrent product development, which was limited when using mathematical methodology. As a result, one of the most significant contributions of this thesis is that the outcomes of concurrent execution of two development stages could be tied to macro-process performance in a product development model. Whereas most models in the existing literature have produced important findings about the concurrent execution of two development stages, the SD model, which integrates these process characteristics into more comprehensive models, allows us to check the combined effect of different development policies, such as overlapping and testing policies, on project performance.

6.3 Limitations

There are several limitations for this study that restrict the extent to which the findings can be generalized. Firstly, although the models are built on previous literature and validated on three case studies in consumer electronics industry, it is important to test and improve them through real applications in other industries since the NPD processes may be different.

Secondly, the models assume that all team members share common goals in a project. Though this is a common assumption in NPD models and is true to most NPD projects, in a cross-company project, functional teams tend to have goals that differ from one another. Goal incongruity may increase the complexity of product development and affects the choice of development policies.

Thirdly, our simulation model does not guarantee an optimal solution to manage overlapped product development. This is a limitation of most simulation models incorporating iteration in complex projects (Cho and Eppinger, 2005).

Fourthly, we assume the opportunity cost of time is constant. However, although the opportunity cost should be constant in short run, it may change in long run. It implies that the opportunity cost of time should be investigated in detail, if the application of CE changes project launch time significantly.

Finally, in this thesis, we focus on process related options (overlapping, functional interaction, and information exchange) to reduce cycle time. However, cycle time can also be reduced through a focus on product design related options such as module based platform development. It should be promising to investigate the process and design related options simultaneously.

6.4 Future Work

There are several opportunities that exist to extend the work presented in this thesis, as well as to develop new areas of research. Avenues for future work are now discussed.

6.4.1 Application of the Models

Researchers can test the broader application of the models by applying them to other development projects and processes. Empirical research focusing on the issues in this thesis would be useful to compare to the models' results. A continued effort towards studying NPD process performance using the methodologies and toolsets presented in this thesis should contribute increasingly to understand how NPD can be successfully managed in order to meet organizational goals. It will provide a basis for the abstraction of more general dynamic lessons for development process improvement.

6.4.2 Communication Technologies

The results of this thesis show that communication efficiency can significantly affect development policies and project performance. Modeling the use of various communication technologies would undoubtedly help us to check the root cause of communication efficiency. Considering the growing use of communication technologies, it becomes valuable to track their roles and impact on project performance.

6.4.3 Knowledge Management in Product Development

It may be useful in the future to focus on knowledge management in NPD. Modeling the effect of knowledge sharing would be useful and necessary. Sharing knowledge of different functions and projects might minimize the need to rework, and may even show how multiple projects can be arranged. In fact, the effect of functional interaction is studied in the first model and it can also be studied by the other two. For knowledge sharing in different projects, no existing model has been able to deal with it.

6.4.4 Multiple Project Interaction

The present models consider only one project, though there may be other projects that may interact with it. Each project is vying for the limited resources within the organization, which places constraints on all of them (Scott, 1997; Gerwin and Barrowman, 2002). This may result in one project not getting enough resources when other projects are considered, making it difficult to dedicate resources to a project.

In our case studies in consumer electronics industry, we observed that team members were sometimes being pulled from one project to another. The resulting effect was span time prolongation and effort increase due to the "warm-up" period required for the engineers to start up again where they had left off. This can seriously constrain a project from achieving low cycle time and cost. Investing it would help to identify the effect of multiple-project interactions, the serious impact of the loss of a team member, and the importance of resource availability to an organization. This can also highlight the difference in performance between projects that have dedicated team members, and those whose team members are subject to fluctuations.

6.4.5 Testing

As was discussed in Chapter 5, review or testing is arranged at the end of each stage. For new product development, review and testing usually accounts for the most of development time and cost. In our models the testing strategies are assumed to be predetermined.

Because of the importance of testing, some researchers are attempting to identify testing strategies that optimize product development performance (Loch et al. 2001; Thomke and Bell, 2001). When testing activities are overlapped with each other, though span time is reduced, the effect of learning is lost, as compared to sequential testing (Thomke, 1998). Loch et al. (2001) suggest that when testing is costly, it should be performed sequentially, and that when the test activity is slow, executing it in parallel will reduce project cycle time.

Although researchers have paid attention to the testing activities in one stage/phase, so far, no work has investigated the relationship between testing activities in different development stages. For example, in mobile phone development, engineers can do one round of prototype testing, or many rounds of prototype testing. Spending more time in prototype testing will reduce the potential problems in pilot production. Therefore, the project cycle time may be reduced. Then, how to balance the testing activities in concept development, detail design, and pilot production? It should be valuable to investigate it in detail.

REFERENCES

Abdel-Hamid, T., 1984. The dynamics of software development project management: An integrative system dynamics perspective. Doctoral thesis. Massachusetts Institute of Technology. Cambridge, MA.

Abdelsalam, H.M.E., Bao, H.P., 2006. A simulation-based optimization framework for product development cycle time reduction. IEEE Transactions on Engineering Management 53 (1), 69-85.

Adler, P.S., 1995. Interdepartmental interdependence and coordination: the case of the design / manufacturing interface. Organization Science 6 (2), 147-167.

Ahmadi, R., Roemer, T.A., Wang, R.H., 2001. Structuring product development processes. European Journal of Operational Research 130 (3), 539-558.

Ahmadi, R., Wang, H., 1999. Managing development risk in product design processes. Operations Research 47 (2), 235-246.

Ali, A., 1994. Pioneering versus incremental innovation: Review and research propositions. The Journal of Product Innovation Management 11 (1), 46-61.

Badiru, A.B, 1993. Quantitative Models for Project Planning, Scheduling, and Control. Quorum Books, Westport, Conn.

Belhe, U., Kusiak, A., 1996. Modeling relationships among design activities. Journal of Mechanical Design 118 (4), 454-460.

Bhuiyan, F., 2001. Dynamic models of concurrent engineering processes and performance. Ph.D. dissertation, Department of Mechanical Engineering, McGill University, Montreal.

Bhuiyan, N., Gerwin, D., Thomson, V., 2004. Simulation of the new product development process for performance improvement. Management Science 50 (12), 1690-1703.

Bhuiyan, N., Thomson, V., Gerwin, D., 2006. Implementing concurrent engineering. Research Technology Management 49 (1), 38-43.

Black, L.J., Repenning, N.R., 2001. Why firefighting is never enough: Preserving high-quality product development. System Dynamics Review 17 (1), 33-62.

Blackburn J., 1991. New Product Development: The New Time Wars. In Time-Based Competition: The Next Battleground in American Manufacturing, ed. by J. Blackburn, 121-163. Business One Irwin, Homewood.

Browning, T.R., 2001. Applying the Design Structure Matrix to system decomposition and integration problems: a review and new directions. IEEE Transactions on Engineering Management 48 (3), 292-306.

Browning, T.R., Eppinger, S.D., 2002. Modeling impacts of process architecture on cost and schedule risk in product development. IEEE Transactions on Engineering Management 49 (4), 428-442.

Carrascosa, M., Eppinger, S.D., Whitney, D.E., 1998. Using the design structure matrix to estimate product development time. ASME Design Engineering Technical Conferences, Atlanta, GA.

Carrillo, J.E., Franza, R.M., 2006. Investing in product development and production capabilities: The crucial linkage between time-to-market and ramp-up time. European Journal of Operational Research 171 (2), 536-556.

Chakravarty, A.K., 2001. Overlapping design and build cycles in product development. European Journal of Operational Research 134 (2), 392-424.

Chen, C.H., Khoo, L.P., Jiao, L., 2004. Information deduction approach through quality function deployment for the quantification of the dependency between design tasks. International Journal of Production Research 42 (21), 4623-4637.

Chi, J.C., 2001. The knowledge evolution framework: a knowledge management perspective on the impact of knowledge segregation on product development projects. Doctoral thesis. Massachusetts Institute of Technology. Cambridge, MA.

Cho, S., Eppinger, S.D., 2005. A simulation-based process model for managing complex design projects. IEEE Transactions on Engineering Management 52 (3), 316-328.

Clark, K.B., Fujimoto, T., 1991. Product Development Performance Strategy, Organization and Management in the World Auto Industry. Harvard Business School Press, Boston, MA.

Clausing, D., 1994. Total Quality Development: A Step-By-Step Guide to World-Class Concurrent Engineering. ASME Press, New York.

Cohen, M.A., Eliashberg, J., Ho, T-H., 1996. New product development: the performance and time-to-market tradeoff. Management Science 42 (2), 173-186.

Cooper, K.G., 1980. Naval ship production: A claim settled and a framework built. Interface 10 (6), 20-36.

Cooper, K.G., 1993a. The rework cycle: Benchmarks for the project manager. Project Management Journal 24 (1), 17-22.

Cooper, K.G., 1993b. The rework cycle: How projects are mismanaged. PMNETwork, February, 5-7.

Cooper, K.G., 1993c. The rework cycle: How it really works…and reworks…. PMNETwork, February, 25-28.

Cooper, R.G., 1994. Third generation of new product processes. Journal of Production Innovation Management 11 (1), 3-14.

Cordero, R., 1991. Managing for speed to avoid product obsolescence: A survey of techniques. The Journal of Product Innovation Management 8 (4), 283-294.

Datar, S., Jordan, C., Kekre, S., Rajiv, S., Srinivasan, K., 1997. New product development structures and time-to-market. Management Science 43 (4), 452-464.

Dean, J.W.Jr., Susman, G.I., 1991. Organizing for manufacturable design: managing product life cycles from start to finish. Harvard Business Review Paperback. Cambridge, MA.

Eisenhardt, K.M., Tabrizi, B.N., 1995. Accelerating adaptive processes: Product innovation in the global computer industry. Administrative Science Quarterly 40 (1), 84-110.

Eppinger, S.D., Whitney, D.E., Smith, R.P., Gebala, D.A., 1994. A model-based method for organizing tasks in product development. Research in Engineering Design 6 (1), 1-13.

Ford, D.N., Hou, A., Seville, D., 1993. An exploration of systems product development at Gadget, Inc.. Working paper D-4460. System Dynamics Group. Sloan School of Management. Massachusetts Institute of Technology. Cambridge, MA.

Ford, D.N., 1995. The dynamics of project management: an investigation of the impacts of project process and coordination on performance. Doctoral thesis. Massachusetts Institute of Technology. Cambridge, MA.

Ford, D.N., Sterman, J.D., 1998. Dynamic modeling of product development processes. System Dynamics Review 14 (1), 31-68.

Ford, D.N., Sterman, J.D., 2003a. Overcoming the 90% syndrome: Iteration management in concurrent development projects. Concurrent Engineering: Research and Applications 11 (3), 177-186.

Ford, D.N., Sterman, J.D., 2003b. The liar's club: Concealing rework in concurrent development. Concurrent Engineering: Research and Applications 11 (3), 211-219.

Ganapathy, B.K., Goh, C.H., 1997. A hierarchical system of performance measures for concurrent engineering. Concurrent Engineering Research and Applications 5 (2), 137-143.

Gerwin, D., Barrowman, N.J., 2002. An evaluation of research on integrated product development. Management Science 48 (7), 938-953.

Gil, N., Tommelein, I.D., Beckman, S., 2004. Postponing design processes in unpredictable environments. Research in Engineering Design 15 (3), 139–154.

Golenko-Ginzburg, D., Gonik, A., 1996. On-line control model for cost simulation network projects. Journal of the Operational Research Society 47 (2), 266-283.

Griffin, A., 1997. PDMA research on new product development practices: Updating trends and benchmarking best practices. The Journal of Product Innovation Management 14 (6), 429 – 458.

Grupp, H., Maital, S., 2001. Managing New Product Development and Innovation: A Microeconomic Toolbox. Edward Elgar, Cheltenham, UK.

Ha, A.Y., Porteus, E.L., 1995. Optimal timing of reviews in concurrent design for manufacturability. Management Science 41 (9), 1431-1447.

Haberle, K.R., Burke, R.J., Graves, R.J., 2000. A not on measuring parallelism in concurrent engineering. International Journal of Production Research 38 (8), 1947-1952.

Hauser, J., Tellis, G.J., Griffin A., 2006. Research on innovation: A review and agenda for marketing science. Marketing Science 25 (6), 687-717.

Hayes, R.H., Wheelwright, S.C., Clark K.B., 1988. Dynamic Manufacturing, Creating the Learning Organization. New York: The Free Press.

Helms, R., 2002. Product data management as enabler for concurrent engineering. Ph.D. dissertation, Department of Technology Management, Technical University Eindhoven, The Netherlands.

Helms, R., 2004. Framework for releasing preliminary information in product development. Advanced Engineering Informatics 18 (4), 231-240.

Henderson, R.M., Clark, K.B., 1990. Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. Administrative Science Quarterly 35 (1), 9-30.

Homer, J., Sterman, J., Greenwood, B., Perkola, M. 1993. Delivery time reduction in pulp and paper mill construction projects: A dynamic analysis of alternatives. Proceedings of the 1993 International System Dynamics Conference. Monterey Institute of Technology. Cancun, Mexico.

Imai, K., Nonaka, I., Takeuchi, H., 1985. Managing the new product development process: how the Japanese companies learn and unlearn in The Uneasy Alliance. Clark, K.B., Hayes, R.H., Lorenz, C., eds. Harvard Business School Press, Cambridge, MA.

Jessen, S.A., 1990. The motivation of project managers, a study of variation in Norwegian project managers' motivation and demotivation by triangulation of methods. Doctoral thesis. The Henley Management College and Brunel University.

Joglekar, N.R., Yassine, A.A., Eppinger, S.D., Whitney, D. E., 2001. Performance of coupled product development activities with a deadline. Management Science 47 (12), 1605-1620.

Joglekar, N.R., Ford, D.N., 2005. Product development resource allocation with foresight. European Journal of Operational Research 160 (1), 72-87.

Jun, H.B., Ahn, H.S., Suh, H.W., 2005. On identifying and estimating the cycle time of product development process. IEEE Transactions on Engineering Management 52 (3), 336-349.

Keisler H.J., 1986. Elementary Calculus: An Infinitesimal Approach. Prindle, Weber & Schmidt, Boston.

Krishnan, V., 1996. Managing the simultaneous execution of coupled phases in concurrent product development. IEEE Transactions on Engineering Management 43 (2), 210-217.

Krishnan, V., Eppinger, S.D., Whitney, D.E., 1997. A model-based framework to overlap product development activities. Management Science 43 (4), 437-451.

Krishnan, V., Ulrich, K.T., 2001. Product development decisions: A review of the literature. Management Science 47 (1), 1-21.

Law, A.M., Kelton, W.D., 1991. Simulation Modeling and Analysis. McGraw-Hill, Second ed., New York.

Lawson, M., Karandikar, H.M., 1994. A survey of concurrent engineering. Concurrent Engineering: Research and Applications 2 (1), 1-6.

Ledet, W., Himmelblau, D., 1970. Decomposition procedures for the solving of large scale systems, Advances in Chemical Engineering 8, 185-224.

Leifer, R., McDermott, C.M., O'Connor, G.C., Peters, L.S., Rice, M., Veryzer, R.W., 2000. Radical Innovation: How Mature Companies Can Outsmart Upstarts. Harvard Business School Press, Cambridge, MA.

Liker, J.K., Sobek, D.K., Ward, A.C., Cristiano, J.J., 1996. Involving suppliers in product development in the United States and Japan: evidence for set-based concurrent engineering. IEEE Transactions on Engineering Management 43 (2), 165-178.

Lin, J., Chai, K.H., Wong, Y.S., Brombacher, A.C., 2008. A dynamic model for managing overlapped iterative product development. European Journal of Operational Research 185 (1), 378-392.

Loch, C.H., Terwiesch, C., 1998. Communication and uncertainty in concurrent engineering. Management Science 44 (8), 1032-1048.

Loch, C.H., Terwiesch, C., Thomke, S., 2001. Parallel and sequential testing of design alternatives. Management Science 47 (5), 663-678.

McDermott, C.M., 1999. Managing radical product development in large manufacturing firms: A longitudinal study. Journal of Operations Management 17 (6), 631–644.

Meixell, M.J., Nunez, M., Talalayevsky, A., 2006. Activity structures in a project-based environment: A coordination theory perspective. IEEE Transactions on Engineering Management 53 (2), 285-296.

Minderhoud, S., Fraser, P., 2005. Shifting paradigms of product development in fast and dynamic markets. Reliability Engineering and System Safety 88 (2), 127-135.

Moder, J.J., Phillips, C.R., Davis, E.W., 1983. Project Management with CPM, PERT, and Precedence Diagramming. Van Nostrand Reinhold, New York.

Morelli, M.D., Eppinger S.D., Gulati R.K., 1995. Predicting technical communications in product development organizations. IEEE Transactions on Engineering Management 42 (3), 215-222.

Nevins, J.L., Whitney, D., 1989. Concurrent Design of Products & Processes, A Strategy for the Next Generation in Manufacturing. McGraw-Hill, New York.

Nevins, T.M., Summe, G.L., Uttal, B., 1991. Commercializing technology: what the best companies do: managing product life cycles from start to finish. Harvard Business Review Paperback. Cambridge, MA.

O'Sullivan, A., 2003. Dispersed collaboration in a multi-firm, multi-team product-development project. Journal of Engineering and Technology Management 20 (1-2), 93-116.

Osborne, S.M., 1993. Product Development Cycle Time Characterization through Modeling of Process Change. Unpublished master's thesis, Sloan School of Management, MIT, Cambridge, MA.

Patrashkova-Volzdoska, R., McComb, S.A., Green, S.G., Compton, W.D., 2003. Examining a curvilinear relationship between communication frequency and team performance in cross-functional project teams. IEEE Transactions on Engineering Management 50 (3), 262-269.

Pisano, G.P., 1997. The Development Factory. Harvard Business School Press, Boston, MA.

Reichelt, Kimberley S., 1990. Halter Marine: A case study in the dangers of litigation. Master's thesis. Sloan School of Management. Massachusetts Institute on Technology. Cambridge, MA.

Repenning, N.P., 2001. Understanding fire fighting in new product development. The Journal of Product Innovation Management 18 (5), 285-300.

Richardson, G.P., Pugh III, A.L., 1981. Introduction to System Dynamics Modeling with Dynamo. MIT Press, Cambridge, MA.

Roberts, Edward B., 1974. A simple model of R&D project dynamics. R&D Management 5 (1), 1-15.

Rodrigues, A., Bowers., J., 1996. System dynamics in project management: a comparative analysis with traditional methods. System Dynamics Review 12 (2), 121-139.

Roemer, T.A., Ahmadi, R., Wang, R.H., 2000. Time-cost tradeoffs in overlapped product development. Operations Research 48 (6), 858-865.

Roemer, T.A., Ahmadi, R., 2004. Concurrent crashing and overlapping in product development. Operations Research 52 (4), 606-622.

Sargent, W., Westerberg, A., 1964. Speed-up in chemical engineering design. Transactions of the Institution of Chemical Engineers and the Chemical Engineer 42, 190-197.

Scott, S.G., 1997. Social identification effects in product and process development teams. Journal of Engineering and Technology Management 14 (2), 97-127.

Siemieniuch C.E., Sinclair M., 1999. Real-time collaboration in design engineering: an expensive fantasy or affordable reality? Behaviour & Information Technology 18 (5), 361-371.

Smith, P.G., Reinertsen, D.G., 1998. Developing Products in Half the Time, 2nd ed. Van Nostrand Reinhold, New York.

Smith, R.P., Morrow, J.A., 1999. Product development process modeling. Design Studies 20 (3), 237-261.

Smith, R.P., Eppinger, S.D., 1997a. Identifying controlling features of engineering design iteration. Management Science 43 (3), 276-293.

Smith, R.P., Eppinger, S.D., 1997b. A predictive model of sequential iteration in engineering design. Management Science 43 (8), 1104-1120.

Sobek, D.K., Ward, A.C., Liker, J.K., 1999. Toyota's principles of set-based concurrent engineering, Sloan Management Review 40 (2), 67-83.

Sterman, J.D., 1984. Appropriate summary statistics for evaluating the historical fit of system dynamics models. Dynamica 10 (2), 51-66.

Sterman, J.D., 2004. Business Dynamics: Systems Thinking and Modeling for a Complex World. Irwin/McGraw-Hill, Boston.

Steward, D.V., 1981. The design structure system: A method for managing the design of complex systems. IEEE Transactions on Engineering Management 49 (4), 428-442.

Swink, M.L., Sandvig, C., Mabert, V.A., 1996. Customizing concurrent engineering processes: five case studies. Journal of Production Innovation Management 13 (3), 229-244.

Takeuchi, H., Nonaka, I., 1991. The new product development game: managing product life cycles from start to finish. Harvard Business Review Paperback. Cambridge, MA.

Terwiesch, C., Loch, C.H., 1999. Measuring the effectiveness of overlapping development activities. Management Science 45 (4), 455-465.

Terwiesch, C., Loch, C.H., DeMeyer, A., 2002. Exchanging preliminary information in concurrent engineering: Alternative coordination strategies. Organization Science 13 (4), 402-419.

Theil, H., 1966. Applied Economic Forecasting. North Holland Publishing Company, Amsterdam.

Thomke, S., 1998. Simulation, learning, and R&D performance: Evidence from automotive development. Research Policy 27 (1), 55-74.

Thomke, S., Bell, D.E., 2001. Sequential testing in product development. Management Science 47 (2), 308-323.

Thompson, J.D., 1967. Organizations in action: Social science bases of administrative theory. McGraw-Hill, New York.

Von Corswant, F., Tunälv, C., 2002. Coordinating customers and proactive suppliers—a case study of supplier collaboration in product development. Journal of Engineering and Technology Management 19 (3–4), 249–261.

Wagner, S.M., Hoegl, M., 2006. Involving suppliers in product development: insights from R&D directors and project managers. Industrial Marketing Management 35 (8), 936-943.

Wang, Z., Yan, H.S., 2005. Optimizing the Concurrency for a group of design activities. IEEE Transactions on Engineering Management 52 (1), 102-118.

Ward, A., Liker, J.K., Cristiano, J.J., Sobek., D.K., 1995. The second Toyota paradox: how delaying decisions can make better cars faster. Sloan Management Review 36 (3), 43-61.

Wheelwright, S.C., Clark, K.B., 1992. Revolutionizing Product Development. The Free Press, New York.

Whitney, D.E., 1990. Designing the design process. Research in Engineering Design 2 (1), 3-13.

Williams, T., Eden, C., Ackerman, F., Tait, A., 1995. The effects of design changes and delays on project cost. The Journal of the Operational Research Society 46 (7), 809-818.

Williams, T., Ackermann, F., Eden, C., 2003. Structuring a delay and disruption claim: An application of cause-mapping and system dynamics. European Journal of Operational Research 148 (1), 192-204.

Williams, T., 2005. Assessing and moving on from the dominant project management discourse in the light of project overruns. IEEE Transactions on Engineering Management 52 (4), 497-508.

Williams, T.M. 1999. Seeking Optimum Project Duration Extension. The Journal of the Operational Research Society 50 (5), 460-467.

Womack, J.P., Jones, D.T., Roos, D., 1990. The Machine that Changed the World, The Story of Lean Production. Rawson Associates, New York.

Yassine, A.A., Chelst, K.R., Falkenburg, D.R., 1999. A decision analytic framework for evaluating concurrent engineering. IEEE Transactions on Engineering Management 46 (2), 144–157.

Yassine A.A., Falkenburg, D., Chelst, K., 1999. Engineering design management: an information structure approach. International Journal of Production Research 37 (13), 2957-2975.

Zhou, K.Z., Yim, C.K., Tse, D.K., 2005. The effects of strategic orientations on technology- and market-based breakthrough innovations. Journal of Marketing, 69 (2), 42-60.

APPENDIX A

PROOFS OF CHAPTER 3

Proof of Proposition 3.1

Assume Δ time units are spent on rework at time t_1 and Δ time units are spent on initial development at time t_2 ($t_2 > t_1$). Then $r\Delta$ work will be affected by $\int_{t_1}^{D_u} \mu_t \, dt$ modifications and Δ work will be affected by $\int_{t_2}^{D_u} \mu_t dt$ modifications. However, if initial development is done first, Δ work will be affected by $\int_{t_1}^{D_u} \mu_t$ modifications and $r\Delta$ work will be affected by $\int_{t_2}^{D_u} \mu_t$ modifications. It is clear that less work will be affected by upstream modifications in the later case. Therefore the priority of downstream development should be given to initial development. □

Proof of Proposition 3.2

(a) By proposition 3.1, only initial development will be done in the overlapped period if $\bar{t} \le t_s + D_d$. Therefore Equation (3.8) can be applied. It is a first order linear equation. The general solution of this equation is

$$\exp\{-k\int_{t_s}^{\bar{t}} \mu_\tau(t)dt\}[\int_{t_s}^{\bar{t}} \exp\{k\int_{t_s}^{t} \mu_\tau(x)dx\}dt + Con_1]$$

The downstream progress at t_s is zero. Therefore

$$\exp\{-k\int_{t_s}^{\bar{t}} \mu_\tau(t)dt\}[\int_{t_s}^{\bar{t}} \exp\{k\int_{t_s}^{t} \mu_\tau(x)dx\}dt + Con_1] = 0$$

$$Con_1 = 0$$

Therefore the downstream progress is

$$\exp\{-k\int_{t_s}^{\bar{t}}\mu_\tau(t)dt\}\int_{t_s}^{\bar{t}}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt.$$

If $\bar{t}\ge t_s+D_d$, the downstream progress at time t can be represented as equation (3.9). Similarly, the general solution of (3.9) is

$$\exp\{-k\int_{t_s}^{\bar{t}}\mu_\tau(t)dt\}[r\cdot\int_{D_d+t_s}^{\bar{t}}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt+Con_2]$$

The downstream progress at D_d+t_s is

$$\exp\{-k\int_{t_s}^{D_d+t_s}\mu_\tau(t)dt\}\int_{t_s}^{D_d+t_s}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt.$$

Therefore

$$\begin{aligned}&\exp\{-k\int_{t_s}^{D_d+t_s}\mu_\tau(t)dt\}[r\cdot\int_{D_d+t_s}^{D_d+t_s}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt+Con_2]\\&=\exp\{-k\int_{t_s}^{D_d+t_s}\mu_\tau(t)dt\}\int_{t_s}^{D_d+t_s}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt\end{aligned}$$

$$Con_2=\int_{t_s}^{D_d+t_s}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt$$

Consequently, if $\bar{t}\ge t_s+D_d$, the downstream progress at time $\bar{t}$ is

$$\begin{aligned}P(\bar{t})&=\exp\{-k\int_{t_s}^{\bar{t}}\mu_\tau(t)dt\}\int_{t_s}^{D_d+t_s}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt\\&\quad+r\cdot\exp\{-k\int_{t_s}^{\bar{t}}\mu_\tau(t)dt\}\int_{D_d+t_s}^{\bar{t}}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt\end{aligned}$$

(b) By proposition 3.2a, the downstream progress at time D_u is

$$\begin{cases}P(D_u)=\int_{t_s}^{D_u}\exp\{-k\int_{t}^{D_u}\mu_\tau(x)dx\}dt & \text{if } t_s\ge D_u-D_d\\[2ex] P(D_u)=\int_{t_s}^{D_d+t_s}\exp\{-k\int_{t}^{D_u}\mu_\tau(x)dx\}dt & \\ \quad+r\int_{D_d+t_s}^{D_u}\exp\{-k\int_{t}^{D_u}\mu_\tau(x)dx\}dt) & \text{if } t_s\le D_u-D_d\end{cases}$$

Substituting the above equations into (3.5), we derive the total amount of downstream rework.

(c) By proposition 3.2a and equation (3.8), if $\bar{t} \le t_s + D_d$ the downstream progress rate can be represented as

$$\frac{\partial P(\bar{t})}{\partial \bar{t}} = 1 - k\mu_\tau(\bar{t})\exp\{-k\int_{t_s}^{\bar{t}}\mu_\tau(t)dt\}\int_{t_s}^{\bar{t}}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt$$

$$= 1 - \mu_\tau(\bar{t})\exp\{-k\int_{t_s}^{\bar{t}}\mu_\tau(t)dt\}\int_{t_s}^{\bar{t}}\frac{1}{\mu_\tau(t)}d\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}$$

$$= 1 - \mu_\tau(\bar{t})\exp\{-k\int_{t_s}^{\bar{t}}\mu_\tau(t)dt\}[\left.\frac{\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx}{\mu_\tau(t)}\right|_{t_s}^{\bar{t}} + \int_{t_s}^{\bar{t}}\frac{\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}}{\mu_\tau^2(t)}\frac{\partial\mu_\tau(t)}{\partial t}dt]$$

$$= \mu_\tau(\bar{t})\exp\{-k\int_{t_s}^{\bar{t}}\mu_\tau(t)dt\}[\frac{1}{\mu_\tau(t_s)} - \int_{t_s}^{\bar{t}}\frac{\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}}{\mu_\tau^2(t)}\frac{\partial\mu_\tau(t)}{\partial t}dt] \qquad (3.22)$$

It is clear that $\frac{\partial P(\bar{t})}{\partial \bar{t}} > 0$ when $\mu_\tau(t)$ is nonincreasing with *t*. It is indefinite when $\mu_\tau(t)$ increases with t.

By proposition 3.2a and equation (3.9), if $\bar{t} \ge t_s + D_d$ the downstream progress rate can be represented as

$$\frac{\partial P(\bar{t})}{\partial \bar{t}} = 1 - k\mu_\tau(\bar{t})\exp\{-k\int_{t_s}^{\bar{t}}\mu_\tau(t)dt\}\int_{t_s}^{\bar{t}}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt$$

$$+ (r-1)[1 - k\mu_\tau(\bar{t})\exp\{-k\int_{t_s}^{\bar{t}}\mu_\tau(t)dt\}\int_{D_d+t_s}^{\bar{t}}\exp\{k\int_{t_s}^{t}\mu_\tau(x)dx\}dt]$$

Similarly, we can prove that $\frac{\partial P(\bar{t})}{\partial \bar{t}} > 0$ when $\mu_\tau(t)$ is nonincreasing with *t*. It is indefinite when $\mu_\tau(t)$ increases with t. □

Proof of Proposition 3.3

If $D_u \le D_d$, the solution is direct. When $D_u > D_d$, the earliest start time either equals zero or satisfies the following equation

$$D_d + D_r(t_e,\tau) = D_u - t_s \tag{3.23}$$

By proposition 3.2b, we have

$$D_r(t_s,\tau) = \frac{D_d - \int_{t_s}^{D_d+t_s} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt}{r}$$

$$+ D_u - t_s - D_d - \int_{D_d+t_s}^{D_u} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt\,.$$

The first derivative of the left-hand side of (3.23) with respect to t_s is

$$-1+\frac{1}{r}\exp\{-k\int_{t_s}^{D_u} \mu_\tau(x)dx\} + \frac{r-1}{r}\exp\{-k\int_{D_d+t_s}^{D_u} \mu_\tau(x)dx\}$$

The first derivative of the right-hand side of (3.23) is -1. Therefore equation (3.23) has either exactly one solution or $D_d + D_r(t_s,\tau) > D_u - t_s$ for all values of t_s, because the first derivative of the left-hand side is strictly larger than that of the right-hand side. Consequently, the earliest start time should be zero if

$$r\int_{D_d}^{D_u}\exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt + \int_0^{D_d} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt \le D_d$$

Otherwise t_e must satisfy the following equation:

$$r\int_{D_d+t_s}^{D_u} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt + \int_{t_s}^{D_d+t_s} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt = D_d \tag{3.24}$$

The first derivative of the left-hand side of (3.24) is

$$-(r-1)\exp\{-k\int_{D_d+t_s}^{D_u} \mu_\tau(x)dx\} - \exp\{-k\int_{t_s}^{D_u} \mu_\tau(x)dx\}$$

It is strictly negative. Therefore we can easily find t_e through a simple binary search.

□

Proof of Proposition 3.4

(a) Assume the functional interaction duration τ is exogenously determined. By proposition 3.2b and equation 3.10, if $t_s \geq D_u - D_d$, we have

$$G = c_t(D_u - \tau - t_s) - c_\tau \tau - (c_t + c_r)\left[\frac{D_u - t_s - \int_{t_s}^{D_u} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt}{r}\right] \tag{3.25}$$

The first and second derivatives of (3.25) with respect to t_s are

$$\frac{\partial G}{\partial t_s} = -c_t + \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{t_s}^{D_u} \mu_\tau(t)dt\}] \tag{3.26}$$

$$\frac{\partial^2 G}{\partial t_s^2} = -\frac{k \cdot \mu_\tau(t_s)}{r} \cdot (c_t + c_r) \cdot \exp\{-k\int_{t_s}^{D_u} \mu_\tau(t)dt\} \tag{3.27}$$

Inspection of (3.27) shows that $\partial^2 G / \partial t_s^2 < 0$ for all $t_s \in [t_e, D_u]$, i.e., the objective function is concave with respect to t_s.

If $t_s < D_u - D_d$, we have

$$G = -\tau(c_t + c_\tau) + c_t D_d - c_r(D_u - t_s - D_d)$$

$$-(c_t + c_r)\left[\frac{D_d - \int_{t_s}^{D_d + t_s} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt}{r} - \int_{D_d + t_s}^{D_u} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt\right] \tag{3.28}$$

The first and second derivatives of (3.28) with respect to t_s are

$$\frac{\partial G}{\partial t_s} = c_r - \frac{(r-1)(c_t + c_r)}{r}\exp\{-k\int_{D_d + t_s}^{D_u} \mu_\tau(t)dt\} - \frac{c_t + c_r}{r}\exp\{-k\int_{t_s}^{D_u} \mu_\tau(t)dt \tag{3.29}$$

$$\frac{\partial^2 G}{\partial t_s^2} = -\frac{k \cdot \mu_\tau(D_d + t_s)}{r} \cdot (r-1) \cdot (c_t + c_r) \cdot \exp\{-k\int_{D_d + t_s}^{D_u} \mu_\tau(t)dt\}$$

$$-\frac{\mu_\tau(t_s)}{r} \cdot (c_t + c_r) \cdot \exp\{-k\int_{t_s}^{D_u} \mu_\tau(t)dt \tag{3.30}$$

Inspection of equation (3.30) shows that $\partial^2 G / \partial t_s^2 < 0$ for $t_s \in [t_e, D_u]$, i.e., the objective function is concave with respect to t_s.

(b) Since the objective function is concave with respect to t_s, complete overlapping must be optimal if

$$\begin{cases} -c_t + \dfrac{c_t + c_r}{r}[1 - \exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\}] \le 0 \quad \mathit{if}\ t_e \ge D_u - D_d \\ \\ c_r - \dfrac{(r-1)(c_t + c_r)}{r}\exp\{-k\int_{D_d+t_e}^{D_u} \mu_\tau(t)dt\} \\ -\dfrac{c_t + c_r}{r}\exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\} \le 0 \qquad\qquad \mathit{if}\ t_e \le D_u - D_d \end{cases}$$

(c) Rewriting (3.13) yields

$$r \ge (1 + \frac{c_r}{c_t}) - (1 + \frac{c_r}{c_t})\exp\{-k\int_{t_s}^{D_u} \mu_\tau(t)dt\} \tag{3.31}$$

In (3.31), the first term is not relevant to the properties of uncertainty, dependency, and upstream evolution. The second term is always negative. Therefore we can conclude that the optimal start time of the downstream stage must be t_e, when $r \ge 1 + c_r / c_t$. By comparing equations (3.13) and (3.14), we have

$$c_r - \frac{(r-1)(c_t + c_r)}{r}\exp\{-k\int_{D_d+t_e}^{D_u} \mu_\tau(t)dt\}$$

$$-\frac{c_t + c_r}{r}\exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\} < -c_t + \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\}]$$

Therefore, the right-hand side must be negative when $r \ge 1 + c_r / c_t$.

(d) *If* $t_e \ge D_u - D_d$ and $-c_t + \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\}] > 0$, there must exist a unique t_s^* where $\partial G / \partial t = 0$. Therefore

$$-c_t + \frac{c_t + c_r}{r}(1 - \exp\{-k\int_{t_s^*}^{D_u} \mu_\tau(t)dt\}) = 0 \tag{3.32}$$

(3.32) can be simplified as

$$\int_{t_s^*}^{D_u} \mu_\tau(t)dt = -\frac{1}{k}\cdot \ln(1-\frac{r\cdot c_t}{c_t+c_r})$$

Similarly, if $t_e \le D_u - D_d$ and

$$c_r - \frac{(r-1)(c_t+c_r)}{r}\exp\{-k\int_{D_d+t_e}^{D_u} \mu_\tau(t)dt\} - \frac{c_t+c_r}{r}\exp\{-k\int_{t_e}^{D_u} \mu_\tau(t)dt\} > 0$$

there must exist a unique t_s^* where $t_s^* \le D_u - D_d$ and

$$c_r - \frac{(r-1)(c_t+c_r)}{r}\exp\{-k\int_{D_d+t_s}^{D_u} \mu_\tau(t)dt\} - \frac{c_t+c_r}{r}\exp\{-k\int_{t_s}^{D_u} \mu_\tau(t)dt\} = 0$$

or $t_s^* \ge D_u - D_d$ and $\int_{t_s^*}^{D_u} \mu_\tau(t)dt = -\frac{1}{k}\cdot \ln(1-\frac{r\cdot c_t}{c_t+c_r})$

The left-hand side of (3.15) and (3.16) strictly decreases when t_s increases. Therefore, through a simple binary search, the optimal start time can be derived. □

Proof of Proposition 3.5

(a) $\left.\frac{\partial^2 G}{\partial t_s \partial k}\right|_{t_s=t_s^*} = \frac{1}{r}\cdot(c_t+c_r)\cdot\int_{t_s^*}^{D_u} \mu_\tau(t)dt\cdot\exp\{-k\int_{t_s^*}^{D_u} \mu_\tau(t)dt\} > 0$ if $t_s^* \ge D_u - D_d$

$$\left.\frac{\partial^2 G}{\partial t_s \partial k}\right|_{t_s=t_s^*} = \frac{1}{r}\cdot(r-1)\cdot(c_t+c_r)\cdot\int_{D_d+t_s^*}^{D_u} \mu_\tau(t)dt\cdot\exp\{-k\int_{D_d+t_s^*}^{D_u} \mu_\tau(t)dt\}$$

$$+\frac{1}{r}\cdot(c_t+c_r)\cdot\int_{t_s^*}^{D_u} \mu_\tau(t)dt\cdot\exp\{-k\int_{t_s^*}^{D_u} \mu_\tau(t)dt\} > 0 \text{ if } t_s^* \le D_u - D_d$$

By the implicit function theorem

$$\frac{\partial t_s^*}{\partial k} = -\frac{\left.\frac{\partial^2 G}{\partial t_s \partial e}\right|_{t_s=t_s^*}}{\left.\frac{\partial^2 G}{\partial t_s^2}\right|_{t_s=t_s^*}} > 0$$

It implies that higher dependency parameter k increases the optimal downstream start time t_s^*.

$$\left.\frac{\partial^2 G}{\partial t_s \partial \mu_\tau(t)}\right|_{t_s=t_s^*} = \frac{k}{r}\cdot(c_t+c_r)\cdot\frac{\partial\int_{t_s^*}^{D_u}\mu_\tau(t)dt}{\partial\mu_\tau(t)}\cdot\exp\{-k\int_{t_s^*}^{D_u}\mu_\tau(t)dt\}>0 \text{ if } t_s^*\ge D_u-D_d$$

$$\left.\frac{\partial^2 G}{\partial t_s \partial \mu_\tau(t)}\right|_{t_s=t_s^*} = \frac{k}{r}\cdot(r-1)\cdot(c_t+c_r)\cdot\frac{\partial\int_{D_d+t_s^*}^{D_u}\mu_\tau(t)dt}{\partial\mu_\tau(t)}\cdot\exp\{-k\int_{D_d+t_s^*}^{D_u}\mu_\tau(t)dt\}$$

$$+\frac{k}{r}\cdot(c_t+c_r)\cdot\frac{\partial\int_{t_s^*}^{D_u}\mu_\tau(t)dt}{\partial\mu_\tau(t)}\cdot\exp\{-k\int_{t_s^*}^{D_u}\mu_\tau(t)dt\}>0 \text{ if } t_s^*\le D_u-D_d$$

By the implicit function theorem

$$\frac{\partial t_s^*}{\partial(a+b\exp\{-\lambda\tau\})} = -\frac{\left.\frac{\partial^2 G}{\partial t_s\partial(a+b\exp\{-\lambda\tau\})}\right|_{t_s=t_s^*}}{\left.\frac{\partial^2 G}{\partial t_s^2}\right|_{t_s=t_s^*}}>0$$

It implies that higher uncertainty level, $a+b\exp\{-\lambda\tau\}$ increases the optimal downstream start time t_s^*.

By the definition of the model, faster evolution decreases $\int_t^{D_u}\mu_\tau(t)dt$, assuming the total amount of upstream modifications is constant. Let e denote the evolution speed. Mathematically

$$\frac{\partial\int_t^{D_u}\mu_\tau(t)dt}{\partial e}<0$$

$$\left.\frac{\partial^2 G}{\partial t_s\partial e}\right|_{t_s=t_s^*} = \frac{k}{r}\cdot(c_t+c_r)\cdot\exp\{-k\int_{t_s^*}^{D_u}\mu_\tau(t)dt\}\cdot\frac{\partial\int_{t_s^*}^{D_u}\mu_\tau(t)dt}{\partial e}<0 \text{ if } t_s^*\ge D_u-D_d$$

$$\left.\frac{\partial^2 G}{\partial t_s \partial e}\right|_{t_s=t_s^*} = \frac{k}{r}\cdot(r-1)(c_t+c_r)\cdot\exp\{-k\int_{D_d+t_s^*}^{D_u}\mu_\tau(t)dt\}\cdot\frac{\partial\int_{D_d+t_s^*}^{D_u}\mu_\tau(t)dt}{\partial e}$$

$$+\frac{k}{r}\cdot(c_t+c_r)\cdot\exp\{-k\int_{t_s^*}^{D_u}\mu_\tau(t)dt\}\cdot\frac{\partial\int_{t_s^*}^{D_u}\mu_\tau(t)dt}{\partial e}<0 \text{ if } t_s^*\le D_u-D_d$$

By the implicit function theorem

$$\frac{\partial t_s^*}{\partial e} = -\frac{\left.\frac{\partial^2 G}{\partial t_s \partial e}\right|_{t_s=t_s^*}}{\left.\frac{\partial^2 G}{\partial t_s^2}\right|_{t_s=t_s^*}}<0$$

It implies that t_s^* decrease when evolution becomes faster.

(b) $$\left.\frac{\partial^2 G}{\partial t_s \partial r}\right|_{t_s=t_s^*} = -\frac{1}{r^2}\cdot(c_t+c_r)\cdot[1-\exp\{-k\int_{t_s^*}^{D_u}\mu_\tau(t)dt\}]<0 \text{ if } t_s^*\ge D_u-D_d$$

$$\left.\frac{\partial^2 G}{\partial t_s \partial r}\right|_{t_s=t_s^*} = -\frac{1}{r^2}\cdot(c_t+c_r)\cdot[\exp\{-k\int_{D_d+t_s^*}^{D_u}\mu_\tau(t)dt\}$$

$$-\exp\{-k\int_{t_s^*}^{D_u}\mu_\tau(t)dt\}]<0 \text{ if } t_s^*\le D_u-D_d$$

By the implicit function theorem

$$\frac{\partial t_s^*}{\partial r} = -\frac{\left.\frac{\partial^2 G}{\partial t_s \partial r}\right|_{t_s=t_s^*}}{\left.\frac{\partial^2 G}{\partial t_s^2}\right|_{t_s=t_s^*}}<0$$

It implies that the overlapping degree increases with r (i.e. t_s^* decreases with r).

(c) $$\left.\frac{\partial^2 G}{\partial t_s \partial c_t}\right|_{t_s=t_s^*} = \frac{1}{r}\cdot[1-\exp\{-k\int_{t_s^*}^{D_u}\mu_\tau(t)dt\}]-1<0 \text{ if } t_s^*\ge D_u-D_d$$

$$\left.\frac{\partial^2 G}{\partial t_s \partial c_t}\right|_{t_s=t_s^*} = -(1-\frac{1}{r})\cdot \exp\{-k\int_{D_d+t_s^*}^{D_u} \mu_\tau(t)dt\}$$

$$-\frac{1}{r}\exp\{-k\int_{t_s^*}^{D_u} \mu_\tau(t)dt\} < 0 \text{ if } t_s^* \le D_u - D_d$$

By the implicit function theorem

$$\frac{\partial t_s^*}{\partial c_t} = -\frac{\left.\frac{\partial^2 G}{\partial t_s \partial c_t}\right|_{t_s=t_s^*}}{\left.\frac{\partial^2 G}{\partial t_s^2}\right|_{t_s=t_s^*}} < 0$$

It implies that the downstream start time t_s^* decreases when the opportunity cost of time c_t increases.

$$\left.\frac{\partial^2 G}{\partial t_s \partial c_r}\right|_{t_s=t_s^*} = \frac{1}{r}\cdot[1-\exp\{-k\int_{t_s^*}^{D_u} \mu_\tau(t)dt\}] > 0 \text{ if } t_s^* \ge D_u - D_d$$

$$\left.\frac{\partial^2 G}{\partial t_s \partial c_r}\right|_{t_s=t_s^*} = \frac{1}{r}\cdot[\exp\{-k\int_{D_d+t_s^*}^{D_u} \mu_\tau(t)dt\} - \exp\{-k\int_{t_s^*}^{D_u} \mu_\tau(t)dt\}]$$

$$+1-\exp\{-k\int_{D_d+t_s^*}^{D_u} \mu_\tau(t)dt\} > 0 \text{ if } t_s^* \le D_u - D_d$$

By the implicit function theorem

$$\frac{\partial t_s^*}{\partial c_r} = -\frac{\left.\frac{\partial^2 G}{\partial t_s \partial c_r}\right|_{t_s=t_s^*}}{\left.\frac{\partial^2 G}{\partial t_s^2}\right|_{t_s=t_s^*}} > 0$$

It implies that the downstream start time t_s^* increases with c_r.

(d) If $D_d \in [D_u, +\infty)$, the optimal overlapping degree can be derived from equation (3.15) which is independent of D_d. □

Proof of Corollary 3.1

(a) This result follows from Proposition 2(c) which shows that the optimal start time of downstream stage must be t_e when $r \geq 1 + c_r / c_t$. For time-driven projects, $c_r / c_t \to 0$. Therefore complete overlapping is optimal for time-to-market problem when $r \geq 1$.

(b) For cost-driven projects, $c_r >> c_t$, the opportunity cost of time can be ignored and thus the right-hand side of (3.26) and (3.29) becomes positive $\forall t_s \in [t_e, D_u]$. That is $\frac{\partial G}{\partial t_s} > 0 \ \forall t_s \in [t_e, D_u]$. As G is concave with respect to t_s, we have $t_s^* \approx D_u$. □

Proof of Proposition 3.6

(a) Assume that optimal overlapping discussed above is followed. If $t_s^* \geq D_u - D_d$, The first derivative of (3.9) with respect to τ is

$$\frac{\partial G}{\partial \tau} = \frac{kb\lambda(c_t + c_r)\exp\{-\lambda\tau\}}{r(a+b)} \int_{t_s^*}^{D_u} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\} \int_t^{D_u} \mu_0(x)dxdt - (c_t + c_\tau) \tag{3.33}$$

If $t_s^* \leq D_u - D_d$, then

$$\frac{\partial G}{\partial \tau} = \frac{kb\lambda(c_t + c_r)\exp\{-\lambda\tau\}}{r(a+b)} [\int_{t_s^*}^{D_d + t_s^*} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\} \int_t^{D_u} \mu_0(x)dxdt$$

$$+ r\int_{D_d + t_s^*}^{D_u} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\} \int_t^{D_u} \mu_0(x)dxdt] - (c_t + c_\tau) \tag{3.34}$$

It is clear that $\int_t^{D_u} \mu_0(x)dx \leq a + b$. Applying it in Equations (3.33) and (3.34), we have

$$\frac{\partial G}{\partial \tau} < \frac{kb\lambda(c_t + c_r)\exp\{-\lambda\tau\}}{r}\int_{t_e}^{D_u} \exp\{-k\int_t^{D_u} \mu_\tau(x)dx\}dt - (c_t + c_\tau)$$

$$< \frac{kb\lambda(c_t + c_r)(D_u - t_e)\exp\{-\lambda\tau\}}{r} - (c_t + c_\tau) \tag{3.35}$$

$$< \frac{kb\lambda(c_t + c_r)(D_u - t_e)}{r} - (c_t + c_\tau)$$

If (3.18) holds, $\partial G/\partial\tau \le 0 \,\forall\, \tau \ge 0$. Thus $\tau^* = 0$.

Equation (3.35) decreases with functional interaction time τ and thus $\partial G/\partial\tau$ is negative when

$$\tau \ge \frac{1}{\lambda}\ln[\frac{kb\lambda(c_t + c_r)(D_u - t_e)}{r(c_t + c_\tau)}]$$

Therefore $0 \le \tau^* \le \max\{0, \frac{1}{\lambda}\ln[\frac{kb\lambda(c_t + c_r)(D_u - t_e)}{c_t + c_\tau}]\}$.

(b) If $t_s^* \ge D_u - D_d$, the Second derivative of (3.9) with respect to τ is

$$\frac{\partial^2 G}{\partial \tau^2} = -\frac{kb\lambda^2(c_t + c_r)\exp\{-\lambda\tau\}}{r(a+b)}$$

$$*\int_{t_s^*}^{D_u}[1 - \frac{kb\exp\{-\lambda\tau\}}{a+b}\int_t^{D_u}\mu_0(x)dx]\exp\{-k\int_t^{D_u}\mu_\tau(x)dx\}\int_t^{D_u}\mu_0(x)dxdt$$

If $t_s^* \le D_u - D_d$, then

$$\frac{\partial^2 G}{\partial \tau^2} = -\frac{kb\lambda^2(c_t + c_r)\exp\{-\lambda\tau\}}{r(a+b)}$$

$$*[\int_{t_s^*}^{D_u + t_s^*}(1 - \frac{kb\exp\{-\lambda\tau\}}{a+b}\int_t^{D_u}\mu_0(x)dx)\exp\{-k\int_t^{D_u}\mu_\tau(x)dx\}\int_t^{D_u}\mu_0(x)dxdt$$

$$+ r\int_{D_d + t_s^*}^{D_u}(1 - \frac{kb\exp\{-\lambda\tau\}}{a+b}\int_t^{D_u}\mu_0(x)dx)\exp\{-k\int_t^{D_u}\mu_\tau(x)dx\}\int_t^{D_u}\mu_0(x)dxdt]$$

It is direct that $1-\frac{kb\exp\{-\lambda\tau\}}{a+b}\int_t^{D_u}\mu_0(x)dx > 1-\frac{kb}{a+b}\int_{t_e}^{D_u}\mu_0(t)dt$. If

$1-\frac{kb}{a+b}\int_{t_e}^{D_u}\mu_0(t)dt \ge 0$, G is concave. Thus, $\tau^*=0$ when $\left.\frac{\partial G}{\partial \tau}\right|_{\tau=0} \le 0$.

(c) If $\lambda\tau_{\max}$ is small, a first order approximation of the functional interaction function is $a+b-b\lambda\tau$.

If $t_s^* \ge D_u - D_d$, then

$$\frac{\partial^2 G}{\partial \tau^2} = \frac{(kb\lambda)^2(c_t+c_r)}{r(a+b)^2}\int_{t_s^*}^{D_u}\exp\{-k\int_t^{D_u}\mu_\tau(x)dx\}[\int_t^{D_u}\mu_0(x)dx]^2 dt > 0$$

If $t_s^* \le D_u - D_d$, then

$$\frac{\partial^2 G}{\partial \tau^2} = \frac{(kb\lambda)^2(c_t+c_r)}{r(a+b)^2}$$

$$*[\int_{t_s^*}^{D_u+t_s^*}\exp\{-k\int_t^{D_u}\mu_\tau(x)dx\}(\int_t^{D_u}\mu_0(x)dx)^2 dt$$

$$+r\int_{D_d+t_s^*}^{D_u}\exp\{-k\int_t^{D_u}\mu_\tau(x)dx\}(\int_t^{D_u}\mu_0(x)dx)^2 dt] > 0$$

G is convex with respect to τ, therefore

$G^* = \max(G|_{\tau=0,t_s^*}, G|_{\tau_{\max},t_s^*})$ □

APPENDIX B

PROOFS OF CHAPTER 4

Proof of Proposition 4.1

(a) It is clear that our model is a constrained problem with an equality constraint. Using the method of Lagrange multipliers, it is converted into an unconstrained function

$$h(Q_{1,n},\lambda)=c_t(D_u-t_0-n\beta)-nc_\beta-\frac{c_t+c_r}{r}\sum_{i=1}^{n}Q_i[1-\exp\{-k\int_{t_{i-1}}^{D_u}\mu(t)dt\}]+\lambda(D_u-t_0-\sum_{i=1}^{n}Q_i)$$

The first derivative of the Lagrangian function with respect to Q_i ($1\le i\le n$) is

$$\frac{\partial G}{\partial Q_i}=-\frac{c_t+c_r}{r}[1-\exp\{-k\int_{t_{i-1}}^{D_u}\mu(t)dt\}-\sum_{j=i+1}^{n}kQ_j\mu(t_{j-1})\exp\{-k\int_{t_{j-1}}^{D_u}\mu(t)dt\}]-\lambda=0 \quad 1\le i\le n\ ,$$

which can be simplified to

$$Q_i=\frac{1-\exp\{-k\int_{t_{i-2}}^{t_{i-1}}\mu(t)dt\}}{k\mu(t_{i-1})} \qquad 2\le i\le n,$$

The first derivative with respect to λ is

$$\sum_{i=1}^{n}Q_i=D_u-t_0.$$

(b) In overlapped process, at least an information exchange should be arranged at time D_u. The potential benefit of more frequent information exchange is no more than $c_t(D_u-t_s)$ and the cost for them is $(n-1)(c_t\beta+c_\beta)$. To optimize project performance, the communication cost should be less than the potential benefit. Therefore,

$n^* < 1 + c_t(D_u - t_0)/(c_t\beta + c_\beta)$. □

Proof of Proposition 4.2

If the downstream starts at t_0 and t^*_{1,n^*} is the optimal communication policy, the project performance can be written as

$$G^*(t_0) = c_t(t_i^* - t_0 - i\beta) - ic_\beta - \frac{c_t + c_r}{r}\sum_{i=1}^{i} Q_j^*[1 - \exp\{-k\int_{t_{j-1}^*}^{D_u} \mu(t)dt\}] + G(t_i^*, t_{i+1,n}^*) . \quad (4.18)$$

If $G^*(t_i^*) > G(t_i^*, t^*_{i+1,n^*})$, then $G^*(t_0)$ can be improved by replacing $G(t_i^*, t^*_{i+1,n^*})$ with $G^*(t_i^*)$. A conflict arises. Therefore, $G(t_i^*, t^*_{i+1,n^*})$ is the optimal solution when the downstream starts at t_i^* . □

Preparation for Proof of Proposition 4.3

Let $\hat{t}_{1,\hat{n}}$ be the optimal communication policy for complete overlapping and $\hat{Q}_i$ be the optimal interval between the $(i-1)^{th}$ and i^{th} information exchange.

LEMMA 4.1.

(a) If $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_i}^{D_u} \mu(x)dx\}] \geq 0$, $G(\hat{t}_{i+1}, \hat{t}_{i+2,\hat{n}})$ *is the optimal solution when* $t_0 \in [\hat{t}_{i+1}, D_u]$.

(b) If $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_i}^{D_u} \mu(x)dx\}] \leq 0$, $G(\hat{t}_{i+1}, \hat{t}_{i+2,\hat{n}})$ *is the optimal solution when* $t_0 \in [t_e, \hat{t}_{i+1}]$.

Proof. (a) If the project starts at $\hat{t}_i$, the project performance with the optimal communication policy can be represented as

$$G(\hat{t}_i,\hat{t}_{i+1,\hat{n}})=G(\hat{t}_{i+1},\hat{t}_{i+2,\hat{n}})+\hat{Q}_{i+1}[c_t-\frac{c_t+c_r}{r}(1-\exp\{-k\int_{\hat{t}_i}^{D_u}\mu(x)dx\})]-c_t\beta-c_r. \qquad (4.19)$$

Assume $t_0^*\in(\hat{t}_{i+1},D_u]$ and $G(t_0^*,t_{1,n}^*)>G(\hat{t}_{i+1},\hat{t}_{i+2,\hat{n}})$. Then $t_0^*-\hat{t}_i>\hat{Q}_{i+1}$. Let the communication policy be t^*_{0,n^*} when the project starts at $\hat{t}_i$. Then, $G(\hat{t}_i,t^*_{0,n^*})$ can be represented as

$$G(\hat{t}_i,t^*_{0,n^*})=G(t_0^*,t^*_{1,n^*})+(t_0^*-\hat{t}_i)[c_t-\frac{c_t+c_r}{r}(1-\exp\{-k\int_{\hat{t}_i}^{D_u}\mu(x)dx\})]-c_t\beta-c_r. \qquad (4.20)$$

Comparing (4.19) and (4.20), we get $G(\hat{t}_i,t^*_{0,n^*})>G(\hat{t}_i,\hat{t}_{i+1,\hat{n}})$. The conflict arises since $G(\hat{t}_i,\hat{t}_{i+1,\hat{n}})$ is the optimal solution when downstream starts at $\hat{t}_i$. Therefore, $G(\hat{t}_{i+1},\hat{t}_{i+2,\hat{n}})\geq G(t_0^*,t^*_{1,n^*})$ and $G(\hat{t}_{i+1},\hat{t}_{i+2,\hat{n}})$ is the optimal profit when $t_0\in[\hat{t}_{i+1},D_u]$.

(b) Assume $t_0^*\in[\hat{t}_i,\hat{t}_{i+1})$ and $G(t_0^*,t^*_{1,n^*})>G(\hat{t}_{i+1},\hat{t}_{i+2,\hat{n}})$. Then $t_0^*-\hat{t}_i<\hat{Q}_{i+1}$. Let the communication policy be t^*_{0,n^*} when the project starts at $\hat{t}_i$. Then, $G(\hat{t}_i,t^*_{0,n^*})$ can be represented as

$$G(\hat{t}_i,t^*_{0,n^*})=G(t_0^*,t^*_{1,n^*})+(t_0^*-\hat{t}_i)[c_t-\frac{c_t+c_r}{r}(1-\exp\{-k\int_{\hat{t}_i}^{D_u}\mu(x)dx\})]-c_t\beta-c_r. \qquad (4.21)$$

Comparing (4.19) and (4.21), we get $G(\hat{t}_i,t^*_{0,n^*})>G(\hat{t}_i,\hat{t}_{i+1,\hat{n}})$. The conflict arises since $G(\hat{t}_i,\hat{t}_{i+1,\hat{n}})$ is the optimal solution. Therefore, $G(\hat{t}_{i+1},\hat{t}_{i+2,\hat{n}})\geq G(t_0^*,t^*_{1,n^*})$ and $G(\hat{t}_{i+1},\hat{t}_{i+2,\hat{n}})$ is the optimal profit when $t_0\in[\hat{t}_i,\hat{t}_{i+1}]$.

It is clear that $c_t-\frac{c_t+c_r}{r}[1-\exp\{-k\int_{\hat{t}_{i-1}}^{D_u}\mu(x)dx\}]\leq 0$. By the same logic, we derive that: $G(\hat{t}_i, \hat{t}_{i+1,\hat{n}})$ is the optimal performance when the downstream starts in $[\hat{t}_{i-1}, \hat{t}_i]$; $G(\hat{t}_{i-1}, \hat{t}_{i,\hat{n}})$ is the optimal performance when the downstream starts in $[\hat{t}_{i-2}, \hat{t}_{i-1}]$, and so on. Combining the above results, it is evident that $G(\hat{t}_{i+1}, \hat{t}_{i+2,\hat{n}})$ is the optimal performance when $t_0 \in [t_e, \hat{t}_{i+1}]$. □

Proof of Proposition 4.3

(a) By Lemma 1(a) and $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_i}^{D_u} \mu(x)dx\}] > 0$, $G(\hat{t}_{i+1}, \hat{t}_{i+2,\hat{n}})$ is the optimal solution when $t_0 \in [\hat{t}_{i+1}, D_u]$.

$c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_{i-1}}^{D_u} \mu(x)dx\}] < 0$ since $\hat{t}_i$ is the smallest one in $t_e, \hat{t}_1, \hat{t}_2, \cdots, \hat{t}_{\hat{n}}$, which satisfies $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_i}^{D_u} \mu(x)dx\}] > 0$. By Lemma 4.1(b), $G(\hat{t}_i, \hat{t}_{i+1,\hat{n}})$ is the optimal solution when $t_0 \in [t_e, \hat{t}_i]$. Combing the results, it is evident that the optimal downstream start time locates in $[\hat{t}_i, \hat{t}_{i+1}]$.

(b) By Lemma 4.1(b), it is clear that $t_0^* = D_u$ when $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_{\hat{n}-1}}^{D_u} \mu(x)dx\}] < 0$.

(c) By Lemma 4.1, it is evident that $G(\hat{t}_{i+1}, \hat{t}_{i+2,\hat{n}})$ is the optimal performance when $c_t - \frac{c_t + c_r}{r}[1 - \exp\{-k\int_{\hat{t}_i}^{D_u} \mu(x)dx\}] = 0$. Consequently, the downstream should start at time $\hat{t}_{i+1}$. □

Proof of Proposition 4.4

(a) Let $T(t_0^*, t_{1,n^*}^*)$ be the optimal solution for the time-to-market problem (i.e. $T^* = T(t_0^*, t_{1,n^*}^*)$). If the downstream stage starts at t_e and the communication policy is t_{1,n^*}^*, then the development cycle time is

$$T(t_e, t_{0,n^*}^*) = T(t_0^*, t_{1,n^*}^*) + (t_0^* - t_e)[\frac{1}{r}(1-\exp\{-k\int_{t_e}^{D_u}\mu(x)dx\}) - 1] + \beta . \tag{4.22}$$

$T(t_e, \hat{t}_{1,\hat{n}})$ is the optimal cycle time for complete overlapping and $T(t_e, \hat{t}_{1,\hat{n}}) \le T(t_e, t_{0,n^*}^*)$, Therefore

$$T(t_e, \hat{t}_{1,\hat{n}}) \le T(t_0^*, t_{1,n^*}^*) + (t_0^* - t_e)[\frac{1}{r}(1-\exp\{-k\int_{t_e}^{D_u}\mu(x)dx\}) - 1] + \beta . \tag{4.23}$$

When $r \ge 1$, (4.23) can be simplified to

$$T(t_0^*, t_{1,n^*}^*) \ge T(t_e, \hat{t}_{1,\hat{n}}) - \beta . \tag{4.24}$$

Together with $T(t_0^*, t_{1,n^*}^*) \le T(t_e, \hat{t}_{1,\hat{n}})$, we derive Equation (4.15).

(b) By Lemma 1(a), it is clear that $t_e \le t_0^* \le \hat{t}_1$. Therefore, we only need to prove that $t_0^* \le t_e + \varphi$, where $\varphi = \beta / \exp\{-k\int_{t_e}^{D_u}\mu(x)dx\}$. Suppose $t_0^* > t_e + \varphi$, then $(t_0^* - t_e)\exp\{-k\int_{t_e}^{D_u}\mu(x)dx\} > \beta$. If the communication policy for complete overlapping is t_{0,n^*}^*, the development cycle time should be

$$T(t_e, t_{0,n^*}^*) = T(t_0^*, t_{1,n^*}^*) - (t_0^* - t_e)\exp\{-k\int_{t_e}^{D_u}\mu(x)dx\} + \beta < T(t_0^*, t_{1,n^*}^*) \tag{4.25}$$

The conflict arises. Therefore $t_e \le t_0^* \le \min(\hat{t}_1, t_e + \varphi)$. □

Proof of Proposition 4.5

(a) For the linear evolution case, the objective function can be simplified to

$$\max : G = c_t(D_u - t_0 - n\beta) - nc_\beta - \frac{c_t + c_r}{r}\sum_{i=1}^{n} Q_i(1 - \exp\{-k\mu\sum_{j=i}^{n} Q_j\}).$$

Using the method of Lagrange multipliers (similar to Proposition 1), we derive

$$\begin{cases} Q_i = \dfrac{1-\exp\{-k\mu Q_{i-1}\}}{k\mu} & 2 \le i \le n \\ \sum_{i=1}^{n} Q_i = D_u - t_0 \end{cases}$$

The first derivative of Q_i with respect to Q_1 is

$$\frac{\partial Q_i}{\partial Q_1} = \exp\{-k\mu(t_{i-1} - t_0)\} \quad 2 \le i \le n \tag{4.26}$$

Inspection of (4.26) shows that $\partial Q_i / \partial Q_1 > 0$. Therefore, $\sum_{i=1}^{n} Q_i$ strictly increases with Q_1 and there must be a unique Q_1 satisfying $\sum_{i=1}^{n} Q_i = D_u - t_0$.

(b) Inspection of (4.26) shows that $\partial Q_i / \partial Q_1 > \partial Q_{i+1} / \partial Q_1$, i.e. Q_i increases faster than Q_{i+1}. Therefore $Q_i > Q_{i+1}$.

(c) Let $G^*(t_0) = G(t_0, t^*_{1,n^*})$. Then the first derivative of $G(t_0, t^*_{1,n^*})$ with respect to t_0 is

$$\frac{\partial G(t_0, t^*_{1,n^*})}{\partial t_0} = -c_t - \frac{c_t + c_r}{r}\sum_{i=1}^{n^*}\frac{\partial Q_i^*}{\partial t_0} - \frac{c_t + c_r}{r}\sum_{i=1}^{n^*}[\exp\{-k\mu\sum_{j=i}^{n^*} Q_j^*\}(-\frac{\partial Q_i^*}{\partial t_0} + k\mu Q_i^*\sum_{j=i}^{n^*}\frac{\partial Q_j^*}{\partial t_0})]$$

By equations (4.8) and (4.17)

$$\frac{\partial G(t_0, t^*_{1,n^*})}{\partial t_0} = -c_t + \frac{c_t + c_r}{r} - \frac{c_t + c_r}{r}\exp\{-k\mu(D_u - t_0)\}(-\frac{\partial Q_1^*}{\partial t_0} - k\mu Q_1^*)$$

$$-\frac{c_t+c_r}{r}\sum_{i=2}^{n^*}[\exp\{-k\mu\sum_{j=i}^{n^*}Q_j^*\}\sum_{j=i+1}^{n^*}\frac{\partial Q_j^*}{\partial t_0}-\exp\{-k\mu\sum_{j=i-1}^{n^*}Q_j^*\}\sum_{j=i}^{n^*}\frac{\partial Q_j^*}{\partial t_0}]$$

$$=-c_t+\frac{c_t+c_r}{r}-\frac{c_t+c_r}{r}[-k\mu Q_1^*\exp\{-k\mu(D_u-t_0)\}-\exp\{-k\mu(D_u-t_0)\}\sum_{i=1}^{n^*}\frac{\partial Q_i^*}{\partial t_0}]$$

$$=-c_t+\frac{c_t+c_r}{r}-\frac{c_t+c_r}{r}\exp\{-k\mu(D_u-t_0)\}(1-k\mu Q_1^*) \tag{4.27}$$

By Proposition 4.5(a), it is evident that Q_1 decreases with n. Consequently, (4.27) decreases with n. Assume $\tilde{t}_{1,\tilde{n}}$ is the optimal communication policy when t_0 and $\tilde{n}$ ($\tilde{n}>n^*$) are given. Then

$$G(t_0,t^*_{1,n^*})>G(t_0,\tilde{t}_{1,\tilde{n}}), \tag{4.28}$$

$$\partial G(t_0,t^*_{1,n^*})/\partial t_0>\partial G(t_0,\tilde{t}_{1,\tilde{n}})/\partial t_0. \tag{4.29}$$

Consequently, n^* is non-increasing with t_0.

(d) To prove that the profit function $G^*(t_0,n)$ is concave with respect to n, it is necessary and sufficient to prove that for any three neighboring points $n-1$, n, $n+1$, the following formula holds

$$G^*(t_0,n)-G^*(t_0,n-1)>G^*(t_0,n+1)-G^*(t_0,n) \tag{4.30}$$

Let $\tilde{t}_{1,n+1}$ be the optimal information exchange policy for $G(t_0,n+1)$, i.e. $G^*(t_0,n+1)=G(t_0,\tilde{t}_{1,n+1})$. Then

$$G^*(t_0,n+1)-G^*(t_0,n)=G^*(\tilde{t}_1,n)-G^*(t_0,n)+c_t(\tilde{t}_1-t_0-\beta)$$
$$-\frac{c_t+c_r}{r}(\tilde{t}_1-t_0)(1-\exp\{-k\mu(D_u-t_0)\}-c_\beta. \tag{4.31}$$

$$G^*(t_0,n)-G^*(t_0,n-1)\geq G^*(\tilde{t}_1,n-1)-G^*(t_0,n-1)+c_t(\tilde{t}_1-t_0-\beta)$$
$$-\frac{c_t+c_r}{r}(\tilde{t}_1-t_0)(1-\exp\{-k\mu(D_u-t_0)\}-c_\beta. \tag{4.32}$$

By (4.29), $G^*(\tilde{t}_1, n-1) - G^*(t_0, n-1) > G^*(\tilde{t}_1, n) - G^*(t_0, n)$. Comparing the right hand sides of (4.31) and (4.32), it is evident that $G^*(t_0, n) - G^*(t_0, n-1) > G^*(t_0, n+1) - G^*(t_0, n)$. This concludes the proof. □