ABSTRACT

ANALYSIS OF COSTS AND DELIVERY INTERVALS FOR MULTIPLE-RELEASE SOFTWARE

by
High-Way Sun

Project managers of large software projects, and particularly those associated with Internet Business-to-Business (B2B) or Business-to-Customer (B2C) applications, are under pressure to capture market share by delivering reliable software with cost and timing constraints. An earlier delivery time may help the E-commerce software capture a larger market share. However, early delivery sometimes means lower quality. In addition, most of the time the scale of the software is so large that incremental multiple releases are warranted.

A Multiple-Release methodology has been developed to optimize the most efficient and effective delivery intervals of the various releases of software products, taking into consideration software costs and reliability. The Multiple-Release methodology extends existing software cost and reliability models, meets the needs of large software development firms, and gives a navigation guide to software industrial managers. The main decision factors for the multiple releases include the delivery interval of each release, the market value of the features in the release, and the software costs associated with testing and error penalties. The input of these factors was assessing using Design of Experiments (DOE). The costs included in the research are based on a salary survey of software staff at companies in the New Jersey area and on budgets of software development teams. The Activity Based Cost (ABC) method was used to

determine costs on the basis of job functions associated with the development of the software. It is assumed that the error data behavior follows the Non-Homogeneous Poisson Processes (NHPP).

ANALYSIS OF COSTS AND DELIVERY INTERVALS FOR MULTIPLE-RELEASE SOFTWARE

by
High-Way Sun

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Industrial Engineering

Department of Industrial and Manufacturing Engineering

May 2002

APPROVAL PAGE

ANALYSIS OF COSTS AND DELIVERY INTERVALS FOR MULTIPLE-RELEASE SOFTWARE

High-Way Sun

4/22/02

Dr. George Abdou, P.E. CMfgE., Dissertation Advisor — Date
Associate Professor of Industrial and Manufacturing Engineering, NJIT

5/17/02

Dr. Athanassios Bladikas, Committee Member — Date
Associate Professor of Industrial and Manufacturing Engineering, NJIT

4/19/02

Dr. One-Jang Jeng, Committee Member — Date
Assistant Professor of Industrial and Manufacturing Engineering, NJIT

4/19/02

Dr. Jian Yang, Committee Member — Date
Assistant Professor of Industrial and Manufacturing Engineering, NJIT

4/15/02

Dr. MengChu Zhou, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author:	High-Way (Steve) Sun
Degree:	Doctor of Philosophy
Date:	May 2002
Date of Birth:	
Place of Birth:	

Undergraduate and Graduate Education

- Doctor of Philosophy in Industrial Engineering
  New Jersey Institute of Technology, Newark, NJ, 2002

- Master's degree in Computer Science
  New Jersey Institute of Technology, Somerset, NJ, 1999

- Master's degree in Industrial Engineering
  University of Wisconsin, Madison, Wisconsin, WI, 1990

- Master's degree in Engineering Management
  New Jersey Institute of Technology, Newark, NJ, 1988

- Bachelors degree in Mechanical Engineering
  National Central University, Chun-Li, Taiwan, 1985

Major: Industrial Engineering

Presentations and Publications:

Sun, H.W., Abdou, G. (2002). "Analysis of Multiple-Release Software Delivery Interval: a Case Study," IEEE Transactions of Reliability. Submitted 2002.

Sun, H.W., Zhou, M.C., Wolf, C. (2001). "A Methodology for Software Development Cost Analysis in Information-Based Manufacturing, " 2001 IEEE International Conference on Robotics and Automation. Seoul, Korea.

Sun, H.W., (1991), *Science and Policy Review 1991*, National Science Council, Taipei

Sun, H.W., (1992), *Science and Policy Review 1992*, National Science Council, Taipei

To my father, Shou-Wei Sun; my mother, Jye Wang; and my wife, Chi-Min Li with their love, I have completed this work.

ACKNOWLEDGMENT

I would like to express my deep gratitude and appreciation to my advisor, Dr. George Abdou, for his continuous guidance, support, encouragement, and inspiration throughout the course of this work.

My sincere thanks are also extended to Dr. MengChu Zhou, Dr. One-Jang Jeng, Dr. Jian Yang and Dr. Athanassios Bladikas for participating in the supervising committee of my dissertation and giving me valuable suggestions.

I am also truly grateful to Dr. Carl Wolf, Dr. Layek Abdel-Malek and Dr. Xiuli Chao for their patient guidance and encouragement during my first year of graduate study.

I would like to take this opportunity to thank Dr. Shui Y. Lee, Dr. Joseph O. Bergholm, Dr. John D. Williams, Mrs. Jennifer Wei, Dr. Chai-Yi Chou, Dr. Chin-Shen Lee, Dr. Jing-Chu Chou and Dr. Peter Ting for providing encouragement during research.

Thanks are also owed to my fellow graduate students, Mr. Yung-Chun Yen, Dr.. Mai-Mai Gou, Dr. HongZhou Wang, Dr. XingDong Wang, Dr. Xie-Mei Zhang, Dr. Tung Fu, Dr. Shin-Ying Huang and Dr. Ru Wu, and Ms. Janet M. Bodner for correcting the grammar.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF TABLES

(Continued)

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

NOMENCLATURE

a	Estimated total number of errors in the software
ABC	Activity Based Cost
AVE	Agile Virtual Enterprise
b	Contact fault detection rate
B2B	Business-to-Business Internet platform
B2C	Business to Customer Internet platform
C_1	Software test cost per unit time
C_2	Cost of removing each error per unit time during testing
C_3	Error penalty cost per unit time.
C_4	Cost of software installation
C_r	Slope of debug cost increase; the default value set at 0
CGI	Common Gateway Interface
CMM	Capacity Maturity Model
CORBA	Common Object Request Broker Architecture
D	The decision point
Data Commonality	Use of standard data structures and types throughout the program.
DOE	Design of Experiment
E (T)	Expected total cost of a software system by time T
E-commerce	Electronic commerce through internet.
Error Seeding	Manager introduced errors and asks the testers to find the errors
Error Tolerance	Damage that occurs when program encounters an error.

Expandability	Degree to which architectural, data, or procedural design can be extended.
Failure Rate Model	The probability of software error happened under specifically condition
Generality	Breadth of potential application of program components.
Instrumentation	Degree to which the program monitors its own operation and identifies errors that do occur.
m(T)	Expected number of errors to be detected by time T
Markov Structure	Markov decision process that software condition transition from module I to module J
MLE	Maximum Likelihood Estimate
Modularity	Functional independence of program components.
MR	Modification Request
MR Status	Action of the MR
MTBF	Mean Time Between Failures
MTTF	Mean Time to Failure
NHPP	Non-Homogeneous Poisson Process
N-Version Programming	1,2,...N independent groups use different algorithms developing software programming applied in the same function.
Operability	The ease of operation of a program
P_i	Release profit of the software at time Ti
Software Error	In this research, software functions do not follow system requirements or customers' needs.
Software Maintenance	In this research, software is maintained following the Total Quality Control guide line.

Software Release Policy	Maximize the software profit
Software Reliability	The probability that software will not fail for a specified period of time under Specified conditions.
SQA	Software Quality Assurance
R(x/T)	Reliability function of software by time T for a mission time x
x	Mission time
T	Software release time
T_1	First software release time
T_i	The second and later release point I=2,3,N
TQM	Total Quality Management
Traceability	The ability to trace a design representation or actual program component back to requirements.
U_y	Expected time to remove an error during testing phase, which is E (Y)
WWW	World Wide Web

CHAPTER 1

INTRODUCTION

The Multiple-Release model presented here describes software deployment behavior. Software project managers prioritize market values and set the software delivery interval time based on market needs and company resources. All software features may not be delivered at once, but rather a few at a time and at set intervals. If the software market value is smaller than the development cost or the software reaches technological limitations, then the software will be retained in a maintenance mode and the limited resources will be assigned to serve existing customers.

The Multiple-Release model is based on the needs of software firms to build new large software platforms or to migrate existing software systems to new applications. This chapter introduces the business characteristics associated with the development of large software and especially large Internet software, the determination of the relevant cost and the definition of the pertinent reliability.

1.1 The Need for a Multiple-Release Model

From 1995-1998 only company information was shown on the Internet in most web sites. Since 1998, E-commerce websites have interacted with customers. It is estimated that by 2005 the core business processes will migrate completely from Intranets to the Internet. The market will be affected by E-commerce and industries will be reconstructed with the emergence of new intermediaries and new business models. Beyond these changing patterns in consumer behavior, the most long-term and profound impact is the structural changes occurring every day in business-to-business (B2B) relationships.

Through the blossoming of E-commerce, a supply chain can support better customer service and channel performance. The four major impacts on the supply chain are cost, quality, delivery and flexibility. The Intranet and Internet data networks inside the organization and among the supply chain partners efficiently link the buyer-seller relationships. Additionally, networks may be used to speed electronic market places characterized by more ephemeral transactions among buyers and sellers. Also plausible is the use of networks to strengthen existing commercial relationships and lock in partners by increasing the costs of switching to new trading partners.

Andrew Grove, CEO of Intel, predicts that "In five years all companies will be Internet companies or they won't be companies at all." For businesses to survive in this new economy they must set their own B2B platforms. The existing legacy computer software has been undergoing an accelerated evolution driven by the Internet. To remain competitive in the E-commerce environment, most companies must upgrade their existing internal and external software and information systems, and connect to the large and united World Wide Web (WWW) Internet family. The upgrade includes deploying integrated IT platforms, designing a new business model, designing Internet payment and systems security, and other processes.

However, the decision of an enterprise to move to E-commerce can be painful. It requires a substantial investment in software and hardware upgrades and the training of a large number of employees. The basic concern is whether the existing business functions can operate correctly and efficiently. Another concern is whether the customers can order all of the existing products correctly through the Internet platform. Will the order processes inside the organization still be traced and sent to the right places? Will the product inventories in warehouses be linked to the new Internet environment precisely without losing any inventory records and while operating more efficiently? Most importantly, the enterprise needs to succeed in its field. The competition is severe. Thus the deployment timing of a B2B system is a critical factor. Software reliability while migrating from the existing software environment to the new Internet environment is a key point of enterprise survival in the large software industry.

The dissertation focuses on the Internet software cost investment, software reliability analysis, and the use of a software reliability model to solve the modern large software cost allocation and optimal release time problems.

1.2 Software Cost Development

The quality of a software system usually depends on the length of testing and the choice of testing methodologies. Generally speaking, the longer the testing time, the more reliable the software is expected to be and the total cost of developing the software is expected to increase. On the other hand, if errors are not detected, the company's reputation and market share suffer. Testing is an efficient way to remove faults in software. It is impossible to remove all the errors, especially in large software that may consist of millions of lines of code. Thus, it is important to determine when to stop testing and when to release the software to the market. In defining important software cost factors, a cost model should help software developers and managers answer the following questions:

1. How should resources be scheduled to ensure the on-time and efficient delivery of a software product?
2. Is the software product sufficiently reliable for release?
3. What information does a manager of software development need to determine whether to release software undergoing testing?

1.3 Software Reliability Development

The existing methods used to determine software reliability are summarized and analyzed in Chapter 2. Most models treat the software as a finished product and are cost/reliability driven. In general, the software multiple release and lifecycle behavior have been investigated extensively by the researchers and the optimal resource allocation and delivery policies are based on these factors: the delivery interval, the error occurrence estimation, the learning factor of debugging, the cost of testing, the cost of fixing bugs and the cost penalty of hanging errors in the released version. It has to be mentioned also, that in an E-commerce software platform, the error tolerance is larger than that of aviation software or life support software. For example, if Yahoo's shopping features temporarily shut down ten minutes to fix a software bug, it may cause some financial loss, but if life support or aviation software is not available for ten minutes, a life may be lost or a plane may crash.

1.4 Dissertation Overview

This dissertation is organized into eight chapters, which incorporate the overall methodology, analysis, and design as illustrated in Figure 1.1.

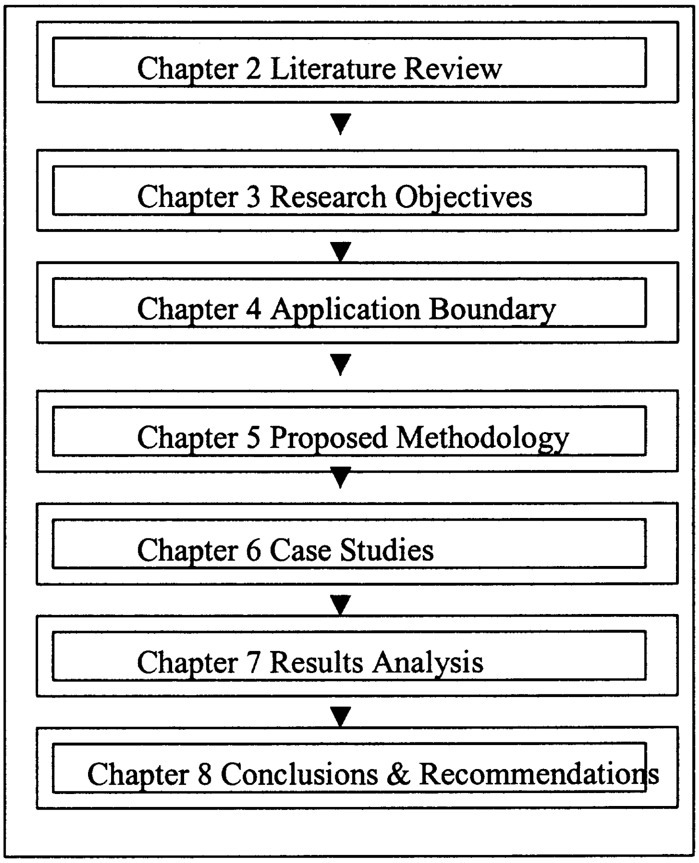
Chapter 2 Literature Review
Chapter 3 Research Objectives
Chapter 4 Application Boundary
Chapter 5 Proposed Methodology
Chapter 6 Case Studies
Chapter 7 Results Analysis
Chapter 8 Conclusions & Recommendations

Figure 1.1 Dissertation flow

Chapter 2 reviews and points out the limitations of the current literature and research conducted by different authors investigating software cost and reliability. Chapter 3 provides the objectives of this research and the proposed methodology to extend the research of existing software cost and reliability models. Chapter 4 defines the boundaries of the model application.

Chapter 5 describes the processes to implement the proposed model, including the ABC method to classify the software development cost items and the technique to collect software error data, using NHPP to estimate the number of errors and the error detection rate. A software cost decision is implemented. The design of experiments methodology is applied to analyze the result. Finally there is a discussion of the software delivery decision and significant factors analysis.

Chapter 6 applies the proposed method to two leading industrial companies, including the X Juice Company and the Y Service Provision Company whose software development process data are used.

Chapter 7 analyzes the data from Chapter 6, deciding on the significant factors that impact the decision. The software delivery policy is derived. The outcomes are also compared with existing software cost models created by other authors. Chapter 8 summarizes the conclusions and suggests directions for future research.

CHAPTER 2

LITERATURE REVIEW

There are many research papers devoted to software reliability, software cost and software quality control already published in leading reliability and software journals. From 1972 till now, many researchers have contributed to these fields. Software reliability is evaluated on the basis of the Software Quality Assurance (SQA) and Total Quality Management (TQM) concepts.

Crosby (1979) developed the Software Quality Assurance concept, which is an "umbrella activity" that is applied throughout the software engineering process. Cavano and McCall (1978) developed a checklist used to assign scores to a set of quality factors. Schulmeyer and McManus (1987) concluded that the scope of quality assurance responsibility might best be characterized by paraphrasing a once-popular automobile commercial: "Quality is Job #1." Musa, Iannino and Okumoto (1984) mentioned that "To model software reliability one must first consider the principal factors that affect it: fault introduction, fault removal, and the environment."

TQM can be defined as the application of quantitative methods and human resources to improve the materials and services provided as inputs to an organization and to improve all of the processes within the organization.

2.1 Review of Software Reliability and Cost Papers

Software reliability, which is part of SQA and TQM, is now an established independent research field. Since 1972, research has been conducted to study the reliability of computer software. As software systems have become more and more complicated to design and develop, intensive studies have been carried out to increase the chance that a software system will perform satisfactorily in operation. Mills (1972), then Knight and Ammenn (1985) first presented the error-seeding model, where the manager introduces errors and asks the testers to find the errors. By finding the induced errors, the total number of unknown errors can be estimated. A program is randomly seeded with a number of known errors and then the program is tested. The probability of finding j real errors of a total population of J unknown errors can be related to the probability of finding k seeded errors from all K errors embedded in the code.

Goel and Okumoto (1979) introduced the time-dependent error detection rate for software and other performance measures of a software system. They used a technique which based on the existing error detection rate obtains the probability of detecting the next error during the next period of time. Sahin (1999) and Yamada (1998) used the Markov structure to get the transition matrix Q_{ij}, that is, the probability that the software will go from state i to state j. Currently, the Non-Homogeneous Poisson Process (NHPP) is widely used by software engineers and designers. Goel and Okumoto (1979), who first introduced the NHPP model, indicated that two factors, the number of errors and the testers' learning curve, should determine the error detection rate. Based on these two factors, the software reliability curve can be obtained. Other researchers have reported some interesting results using NHPP models. Yamada and Ohba (1983) observed that software error detection data as a function of time are often S-shaped. Recently, Normann and Pham (1999), and also Pham and Zhang (1999), found in their two papers that software reliability is also a function of the number of additional errors that are introduced during the test period. Testers improve their testing skill and their error detection rate over time, and the researches use the error numbers and the testers' learning curve to derive reliability parameters.

Many software reliability researchers, for example Pham (1994), have shown interest in other related fields, such as multiversion programming for security reasons in some life critical software. To prevent software failure caused by unpredicted conditions, different programs are developed separately, preferably based on different programming logic, algorithms, languages, coding methods, etc. These diverse programs are normally utilized in a set of recovery blocks of N-version programming.

Fault-tolerance is a widely used technique to increase the reliability of software coding. Abdou (1990) and Arlate (1990) studied a fault-tolerance model and applied an electronic components' reliability concept to software reliability. In general, fault-tolerant approaches can be grouped into fault-removal and fault-masking techniques. Fault-removal techniques can be classified into either forward or backward error recovery methods. The former aims to identify the error and correct the system state containing the errors. The latter corrects system errors before they can manifest themselves as faults. At present, software reliability modeling is considered to be an integral part of software quality and software engineering.

A bridge between software engineers in the industrial field and the statistical researchers in the academic field has been built, although the gap between practitioners and theoreticians is still waiting to be filled. Nordmann and Pham (1999) presented a software cost model and created a practical users' guide for software developers in industry. Kapur and Lamberson (1997) developed a model to be used for determining the optimum release time of new versions of a program package. Zhang and Pham (1998) discussed software costs taking into consideration the impacts of the software's life cycle, and imperfect debugging and its penalties. Zheng (2002) published a paper about the dynamic release policy for a software system and discussed the tradeoff between the failure penalty and the testing cost. Lee et al. (2000) presented the methodologies of priority setting with applications for software development. Pham and Zhang (1998) published a study of environmental factors and software reliability. Some of the factors, like program complexity, programmers' skills, testing coverage, testing effects, and the testing environment were considered to be key factors that impact software quality.

Sahin and Zahhedi (1999) considered the warranty policy and upgrade cost from the industrial point of view. Wall (1997) also presented the benefits of software maintenance. Furyyamma (1993) considered the mental stress impacting software development and testing from the human factors point of view.

Cost is the most important issue. Hansen & Mowen, [1999] used Activity Based Cost (ABC) analysis. Pham and Nordmann (1999) pointed out some errors in detection cost and testing cost but, they simply assigned a constant cost number to all factors. Tsay (1999) considered the supplier and customers as a whole system. Under this system, the customers can be guaranteed good quantity, good price and on-time delivery by the supplies. Shin, Collier and Wilson (2000) discussed four factors that can be used to judge quality: suppliers cost, product quality, delivery time and flexibility.

Price policy is also an important field of research. Zhao and Zheng (2000) investigated a pricing policy that could improve revenue for perishable assets. Alfredsson and Verrijdt (1999) modeled on emergency supply option to shorten the lead-time and get optimal benefits. The issue of choosing supplier alliance partners is also important. McCutcheon and Stuart (2000) ranked suppliers' quality to suggest a good alliance. An Agile Virtual Enterprise (AVE) in the supply chain may also play an important role. Wu et al. (1999) and DeVor et al. (1997) pointed out that agile manufacturing is a new concept used to represent the ability of a producer of goods or services to thrive in the face of continuous change.

Lin and Chen (1993) and Ashrafi, Berman, and Cutler (1994) found that fault tolerant software uses redundancy to improve reliability, but such redundancy requires additional resources and tends to be costly. Therefore, the redundancy level needs to be optimized within a software system under the assumption that functionally equivalent software components fail independently. An illustration is the tradeoff between the cost of using N-version programming and improved reliability for a software system. Major assumptions of application of the N-version programming are:

1. Several versions of each module, each with an estimated cost and reliability, are available;
2. These module versions fail independently.

Optimization models are used to select the optimal set of versions for each module such that the system reliability is maximized and total cost remains within budget. Maghsoodllo, Brown and Lin (1992), present the reliability cost analysis of an automatic prototype generator. The two models of software testing were compared on a cost and reliability basis. This extends the work of Maghsoodllo, Brown and Lin (1992), in which a cost model was developed to obtain optimum software testing based upon traditional testing.

Smidts, Stutzke and Stoddard (1998) considered software reliability modeling to make early predictions. Models for predicting software reliability in the early phases of development are of paramount importance since they provide early identification of cost overruns, software development process issues, optimal development strategies, etc. Khoshgoftaar (2000) presented a Bayes framework for the quantification of software process failure mode probabilities. This framework can be useful since it allows use of historical data that are only partially relevant to the software at hand. A summary of software reliability and cost models are listed in Table 2.1

Table 2.1 Summary of Software Reliability and Cost Models

Software Reliability MODEL	Description	Year	Author(s)
ERROR SEEDING	$P(k,N,n1,r) = \frac{\binom{n1}{k}\binom{N}{r-k}}{\binom{N+n1}{r}}$	1972 b	Mills, H.D
Non-Homogeneous Poisson Process	$m(t) = a(1-e^{-bt})$ a(t) = a b(t)= b $R(t) = e^{-m(t)}$	1979	Goel, A. and Okumoto, K.
NHPP Delayed S-Shaped	$m(t) = a(1-(1+bt)^{-bt})$ $a(t) = a$ $b(t) = \frac{b^2t}{1+bt}$	1983	Yamada, S.
NHPP Inflection S-Shaped	$m(t) = \frac{a(1-e^{-bt})}{1+\beta e^{-bt}}$ $a(t) = a$ $b(t) = \frac{b}{1+\beta e^{-bt}}$	1984	Ohba, M. and Yamada, S.
NHPP Pham-Nordmann	$m(t) = \frac{a(1+bt^{-bt})(1-\frac{\alpha}{b})+\alpha t}{1+\beta e^{-bt}}$ $a(t) = a(1+\alpha t)$ $b(t) = \frac{b}{1+\beta e^{-bt}}$	1999	Pham, H, and Nordmann, L.
NHPP Pham-Zhang	$m(t) = \frac{1}{1+\beta e^{-bt}}[(c+a)(1-e^{-bt})$ $-\frac{ab}{b-\alpha}(e^{-\alpha t}-e^{-bt})]$ $a(t) = c + a(1+\alpha t)$ $b(t) = \frac{b}{1+\beta e^{-bt}}$	1999	Pham, H.
Other NHPP	Based on different assumptions	1998 1996 1985 1992 1992	Gai, K.Y. Zhao, M. Yamada. S. Zhao, M. Xie, M.
Failure Rate Model	$\lambda(ti) = \theta[N - P(i-1)]$ $R(ti) = e^{-\theta[N-(P\ i-1)]ti}$	1979	Goel, A.

Table 2.1 Summary of Software Reliability and Cost Models (Continued)

Markov Structure	Markov decision process Qij= probability that transition from module i to module j fails	1998 1999	Yamada, S. Sahin, I.
General Fault Tolerance	The computer systems are capable of recovery from hardware or software failure to provide uninterrupted real-time service.	1990 1990 1990 1990 1992 1981	Abbott, J. Abdou, G.H. Arlate, J. Schenedewind, N. Munson, J. Moranda, P.
Software Maintenance	SQA – Total Quality Control	1998 1993	Bank, R. Paulk, M.
N-Version Programming	1,2, ….N Independent groups use different algorithms for developing modules.	1994	Pham, H.
Software Release Policy	C3 > C1+C2 C3- cost of errors that occur with no SQA program C1- is the cost of the SQA C2 - the cost of errors not found by SQA activity Maximize the software profit	1999 1992 1996 1999 1999 1997 1990 1992	Pham, H. Kapur, P.K. Pham, H. Zheng, S. Lee, M. Hou, R. Ohtera, H. Munson, J.

2.2 The Limitations of Existing Models

Based on the literature review, four imitations of existing models have been observed:

1.The authors of the reviewed papers consider only the testing cost and development cost. Many authors assigned an underestimated constant value for both costs.

2. The researchers papers have applied only one single release of the same software. Although, some papers also mentioned about the Multiple-Release, they are based on the different softwares deliveries. The authors treated the unique software release as only one shot, and then derived the optimal solution.

3. The software lifecycle concept was not addressed in the literature although most firms

in the software industry use it. The overall total cost of software development could certainly be wrong, if it is not taken into consideration.

4. In a few studies, only the roles of the testers and developers were considered. The importance of the system engineers in the designing stage and the customer support staff in the maintenance stage was ignored.

The Multiple-Release model presented in this dissertation was developed to fill the voiding in the existing literature on software cost, reliability and delivery intervals. Table 2.2 compares the proposed model with existing models.

Table 2.2 Comparison of Proposed Model with Existing Models

	Decision time D- discrete C- continuous	Discard Y- yes N- No	Warranty Period	Debug P- Perfect I - imperfect	Cost by R-R (x/t) N- Num	Decision Method	Release
Proposed Model	D	Y	Y	P	N	NHPP	Multiple
Pham (1999)	C	N	Y	P	R	NHPP	Single
Khoshgoftaar, T. (2000) 1	C	N	N	P	%	Bayesian	Single
Khoshgoftaar, T. (2000) 2	C	N	N	P	N	D-Tree	Multiple
Miller, H. (1972)	C	N	N	P	None	Prob.	Single
Zhang, X. (1998)	C	N	Y	P	R	NHPP	Single
Zheng, S. (2002)	C	Y	Y	P	N	NHPP	Single
Sahin, I. (1999)	D	Y	Y	I	Prob.	Markov	Multiple

Where
R(x/t) means reliability during (t, t+s) given that the last error occurs at time t.
Num. means the number of errors in the software.

CHAPTER 3

RESEARCH OBJECTIVES

To remove the limitations of existing models and make a more informed and accurate decision, an effective model had to be developed. Therefore, the main objectives of this dissertation are:

1. To identify all pertinent costs, including testing and development costs.
2. To determine optimal software delivery intervals and accordingly the life cycle of the software.

The following two major efforts were under taken to achieve these objectives.

3.1 Identification and Analysis of Software Cost Using the Activity Based Cost (ABC) Method

A number of software development costs are considered such as the cost of staff, hardware/software, office rental, etc, during the software development processes. The software development cost analysis uses the ABC cost management method to sort and classify all the necessary cost items. The salary of in-house employees, and the cost of outsourcing consultant agents are included. As a result, the outcome of the analyses can help project managers make the decision on whether to out-source or not the software. The main focus of the ABC analysis is to provide a more realistic representation of costs than the single release models used in the existing literature.

3.2 Development of A Multiple-Release Model

A Multiple-Release decision model was developed to improve on the existing a single release models. Obviously, large software cannot be delivered in just one release. Many new features may be added in every subsequent release during the life cycle of the software. The objective of the Multiple-Release model is to maximize the long run profit of the company. Different factors that impact the delivery decision of the multiple releases have been studied. The outcomes of the model can help software project managers in determining the optimum delivery interval and the significant factors impacting their decisions. In addition, a case study is presented; the model is applied using data from a leading software company. Finally, the Multiple-Release model is compared with existing software cost models. The procedure undertaken contains the following steps:

1. Use ABC analysis to determine costs.
2. Collect and analyze weekly error report.
3. Apply NHPP model to estimate the number of errors and the error detection rate.
4. Develop a cost model to determine the total software profit.
5. Conduct Design of Experiments (DOE) to determine:
    - Optimal decision: Release/ No Release (End of software development)
    - Optimum software delivery interval
    - Software life cycle
    - Significant factors affecting total profit

CHAPTER 4

SCOPE OF THE MULTIPLE-RELEASE MODEL

The scope of the Multiple-Release model, as applied to a medium or large software-developing company, includes the following factors: the software staff is above 40 people; the development time of each release is around one year; the interval time for each release is around three to six months, and the total development cycle time of the software is around two to five years. The Multiple-Release model monitors the software and assists managers to make decisions about the application features to be included, design of the software/hardware architecture, the coding and testing of the program, and repetition of the release cycle until the software is taken off the shelf and placed in the maintenance mode.

4.1 Decision Factors

A number of factors impact the total profit predicted by the Multiple-Release model. One factor is the time interval between releases. Some companies use a release interval of about three months while some may use annual releases. Three to six months may be a reasonable release interval for an Internet-based project. The error behavior and error detection rate are also important factors that impact the decision. Software errors are a key parameter for judging software quality. The error estimation and error-debugging rate impact the release time and may even cause a project termination decision. Testing and debugging costs also impact the decision.

The software staff salary floats with the software market. Right now (2001-2002) all the leading software companies like IBM and Y Service Provision Company, used here as a case study, are outsourcing their software projects to smaller companies or overseas to India and China where software staff salaries are relatively lower than in the US. Another main decision factor is the market value of the features; for example, the shopping platform provided before and after the Christmas season makes a significant difference in market value.

This dissertation discusses the software elements of an order system and an inventory database system. It focuses on the reliability of the user-interface transition, the database migration, and the reengineering of the data flow transaction processes. The cost estimation is based on the ABC methodology from the industrial standard. The software error detection is the errors that occurred in the time domain. The errors are based on the present industrial standard classifications from severity one (the system is corrupt) to severity four (minor user inconvenience).

4.2 Framework of the Multiple-Release Environment

Most traditional companies may use Visual Basic or Visual C as the language behind the graphical user interface (GUI) in the PC Microsoft environment, or they may use motifs in the Unix X-windows environment. The application middleware may come from middleware vendors or coding by C or C++. The typical backend database, DB3, shown in Figure 4.1, is an example of the high level of server-client architecture.

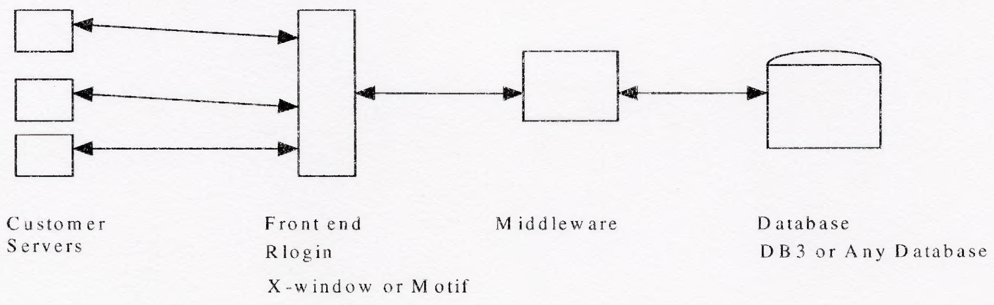

Figure 4.1 Traditional computer server-client architecture

As shown in Figure 4.2, E-commerce order systems today use a newer graphical user interface (GUI) based on Java and HTML. The companies buy middleware, as usual, from the middleware software vendors. The common database systems are ORACLE or Informix and Sybase.

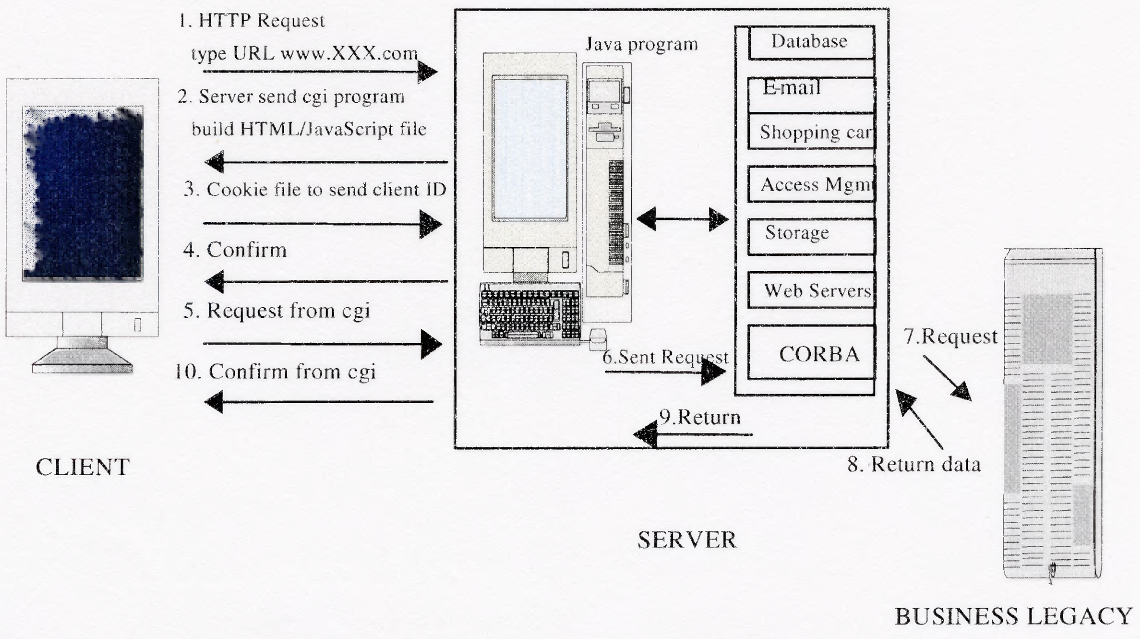

Figure 4.2 Example of software/hardware architecture for E-commerce

A company using a traditional computer system needs to migrate to E-commerce.

One example of the data flow steps is as follows:

1. Users from the web browser type the web site URL, for example "www.xxx.com".
2. The server side's directory /cgi/build.cgi sends the HTML/JAVA script file to the client and builds the browser display.
3. The user browser sends the client's cookie ID, which includes the name, value, expiration date, path, and domain that validate the security of the client.
4. The server confirms through Common Gateway Interface (CGI)
5. The client sends a request through cgi.
6. The server runs the JAVA program to contact the CORBA (Common Object Request Broker Architecture) communication interface.
7. The CORBA requests data from a business legacy system.
8. The legacy system confirms to CORBA.
9. The CORBA sends data to JAVA in the server.
10. The server sends the confirmation to the client through cgi.

This example includes 10 steps to process a basic service. Errors may happen in each step. This project studies development of E-commerce software, especially on the CORBA connection to the old legacy systems. The legacy system re-engineering reconstructs the legacy data migration to the new database schema.

4.3 Application Domain

This research concentrates on the reliability of software developed for errors PC or workstation platform. The product order systems will run on the different browsers of the Internet, such as, Netscape or Internet Explorer. This research aims to analyze the cost investment and reliability performance review. The application domain is limited to the B2B or business-to-customer B2C order systems. The goal of such systems is to correctly place an order from the Internet GUI, correctly process the order through the internal organization, and correctly trace and efficiently handle the inventory.

4.4 Dimensions and Tolerances

The Capability Maturity Model (CMM) for software was produced for the Software Engineering Institute (SEI) by a dedicated group of people who spent many hours discussing the model and its features and then documenting it in the two versions of CMM. The CMM is the major technique used by large IT companies to improve software quality and has the following characteristics:

- It is based on actual practices;
- It reflects the state of the art;
- It is publicly available;
- It is documented; and
- It reflects the needs of individuals performing software process improvement and software process appraisals.

A software reliability model is applied to the Y Service Provision Company. It includes all the errors associated with the new coding using Java, the new PC platform, the new relational database structure and, business data migration to the new platform. The tolerances are dependent on a company's policy. Some companies may set a policy of 100 to 1, asking the system tester to find as many errors as possible before the program is released, considering the penalty of finding 100 errors by testers the equivalent of one error found by a customer.

The terminology used in the existing literature is not consistent. The terms fault, error, failure, bug, mistake, malfunction, and defect are often used interchangeable. Software is said to contain a fault if, for some input data, the output result is incorrect. Although the definitions of fault may vary for different software and different situations, faults always exists in parts of a software and can be removed by correcting the erroneous part. By software faults the researchers generally mean all those parts in the software that may cause problems. Each execution of the software program where the output is incorrect, constitutes a software failure. A failure may be caused by a software fault or by another reason, such as a human mistake or hardware failure. Thus, failure and success are two different possible stages of the output. During the software-testing step, programs are invoked and errors are found. Each incorrect output may be counted as a failure. Faults that caused the failure are identified and removed. After faults are removed, the reliability of the software increases.

4.5 Constraints

The usefulness of the Multiple-Release model is limited to large commercial software. It is limited on the project budget and delivery timing, the budget sometimes is limited below 20 millions dollars; and the delivery time is within 5 years. Also the software project must be well managed. It should pass at least the second level of the CMM so the software processes are predictable and well documented. As imperfect human beings write the software, some errors can be expected. Miscommunications among managers, system designers, developers and testers may also cause some errors. Budget limitations force some different decisions to be made, e.g., how large and how many portions of the existing software need to be changed, and whether to outsource to software consultanting firms or to develop it by the company software staff.

The final goal within the project budget is to find the best delivery timing with a tradeoff between software reliability and market need. To achieve lasting results from process improvement efforts, it is necessary to design an evolutionary path that increases an organization's software process maturity in stages. The CMM orders these stages so that improvements at each stage provide a foundation on which to build improvements undertaken at the next stage.

A *maturity level* is a well-defined evolutionary plateau toward achieving a mature software process. Each maturity level comprises a set of process goals that, when satisfied, stabilize an important component of the software process. Achieving each level of the maturity framework establishes a different component in the software process, resulting in an increase in the process capability of the organization. Organizing the CMM into five levels prioritizes improvement actions for increasing software process maturity.

The five levels can be briefly described as:

1. Initial: The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
2. Repeatable: Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3. Defined: The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved version of the organization's standard software process for developing software.
4. Managed: Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
5. Optimizing: Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

These five levels reflect the fact that the CMM is a model for improving the capability of software organizations. The priorities in the CMM, as expressed by these levels, are not directed at individual projects. A troubled project might well prioritize its problems differently from the taxonomy given by the CMM. Its solutions might be of limited value to the rest of the organization, because other projects might have different problems or be unable to take advantage of its solutions because they lack the necessary foundation to implement the solutions. The CMM focuses on processes that are of value across the organization.

4.6 Operation Plan

The Multiple-Release models analyze the cost from the local software staff's salary survey. The error data are collected and maintained in the Y Service Provision Company Modification Request (MR) system or a similar device. The number of errors observed & fixed per unit period follow the Non-Homogenous Poisson Process (NHPP). The data analysis, the dissertation used the Design of Experiment (DOE) methodology to analysis the impacts of different factors.

Clearly, the new Internet software platform has to operate at least the existing business functions. Moreover, a company can take advantage of this opportunity to simplify the existing operating processes and upgrade their database management systems. From the research point of view, a final result may come to a formula or guidelines to a similar enterprise. The manager can use them to judge the best strategy. The managers can increase the investment and change developer salaries in the job market. These actions will change the scale of the software expensive costs.

CHAPTER 5

SOFTWARE MULTIPLE-RELEASE METHODOLOGY

The main reasons for developing this model are the vast scope and the profit-oriented behavior of most application software. To obtain software development costs, an ABC analysis has been conducted in which the expenses of all activities are classified. Then, the total expenses of each activity group are calculated. This calculation is the foundation for the subsequent Multiple-Release model, especially the testing cost activity function of the software.

Usually, software project managers decide the release intervals and release features each time, and the Multiple-Release model provides a decision mechanism to help the project managers decide optimal delivery timing. The model also helps project managers analyze the impact of different decision factors. In addition, the model helps with the critical decision on whether to discard certain features if the error penalty is larger than the delivery profit.

5.1 ABC Analysis of Software Costs

The cost analysis is based on ABC the methodology by Hansen and Mowen (1997). The first stage of ABC is to identify activities and then get the cost information from the associated individual activities. Then the cost is separated into homogeneous sets.

To streamline the process, activities are grouped in homogeneous sets based on similar characteristics: a logical relationship, and the same consumption ratios for in whole project. The collection of overhead costs associated with each set of activities is called a homogeneous cost pool. Once the pool is defined, the cost per unit is determined. This is called the pool rate. Computation of the pool rate completes the first stage.

Thus, the first stage produces four outcomes: activities are identified, costs are assigned to activities, related activities are summed to defined homogeneous cost pools, and pool (overhead) rates are computed. Then, the overhead assigned from each cost pool to each project is computed as follows: Applied overhead (to a project) = Pool rate x Activity usage. The ABC analysis uses the working characteristics of the software staff to separate different costs into different categories, including supervision, system engineering, and software development and staff overhead. The cost-driving factors are that to determine total cost are considered to be :

Staff has the following characteristics:

A. Level of education.

B. Experience with application.

C. Knowledge of languages.

D. Familiarity with the infrastructure environment.

In performing its functions, the staff engages in six entities as shown in Figure 5.1. These terms are called: define, design, construct, test, install, and maintain.

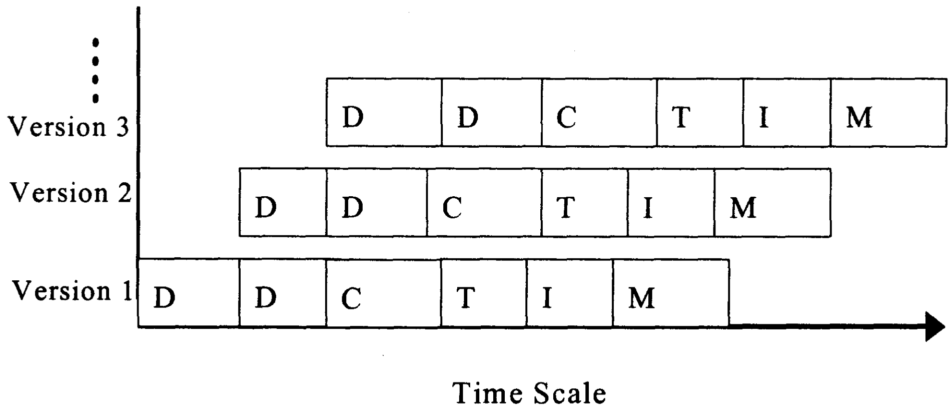

Figure 5.1 Software job functions of software staff

For example, in a B2B software development project, the six entities can be described as in Table 5.1.

Table 5.1 Job Descriptions of B2B Software Development

Activities	Description
Define	Business-to-Business Internet platform and the internal requirements from customer service, continuous replenishment program, the logistics department, and the dispatch center. In addition, the outside environment should consider the retailers' order interface, the shipping contractors, etc.
Design	Software/hardware architecture and the data flow transmission design.
Construct	Coding processes by the programmers.
Test	Software system tests of different features and in different scenarios.
Install	Installation inside X Juice Company and all members in the supply chain.
Maintain	Debugging the errors and keeping the system running.

The hardware & software annual cost includes hardware, software licenses and office space used by the development staff. The software/hardware cost is influenced by:

A. Compatibility between development computers and target computers.

B. Computing conditions during the development.

C. Availability of the development computers.

D. Size of the development database.

E. The response time of the development computers.

F. Equipment available in the network place.

5.2 Multiple-Release Decision Model

The assumptions of the Multiple-Release model and its implementation process are shown below:

1. Due to the vast scope of the software, it is being developed in Multiple-Releases.
2. The release relationship is release 1 -> release 2 ->...release N Release i must be delivered before release j. ($i, j \leq N$ an i < j)
3. To simplify this model, release N can only be delivered on time at T_N or during the next delivery interval T_{N+1}.
4. Since deployment of a new release requires that the business server be temporarily shut down, deployment is done late at night or a weekend to prevent interference in the normal course of business.

5. If the release cost is larger than the release profit, the software project should be terminated.
6. The cost to perform testing is proportional to the testing time.
7. The cost to remove errors during the testing phase is proportional to the number of errors and constant.
8. The testing starts at time 0, the first due date delivery point is set to Time Td, and then Ti is the interval of the next delivery.
9. The software at the decision point depends on the error condition to decide which delivery point to go to obtain the maximum company profit.
10. Each release of the software the software firm obtains the same profit(Pi).

A detailed description of is shown in Figure 5.2:

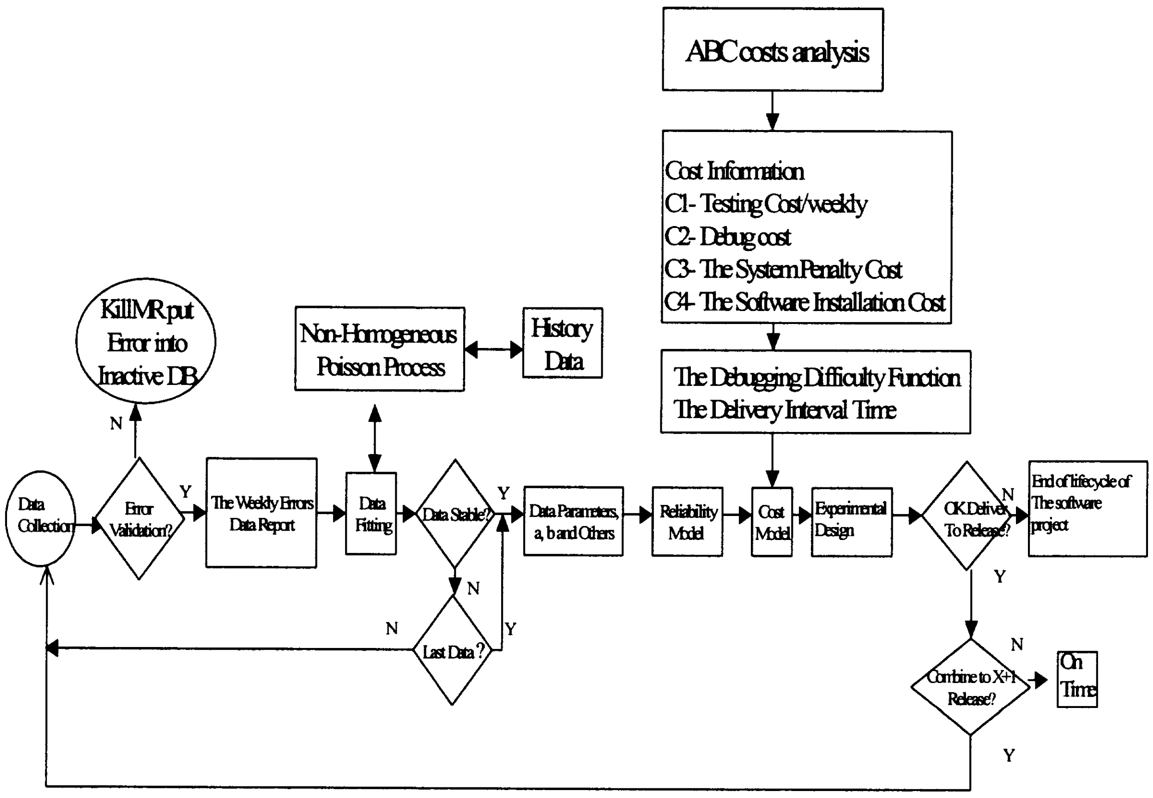
ABC costs analysis
Cost Information
C1- Testing Cost/weekly
C2- Debug cost
C3- The System Penalty Cost
C4- The Software Installation Cost
KillMRput Error into Inactive DB
Non-Homogeneous Poisson Process
History Data
The Debugging Difficulty Function
The Delivery Interval Time
Data Collection
Error Validation?
The Weekly Errors Data Report
Data Fitting
Data Stable?
Data Parameters, a, b and Others
Reliability Model
Cost Model
Experimental Design
OK Deliver To Release?
End of lifecycle of The software project
Last Data ?
Combine to X+1 Release?
On Time
Y
N

Figure 5.2 Flowchart of Multiple-Release model

5.2.1 Data Collection

The testers operate the system and find errors during the system test period. Certain types of errors can be found. The errors can be associated with data incompatibility and format discrepancies, program logic errors, and anything else that does not meet the requirements and customers' needs. The Modification Request (MR) is generated. The MR information includes:

1. Product: The software name
2. System: Customer who uses the software system
3. Subsystem: Software module of the whole software system
4. Release Detected: Release number
5. MR Severity: the importance of the MR

   The valid error definitions were divided into 4 categories:

   Severity 1: System is non-operational.

   Severity 2: Major functions of the system are unavailable, unusable, or system performance is well below normal operation.

   Severity 3: Minor functions of the system are unavailable. (Bug fix)

   Severity 4: Minor deficiency with minimal impact to users. (Enhancements)
6. Required Date: Due date to fix this MR
7. Originator: The tester who finds the MR
8. Abstract: Title of the description
9. MR Description: Detailed description.

5.2.2 Error Validation and Kill MR

The MR board members, including the software managers, the system engineers, system developers and system testers, validate the errors to judge if they are valid errors or just the users' operation errors. A routine software MR review takes place at a board meeting, including all the managers, project leaders and coordinators; they decide whether the MR is valid. Sometimes it is just that the testers don't know how to operate the system; in that case, it will be considered as a user training or user-friendly issue.

After the board members decide the validity of the MRs, they assign the MR to developers to fix the related code. If it is not a valid MR, the board members decide to "kill MR" without taking any action. In the Weekly Data Report, the board members classify the causes of the errors. For example, Pham (1998) has a list of all possible software errors (see Table 5.2).

Table 5.2 Catalog of All Possible Errors

1. Program complexity	2. Documentation	3. Amount of programming effort
4. Programmer skills	5. Program workload	6. Program categories
7. Testing coverage	8. Testing tools	9. Work standards
10. Testing effort	11. Programming organization	12. System software
13. Testing environment	14. Domain knowledge	15. Volume of program design documents
16. Frequency of specification change testing methods	17. Difficulty of programming	18. Development team size
19. Requirement analysis	20. Design methodologies	21. Programming language
22. Percentage of reused code	23. Human nature	24. Processor
25. Relationship of detailed coding	26. Development management	27. Telecommunication device
28. Design and requirement	29. Testing resource allocation	30. I/O device

The research can extend the list to subtitles: for example, the I/O device errors can be computer servers to servers or computer servers to databases. However, if more than three users report the same user-friendly issue, it will be considered a bug needing to be fixed. The error validation processes are shown in Figure 5.3.

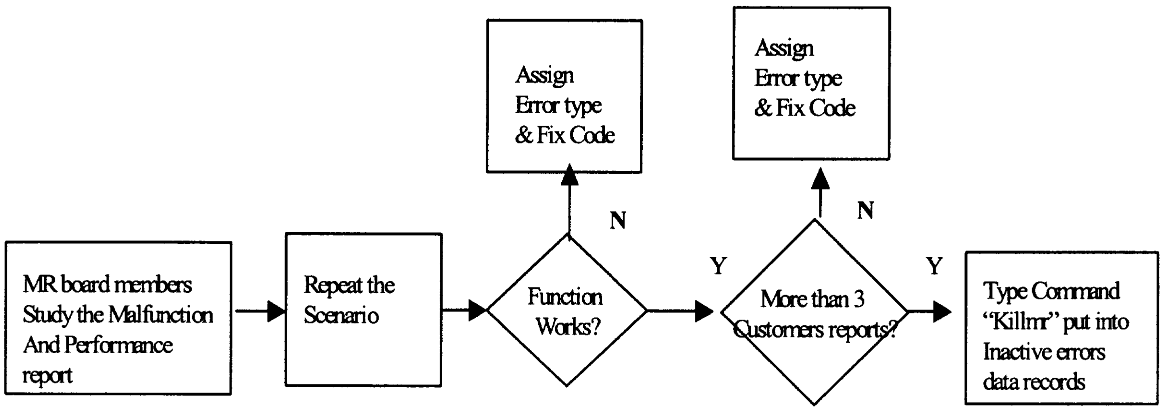
MR board members Study the Malfunction And Performance report
Repeat the Scenario
Function Works?
N
Assign Error type & Fix Code
Y
More than 3 Customers reports?
N
Assign Error type & Fix Code
Y
Type Command "Killmr" put into Inactive errors data records

Figure 5.3 Error validation processes

5.2.3 Monthly Data Reports

The programmers usually fix the errors right away and document the summary weekly. The number of errors is accumulated weekly. The project is based on multiple releases, so the errors can be attributed to separate releases. The delivery time is very long, so monthly data reports are used. The monthly report can be obtained from the MR system; the processes are shown in Figure 5.4.

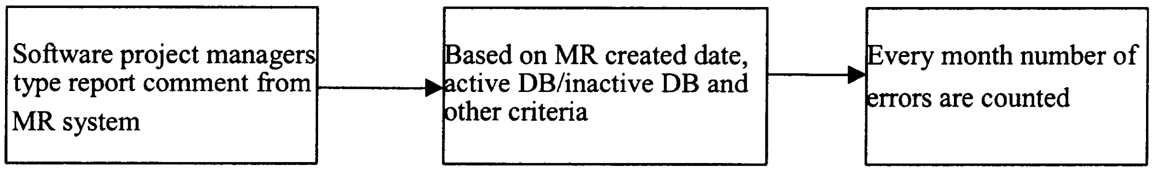
Software project managers type report comment from MR system
Based on MR created date, active DB/inactive DB and other criteria
Every month number of errors are counted

Figure 5.4 Monthly errors data report processes

5.2.4 Data Fitting

Goel (1985) divided the existing models into four groups:

1. Times between failure models –Markov model
2. Failure count models – Non-Homogeneous Poisson Process models
3. Fault Seeding models – Any models in which a number of faults are seeded and a capture-recapture technique is used in modeling and testing.
4. Input domain based models - The models input a test scenario into the software and get the sampling result. The result is used to construct a formula of the software reliability.

5.2.4.1 Data Fitting – Historical Data. The seed data of the estimation total software errors "a" and estimation error detection rate "b" are input in the data fitting regression method. The seed data are the initial values of the regression method to get the final estimated "a" and "b". The seed data come from the judgment of the project manager.

5.2.4.2 Non-Homogeneous Poisson Process. To study the software reliability, the first step should be an analysis of the error data. This research uses the Non-Homogeneous Poisson Process (NHPP), which is the most commonly used model in industry and research. The exponential NHPP model is based on the following assumptions:

1. All faults in a program are independent.
2. The number of failures detected at any time is proportional to the current number of faults in a program. This means that the probability of detective a fault is constant.

3. If a fault is detected and isolated, it is removed prior to future testing.

4. In the process of removing an error, no additional errors are introduced.

5.2.4.3 Data Fitting – NHPP Goel Model. If the testing errors follow a monotonic decrease, the data can apply to the NHPP Goel model.

For the Type I data: the accumulated errors data

Assuming that the cumulative number of detected errors yi in a given time-interval (0,ti) where I=1,2,.n and 0<t1<t2<..<tn, the estimate of parameters a and b using the Maximum Likelihood Estimation (MLE) method can be obtained by solving the following equations from Pham and Zhang (1999) simultaneously:

$$a=\frac{y_n}{(1-e^{-bt_n})} \qquad (5.1)$$

$$\frac{y_n t_n e^{-bt_n}}{1-e^{-bt_n}} = \sum_{k=1}^{n} \frac{(y_k - y_{k-1})(t_k e^{-bt_k} - t_{k-1} e^{-bt_{k-1}})}{(e^{-bt_{k-1}} - e^{bt_k})} \qquad (5.2)$$

Where

a	Total number of software errors
b	Error detection rate
y_i	The cumulative number of detected error in time (0, ti)

5.2.4.4 Data Fitting - The S-Shaped NHPP Model. The error detection data pattern is s-shaped as pointed. The error detection rate is slow at first. Then, the detection rate goes up as the testers become more familiar with the system. Afterwards, the errors are difficult to find, and the detection rate goes down. This is a typical learning process and it should not be surprising that it generates S-shaped results. The delayed *S*-shaped NHPP model (Yamada, 1983) is used in this dissertation.

Assume $b(t) = \frac{b^2 t}{bt + 1}$ where b is the error detection rate in the steady state. The mean value function can be obtained as $m(t) = a[1-(1+bt)e^{-bt}]$ which shows an S-shaped curve. This is called the delayed S-shaped NHPP model for such an error detection process, in which the observed growth curve of the cumulative number of detected errors is S-shaped.

For Type I data, the estimate of parameters a and b using the MLE method can be obtained by solving the following equations Pham and Zhang (1999) simultaneously:

$$a = \frac{y_n}{[1 - (1 + bt_n)e^{-bt_n}]} \tag{5.3}$$

$$\frac{y_n t_n^2 e^{-bt_n}}{[1 - (1 + bt_n e^{-bt_n})]} = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})(t_i^2 e^{-bt_i} - t_{i-1}^2 e^{-bt_{i-1}})}{[(1 + bt_{i-1})e - bt_{i-1} - (1 + bt_i)e - bt_i]} \tag{5.4}$$

Where

a	Total number of software errors
b	Error detection rate
y_i	The cumulative number of detected error in time (0, ti)

5.2.5 Data Stability

In order to make the estimation errors "a" and the detection rate steady state, input the error data using the regression method in (5.3) and (5.4). First, input two months' error data to get the first a_2 and b_2, then input the first three months data to get a3 and b_3, and so on. Use the n months' data to get an and b_n. If the estimated "a" and estimated "b" have not come to a steady state, in this dissertation set the data change within ± 5% is steady state, then input more data into the fitting program. If "a" and "b" become steady, then no more data are inputted. Then, the "a" and "b" are passed to the next step. If out of data the "a" and "b" still have not come to steady state, then choose the final a_n and b_n.

5.2.6 Reliability Model

From the Geol-Okumoto model (1979) the reliability function is as follows:

$$\hat{m}(t) = \hat{a}[1 - e^{-\hat{b}t}] \tag{5.5}$$

$$\hat{R}(x/t) = e^{-\hat{a}[e^{\hat{b}t} - e^{\hat{b}(t+x)}]} \tag{5.6}$$

$$\hat{m}(t) - Z\alpha\sqrt{\hat{m}} \le N(t) \le \hat{m}(t) + Z\alpha\sqrt{\hat{m}} \tag{5.7}$$

Where

$\hat{m}$	Maximum Likelihood Estimation (MLE) number of errors detected by time t.
$\hat{R}(s/t)$	MLE Reliability during (t, t+s) given that the last error occurs at time t.
$\hat{a}$	Total number of software errors
$\hat{b}$	Error detection rate
$Z\alpha$	Parameter α of the standard normal distribution.

5.2.7 Cost Model

Following are the definitions of the notations for the Multiple-Release model.

Notations

m(t) Expected number of errors to be detected by time t

a	Total number of software errors
b	Error detection rate
T_1	First software release time
Ti	Second and later release point i=2,3,…,N
C_1	Software test cost per unit time
C_2	Cost of removing each error per unit time during testing
C_3	Error penalty cost per unit time.
C_4	Cost of software installation
Cr	Slope of debug cost increase, the default value is set to 0
Pi	Release revenue of the software at time Ti
D	Decision point
Li	Software i release target time

The Multiple-Release model is considers of the following steps:

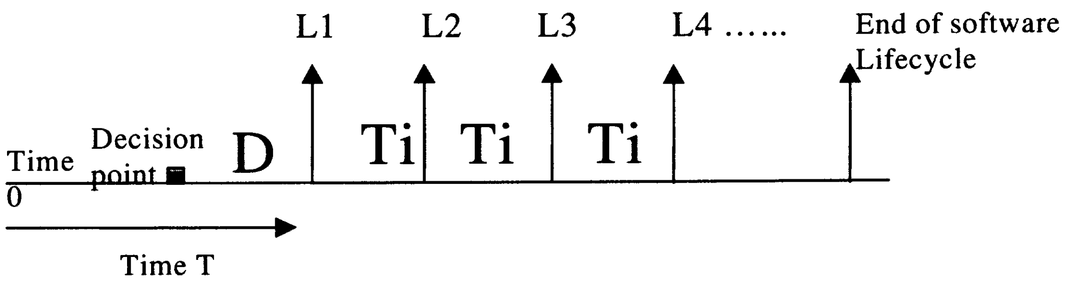

Figure 5.5 Release decision of the multiple releases

Start at the decision point, compute:

1) The cost to perform testing is given by:

$E(C_1)= C_1(D+Ti . N)$ where $N= 0,1,2,3$ (5.8)

The cost to remove all errors detected during the testing period is the product of removal removing time multiplied by the removal cost. When the removal out is more and more difficult, the removal time is assumed to be Tb + i. R for i=1,2,3,4 This means the first error takes Tb time, the second error takes Tb+1R, the third error takes Tb+2R, and so on.

The cost to remove the errors is:

E(C)= Cost. (Tb+ i. R) = C + i. Cr for i=1,2,3...

However, since the process , the cost for the period (0,T) should be computed as:

$$\begin{aligned}
&E[\sum_{i=0}^{N(t)}(C+iC_r)] \\
&=\sum_{n=1}^{\infty}\{E[\sum_{i=0}^{N(T)}(C+iC_r)].P(N(T)=n)\} \\
&=\sum_{n=1}^{\infty}\{E[\sum_{i=0}^{n}(C+iC_r)].P(N(T)=n)\} \\
&=\sum_{n=1}^{\infty}\{[nC+\frac{n(n-1)C_r}{2}].P(N(T)=n)\} \\
&=(C-\frac{1}{2}C_r)\sum_{n=1}^{\infty}n.P(N(T)=n)+\frac{1}{2}C_r\sum_{n=1}^{\infty}n^2.P(N(T)=n) \\
&=(C-\frac{1}{2}C_r)E(N(T))+\frac{1}{2}C_rE[N(t)]^2 \\
&=(C-\frac{1}{2}C_r)m(T)+\frac{1}{2}C_r\{Var(N(T))+(E[N(t)])^2\}
\end{aligned}$$

For a Poisson process, E (N (T))=Var (N (T))= m (T)

$$\begin{aligned}
&E(N(T))=Var(N(t))=m(T) \\
&=(C-\frac{1}{2}C_r)m(T)+\frac{1}{2}C_r\{m(T)+[m(t)]^2\} \\
&=Cm(T)+\frac{1}{2}C_r[m(T)]^2 \\
&=m(t)[C+\frac{1}{2}C_rm(T)]
\end{aligned}$$

E (C_2) =

$$m(T + Ti.N)[C_2 + \frac{1}{2}C_r.m(T + Ti.N)] - m(T - D)[C_2 + \frac{1}{2}C_r.m(T - D)] \qquad (5.9)$$

The penalty cost due to software failure and the wait for fixing the next release is:

E (C_3)= C3 [a-m (T+Ti)]. Ti/2 (5.10)

The delivery profit is:

The total revenue - total cost = net profit:

Pi - E1 (C_1)-E2 (C_2)-E3 (C_3)-C_4=

$$Pi - C_1(D + T_i.N) - \{m(T + T_i.N)[C_2 + \frac{1}{2}C_r.m(T + TiN)] - m(T - D)[C_2 + \frac{1}{2}C_r.m(T - D)]\} - \{C_3[a - m(T + Ti)].Ti/2\} - C_4 \qquad (5.11)$$

Assuming the decision time is only at the start point(time 0).

Total profit (t)= Pi - C_1 (T+Ti.N) - C_2 M(T+Ti.N) - C_3 [a-M(T+Ti.N)].Ti/2 – C_4

For M (t) = a (1- e^{-bt})

$\partial profit(t)/\partial Ti$ =

$C_1.N + C_2(a.b.N.\ e^{-b(T+N.Ti)}) - C_3\{(a.b.N.\ e^{-b(T+N.Ti)})Ti/2 + ½[a-m(T+Ti)]\}$ (5.11)

$\partial profit(t)/\partial Ti^2$ =

$- C_2.\ (a.\ b^2.N^2\ e^{-b(T+N.Ti)}) + C_3.\{\ (a.\ b^2.N^2\ e^{-b(T+N.Ti)}).Ti/2 + (a.b.N.\ e^{-b(T+N.Ti)}).1/2$

$-1/2.a.b.N.\ e^{-b(T+N.Ti)}\}$ (5.12)

when $\partial profit(t)/\partial Ti$ =0 and $\partial profit(t)/\partial Ti^2$ <=0 profit is the maximized.

There are $\left\lfloor \frac{lifeCycle}{Ti} \right\rfloor$ releases during the software lifecycle, and the total profit is the summation of these $\left\lfloor \frac{lifeCycle}{Ti} \right\rfloor$ releases.

$$\sum_{i=1}^{\left\lfloor \frac{\text{lifecycle}}{Ti} \right\rfloor} Pi - C_1(T+T_i.N) - \{m(T+T_i.N)[C_2 + \frac{1}{2}C_r.m(T+TiN)]\} - \{C_3[a-m(T+Ti)]Ti/2\} - C_4 \quad (513)$$

The total profit for a predetermined time period (Time) is:

$$Pi - C_1(T+T_i.N) - \{m(T+T_i.N)[C_2 + \frac{1}{2}C_r.m(T+TiN)]\} - \{C_3[a-m(T+Ti)]Ti/2\}$$

$$-C_4. \left\lfloor \frac{lifeCycle}{Ti} \right\rfloor . (\frac{Time}{LifeCycle}) \quad (5.14)$$

When the delivery interval time Ti is increased, the error penalty is increased, it takes longer for the corrected version to reach the customers. How, installation fees are reduced with fewer installations. These should be taken into consideration by the project manager when decide the optimal delivery interval Ti.

5.3 Design of Experiments

After the testers collect all the error data and costs and get the result of the total costs all are determined, the project manager needs to make a final decision on the basis of the final results of the Multiple-Release model options: 1. Deliver on time; 2. Postpone to next delivery; 3. Stop the software project. Six factors are considered to have an impact on profit, the final decision, and are following:

A The error detection rate/week

B The amount of software revenue decay rate/week

C The delivery interval in weeks

D The software test cost/week

E The cost of removing error during test

F The error penalty cost/week

The factorial experiment sets two levels of A, B, C, D, E, F six factors $2^6=64$ results in On-Time delivery. Also two levels of A, B, C, D, E, and F: $2^6=64$ results in Delayed delivery. The corresponding contrasts of the mean $\overline{Y}$, $(S/N)_i = -10.\log_{10}\frac{1}{n}(\sum_i 1/y_i^2)$ and $\log_{10}(S)$, $\overline{Y}$ is the mean of total profit, $\log_{10}(S)$ is the observations of the standard deviation estimate of variability of the particular factor, and the $(S/N)_i = -10.\log_{10}\frac{1}{n}(\sum_i 1/y_i^2)$ signal-to-noise ratio.

These three analyses can point out the significant factors under the designated conditions. The research uses a probability plot to analyze which factors are significant to change the three contrasts. The plot also can consider what is significant to change the profit. Probability plots use the notions of sample percentile and rank, which must be retrieved and digested. An observed data set is written x_1, x_2, $x_3 \ldots x_n$ in the order in which the observations are recorded. The notation indicates that the data set has n data points. The notation for the ranked data set, which represents x_1, x_2, $x_3 \ldots x_n$ rank order from smallest to largest, is $x_{(1)}<=x_{(2)}<=x_{(3)}<=\ldots<=x_{(n-1)}<=x_{(n)}$. The percentile is each ranked data point in the sample percentiles; x (i) is the $100(\frac{i-1/2}{n})th$ sample percentile.

Pair-wise comparison t-Test

The Multiple-Release model analyzes different policies to decide which one can generate maximum profit. It compares the following for

1. The On-Time delivery profit with the Delayed delivery profit.
2. Under a high market value dropped condition, the Delayed delivery profit is compared with On-Time delivery profit.
3. The 12-week interval time profit is compared with the 16-week interval time profit.

Another example of a test of a hypothesis is using the pair-wise comparison t-Test. There is a significant different on two condition ($\mu_1 - \mu_2$). The two samples are very heterogeneous with respect to their variances. A test on means is desired. The d_1, d_2, $d_3 \ldots d_n$ are the differences between 2 samples. This situation is often referred to as the

$$H_0 : \mu_D = 0$$
$$H_{1:} : \mu_D \neq 0$$

$$t = \frac{\bar{d}}{Sd / \sqrt{n}}$$

With a degree of freedom n-1 when n is the sample number of test data. The probability of a type I error is designed as α . Reject if $|t| \geq t_{1-\alpha/2}$ or $t \geq t_{1-\alpha}$ in the research case, two-tail situation, $|t| \geq t_{1-\alpha/2}$ is applied.

5.4 Software Delivery Decision

1. In the software multiple releases, after simulating the total costs, the project manager

needs to make a final decision. There are three decisions: 1. Deliver on time; 2. Postpone to next delivery; 3. Stop software development and terminate the project. If the on time delivery generates profit and the delayed delivery cannot generate significant improvement to the profit, On-Time delivery is selected. If the On-Time delivery generates negative profit or less than the one from delayed delivery, Delayed delivery is selected. If both On-Time and Delayed delivery generated a loss, the project manager can consider terminating the $N+1^{st}$ release after delivering N release. The software lifecycle is then stopped. Software will be off the shelf in maintenance mode. Based on the contract with the customer or a market strategy, project managers decide on the level this maintenance resources made available to maintain existing features of the software.

CHAPTER 6

CASE STUDIES OF SOFTWARE COST ANALYSIS AND THE MULTIPLE-RELEASE MODEL

Two cases are presented in this chapter. The first case analyzes the software cost of the X Juice Company which intends to move the existing service provision software platform to a new B2B Internet platform. This case catalyzes all the project expenses using the ABC method. At the same time, it compares in-house development costs with costs of out-sourcing the project to consultant firms.

The second case applies the Multiple-Release model to analyze a large re-engineering project at the Y Service Provision Company. In the second case, the existing systems are still running to handle the route business functions. After each software release, the new software systems and migrate the existing system functions to an Internet platform totally replace the old systems. This is a good example of a multiple release case in which the system migrates little by little to the new platform. The project has followed the Capability Maturity Model (CMM) control and Software Quality Assurance (SQA) guidelines. This assures that the code is reusable and the processes are well documented and repeatable.

6.1 X Juice Company Case Study

The case as presented here was published in Sun, H.W., Zhou, M.C., Wolf, C. (2001),"A Methodology for Software Development Cost Analysis in Information-Based Manufacturing", 2001 IEEE International Conference on Robotics and Automation. Seoul, Korea.

A Methodology of Software Development Cost Analysis

The long lead-time response is always a problem for supply chain distribution systems, especially in the perishable food industry and the rapidly advancing computer industry. An Internet based environment can support suppliers and customers with on-line information and can provide quick response time. This report is the result of a project between New Jersey Institute of Technology (NJIT) and the X Juice Company. One goal of the project was to use information technology to automate the present system and thus shorten the response time from seven to three days.

The existing structure uses an Electronic Distribution Interface (EDI), fax, telephone or Email to place orders. The project proposed an integrated Internet working environment for all members of the supply chain and focused on the cost analysis of the related software. Activity Based Cost (ABC) and lifecycle engineering methods were used to catalog all the cost expenses and determine the level of effort the software development takes. The cost analysis involves both in-house and outsourcing developments.

Case Introduction

Information-based manufacturing is critical if manufacturers are to compete intoday's global market. The development cost analysis becomes an important issue of industrial relevance for a software project involving both in-house development and outsourcing expenses. This work is motivated by the need to implement information-based manufacturing for the X Juice Company.

The project is based on the architecture of the company's Distribution Center. Juices that have a shelf life of approximately 65 days arrive by train and trucks from the regional customers come to the facility to pick up orders. This facility consists of an Automated Storage and Retrieval System, which is fully integrated into an Automated Warehousing System. To shorten the customers' order process time from seven to three days, is was proposed to use the Internet to streamline the distribution process.

According to Seybold et al. (1999), E-commerce is defined as the use of the Web, e-mail and other devices to streamline the business process, improve customer service time it was reduce cycle time, etc. To succeed in implementing an E-commerce strategy, proposed to use a team of about 97 skilled software developers. (57 regular employees in-house and about 40 contractors from an outsourcing software consultant house.)

The X Juice Company's current practice faces three problems: first, its Internet software is not connected to the entire supply chain; second, there is a lack of efficient and effective ordering policies; and third, there is a need to combine marketing strategies with inventory levels. Currently, X Juice Company managers control about 15%-20% of the inventory orders from retailers through the Company's Continuous Replenishment Program (CRP) system. Meanwhile, individual stores randomly place the other 80%-85% of orders, which are not under the Company's control. The X Juice Company customer service department administers orders from those individual customers.

An Internet interface between customer service and customers, from the supply chain perspective, can benefit both the customers and the warehouse. The advantage to the warehouse is that it is able to centralize demand information for individual stores as the company decides to ship more juice. The retailers benefit from in-time delivery and less stock-out penalties. This helps reduce random variation and hence demand uncertainties demand in the warehouse.

The second problem is the central ordering of juices that are shipped to the distribution center from the plant. The distribution center sometimes builds up an inventory of certain classes of juices that are close to their expiration date, and the company has to get rid of them either at a very low price with sales promotion or donate them to charity. A carefully designed and sophisticated coordination in ordering policies can reduce the occurrence of these problems and result in savings. One approach would be to create an incentive for the customers to synchronize their order function with X Juice Company. This is the so-called supplier-retailer coordination problem. A carefully designed coordinated system can benefit every player in the supply chain network. Shin, Collier and Wilson (2000) discuss the four factors to judge supplier quality: cost, quality, delivery and flexibility. This may require the design of contracts or cost sharing agreements with customers.

The third problem is how to combine marketing strategies with inventory levels and other factors. Marketing strategies such as sales incentives can influence demand. Foreseeing an inventory buildup problem, the company can use price as a tool to increase or reduce demand when insufficient inventory is on hand. F. Chen, et al. (2000) indicated that a company could use promotional initiatives to catch the emerging orders and gain larger profit.

The objective of this study was to evaluate the Internet software development cost and for the X Juice Company's Distribution Center. The method used is applicable to other industries. Section 6.1.1 Introduces the existing structure and explains the re-engineering model and cost analysis methodology. Section 6.1.2 discusses the ABC cost analysis of the re-engineering process and section 6.1.3 presents the conclusions.

6.1.1 Existing Order Processes and Internet-based Structure

The existing X Juice Company data flow is shown in Figure 6.1 about 80%-85% of the customer orders come directly from the customers and the remaining 15%-20% are generated from the Continuous Replenishment Program (CRP) that provides the information of customer inventory and demand through the Electronic Distribution Interface (EDI). Then the Customer Service Department uses Order and Inventory Management System (O&IMS) to maintain the orders and assign carriers to them. The traffic department picks up orders from the remote printer and manually assigns a pickup date, time, and carrier to each order ticket. Finally the dispatch controls the logistics and operation of inbound and outbound trucks.

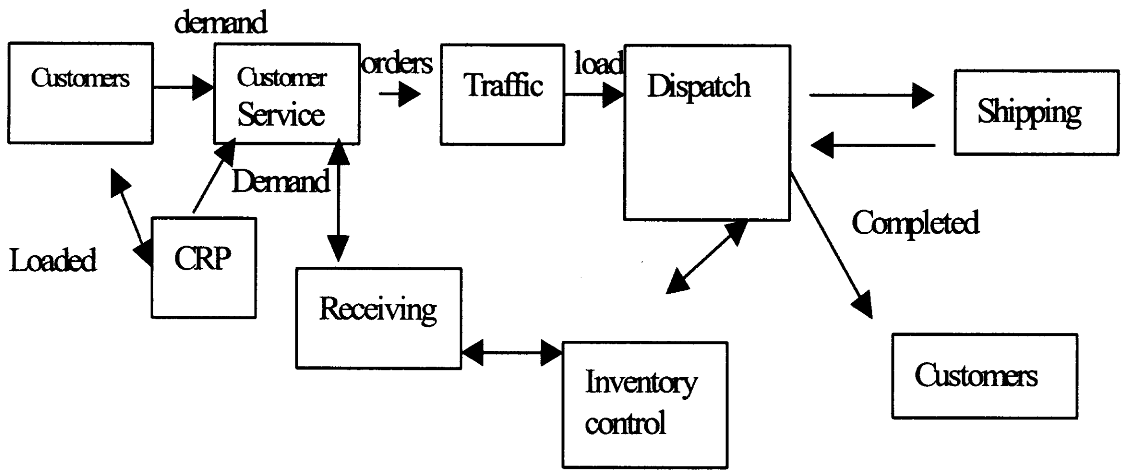

Figure 6.1 Existing X Juice Company service flowchart

To shorten the process time and provide a common work environment for all the components of the supply chain, it is suggested to create an Internet E-commerce environment as shown in Figure 6.2. The Internet can link all the existing software and databases. All parties including customers, the customer service department, the warehouse, the traffic department, manufacturing plants and top managers can view the ordering processes through a clear user interface, which saves communication time and prevents mistakes.

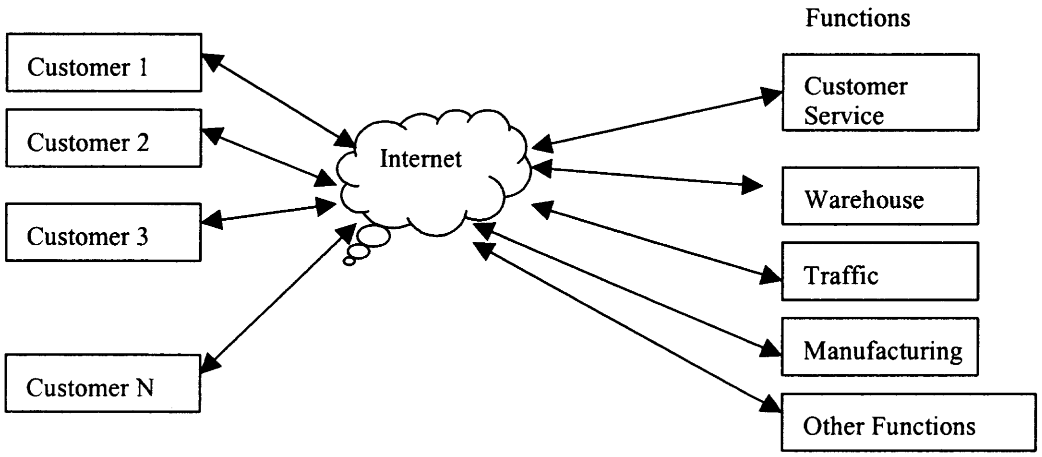

Figure 6.2 Internet B2B service flowchart

6.1.2 ABC Methodology

To use the ABC method, salaries of software professionals were collected and the summary of the survey is shown in Table 6.1.

Table 6.1 Annual Salaries of Software Staff

Personnel Resource	Number	Salary	Amount
X Juice Company employee			
Supervisor			
Senior	3	$120,000	$360,000
Junior	3	$100,000	$300,000
System Engineer			
Senior	5	$100,000	$500,000
Junior	5	$80,000	$400,000
System Analysis			
DB analysis	3	$75,000	$225,000
Tool analysis	3	$60,000	$180,000
Software developer			
15 years and higher	5	$90,000	$450,000
10 years and higher	10	$80,000	$800,000
5 years and higher	10	$65,000	$650,000
Network and System Support			
Database administrator	2	$120,000	$240,000
WEB administrator	2	$100,000	$200,000
Technical Document Writer			
20 years experience	3	$60,000	$180,000
Customer Support Experience	3	$100,000	$300,000
Subtotal	*57*		*$4,785,000*
Benefits (insurance, 401K and Stock option)		(1.4 X $4,785,000)	
X Juice Company TOTAL			**$6,699,000**
Contractors from software consultant companies			
Software Developers			
Senior	10	$120,000	$1,200,000
Junior	10	$80,000	$800,000
Testers			
Senior	10	$80,000	$800,000
Junior	10	$50,000	$500,000
Subtotal	*40*		*$3,300,000*
Total	97		**$9,999,000**

Using the data of Table 6.1, the personnel cost is:

X Juice Company $6,699,000/9,999,000 = 67.0%.

Software consultant $3,300,000/9,999,000 = 33.0%

Average personnel cost at X Juice Company $6,699,000/57 = $117,526

Average personnel cost at Software consultant $3,300,000/40 = $82,500.

According to their characteristics, the activities can be classified into six pools as shown as Table 6.2.

Table 6.2 Salary Analyses of Six Pools of Software Staffs

Personnel Resource	Number	Salary	Amount
Pool 1 : Supervisor			
Senior	3	$120,000	$360,000
Junior	3	$100,000	$300,000
Subtotal	6		*$660,000*
Benefit Factors (x 1.4)			$924,000
Average		*$154,000*	
Pool 2 : System Engineers			
System Engineer			
Senior	5	$100,000	$500,000
Junior	5	$ 80,000	$400,000
System Analysis			
DB analysis	3	$75,000	$225,000
Tool analysis	3	$60,000	$180,000
Subtotal	16		*$1,305,000*
Benefit Factors (x 1.4)			$1,827,000
Average		*$114,187*	
Pool 3 : Software Developer			
Software Developer			
15 years and higher	5	$90,000	$450,000
10 years and higher	10	$80,000	$800,000
5 years and higher	10	$65,000	$650,000
Network and System Support			
Database administrator	2	$120,000	$240,000
WEB administrator	2	$100,000	$200,000
Subtotal	29		*$2,340,000*
Benefit Factors (x1.4)			$3,276,000
Average		*$112,965*	
Pool 4 : Supporting Overhead			
Technical document writer			
20 years experience	3	$60,000	$180,000
Customer Support			
Experience	3	$100,000	$300,000
Subtotal	6		*$480,000*
Benefit Factors (x 1.4)			$672,000
Average		*$112,000*	
Pool 5 : Contractors from software consultant companies			
Software Developers			
Senior	10	$120,000	$1,200,000
Junior	10	$80, 000	$800,000
Subtotal	20		*$2,000,000*
Average		*$100,000*	
Pool 6 : Testers			
Senior	10	$80,000	$800,000
Junior	10	$50,000	$500,000
Subtotal	20		$1,300,000
Average		$65,000	

6.1.3. ABC Results Analysis

It can been seen from Table 6.2 that at a similar computer skill level, testers from the consulting company can replace support people from X Juice Company. The outsourced staff can lower the cost significantly. For example, the average support staff cost at X Juice Company is $112,000 compared with the average salary of testers from a consultant house of $65,000. If six people are replaced, the saving are 6 x ($112,000-65,000) = $282,000. The project manager can also conduct a benefit comparison for software programmers from the in-house and outsourcing developers. If they replace 29 developers between outsource, the project can save, 29 x ($112,695-100,000) = $368,155. The manager needs to decide whether the outsourcing portion should be increased to reduce the total project cost. Other then personnel costs are summarized in Table 6. 3.

Table 6.3 Annual Costs of Hardware, Software License and Office Space Analysis

Personnel Resource	Number	Cost	Amount
Depreciation equipment			
HP workstation	100	$6,500	$650,000
PC Pentium III 800MHz	100	$2,900	$290,000
Notebook Toshiba 600 MHz	50	$3,000	$150,000
MS Office	100	$950	$95,000
Subtotal			*$1,185,000*
Suppose 2 years depreciation x 0.5			$592,500
Annual expense fee			
Query Language Oracle	100	$100	$10,000
Office Space	100	$7,200	$720,000
Subtotal			*$730,000*
Total equipment Annual fee			**$1,322,500**

6.2 Y Service Provision Company Case Study

The Y Service Provision Company case is based on a large software migration to a new B2B platform. The data was collected from December 1997 to April 2000. It includes four software feature releases. It is a typical application of multiple software releases.

Collection of Errors Data

A the testing takes place detailed information about errors is selected from the Modification Request (MR) from testers or invited potential customers. The MRs include systems that do not function properly or are user-unfriendly and are kept in an MR database. An example of an MR report is shown in Figure 6.3. From this MR the user found that the inventory system (IM) does not function well enough to keep the record of the product CISCO 12012 in the TESTNJ01 location.

```
<------------------- INFORMATION FOR MR nsdi010519    ---------------->

  Product: NSDI              Release Detected: v7.0
   System: NetPlan               MR Severity: 2
Subsystem: IMS-FAC                 Required Date: 10/31/01
                          Originator: Joanne
 Abstract: FRF8457 - err msg exits user out of browse update
MR Description:
  When an error is encountered in browse update in IM, the user is exited from
browse update back to main IM screen when click OK on err msg - regardless of
err msg.  See CISCO 12012 in TESTNJ01 and try to add row w/incorrect
qualified ID.

<------------------------------------------------------------------------------>
```
Figure 6.3 Sample of errors data MR report

Data Validation

A review by the MR board members follows, and then a retest of the scenario that produced the eorr. If the board members identify this error as a user error, they will decide "killmr", i.e., put this issue into the inactive database. If it is a valid MR it will be put into the active database. For an example, MR number nsdi010519 shown in Figure 6.1 is not valid. It was decided to kill this MR as shown in Figure 6.4.

```
-------------------------- REPORT FOR MR nsdi010519 -------------------------
 MR Status:    mra_study                      Duplicate MR:
MR Severity:  2                              Product:     NSDI
Release Det.: v7.0                           System:      NetPlan
Phase Det.:   system test                    Subsystem:   IMS-FAC
Category:     testing                        Date:    10/30/01
Create Date:  10/30/01 11:03:26              Site:        MT-B
Reason Code:                                 Reqd. Date:  10/31/01
Compl. Date:                                 Originator:  joanne
Creator:      Joanne                          Closer:
Study Dev:  Joanne                            Due Date:
Abstract:  FRF8457 - err msg exits user out of browse update
Description:
When an error is encountered in browse update in IM, the user is exited from
browse update back to main IM screen when click OK on err msg - regardless of
err ms
g.  See CISCO 12012 in TESTNJ01 and try to add row w/incorrect qualified ID.
 Reason Deferred:
None.
 MR Proposed Solution:
We went thru the process with Joanne, and the Browse/Update equipment
Window does not disappear from the window.
Please kill this MR
<------------------------------------------------------------------------------------->
```

Figure 6.4 Example of an MR reviewed report

Monthly Data Report

The project manager can type the "report" command in the MR system and get the monthly report summary as shown in Table 6.4 below.

Table 6.4 Monthly Data of Four Software Releases.

YY/MM	RELEASE 1	RELEASE 2	RELEASE 3	RELEASE 4
97/12	10			
98/1	48			
98/2	35			
98/3	74			
98/4	67			
98/5	76			
98/6	99			
98/7	46			
98/8	31	9		
98/9	29	86		
98/10	40	83		
98/11	21	51		
98/12	10	41		
99/1	3	39		
99/2	0	37	12	
99/3	3	42	31	
99/4		26	84	
99/5		13	81	
99/6		10	101	
99/7		5	21	
99/8		0	20	
99/9		0	23	
99/10		1	14	
99/11			2	
99/12				5
00/1				18
00/2				108
00/3				83
00/4				43
00/5				55
00/6				67
00/7				23
00/8				24
00/09				1
00/10				1

Based on the time frame, the manager can use Microsoft Excel to plot the error numbers as in Figures 6.5 and 6.6

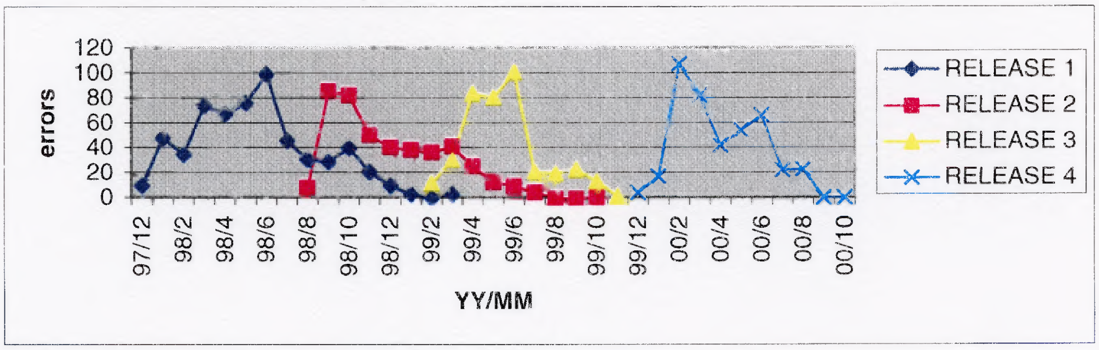

Figure 6.5 Monthly error chart of four releases

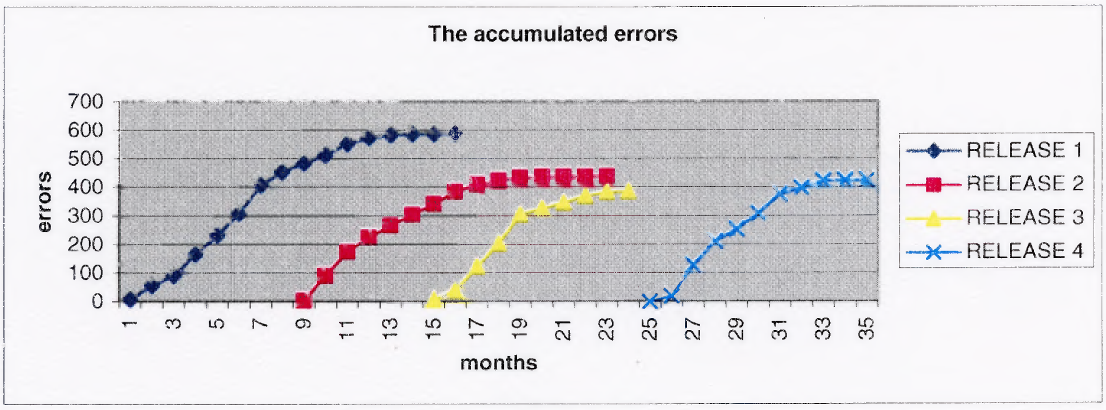

Figure 6.6 Accumulated errors of four releases

The Data Fitting – NHPP Goel Model

There are also two factors that can help in making the data fitting decision: The Non-Homogeneous Poisson Process and the Data Characteristic Decision. The seed data of the estimated errors "a" and estimated error detection rate "b" can use historical data from a previous project. Following NHPP data for the four releases, Tables 6.5, 6.6, 6.7 and 6.8 can be generated.

Table 6.5 Goel's Model Analysis of the Release One Error Data

tk	yk-yk-1	yk	tk*exp(-b*tk)	tk-1*exp(-b*tk-1)	exp(-b*tk-1)	exp(-b*tk)	(b*(d-e)/9f-g)
0	0	0					
1	9	9	0.900076967	0	1	0.900077	81.069324
2	95	104	1.620277094	0.900076967	0.90007697	0.810139	760.7317533
3	178	282	2.18756114	1.620277094	0.81013855	0.729187	1247.371075
4	229	511	2.625297863	2.18756114	0.72918705	0.656324	1375.763911
5	270	781	2.953712673	2.625297863	0.65632447	0.590743	1352.07972
6	309	1090	3.190282495	2.953712673	0.59074253	0.531714	1238.380124
7	346	1436	3.350083092	3.190282495	0.53171375	0.478583	1040.665123
8	388	1824	3.446094434	3.350083092	0.4785833	0.430762	778.9886347
9	414	2238	3.489469006	3.446094434	0.4307618	0.387719	417.1889041
10	427	2665	3.489767423	3.489469006	0.38771878	0.348977	3.289038739
11	437	3102	3.455165207	3.489767423	0.34897674	0.314106	-433.6339346
12	442	3544	3.392634132	3.455165207	0.31410593	0.28272	-880.5954213
13	442	3986	3.308101161	3.392634132	0.28271951	0.254469	-1322.595421
14	442	4428	3.206587635	3.308101161	0.25446932	0.229042	-1764.595421
15	443	4871	3.092331079	3.206587635	0.22904197	0.206155	-2211.587719
16							
17							
18							
19							
20							1682.51969
0.105275		1682.009097					
558.0437317							
Test time / months	R(0.1/T)	R(1/T)					
16	0.299761365	3.20937E-05					
17	0.338109849	9.02468E-05					
18	0.376804249	0.000228864					
19	0.415404874	0.000528858					
20	0.453518558	0.001123966					

From Chapter 5 equation (5.3), (5.4), (5.5) and (5.6)

$$\hat{a} = 661.218$$

$$\hat{b} = 0.141$$

$$\hat{m} = 661.218\,X(1 - e^{-0.141})$$

$$R = (x/t) = e^{-(661.218)[e^{-(0.141)t} - e^{-(0.141)(T + x)}]}$$

Table 6.6 Goel's Model Analysis of the Release Two Error Data

tk	yk-yk-1	yk	tk*exp(-b*tk)	tk-1*exp(-b*tk-1)	exp(-b*tk-1)	exp(-b*tk)	(b*(d-e)/9f-g)
0	0	0					
1	9	9	0.900076967	0	1	0.900077	81.069324
2	95	104	1.620277094	0.900076967	0.90007697	0.810139	760.7317533
3	178	282	2.18756114	1.620277094	0.81013855	0.729187	1247.371075
4	229	511	2.625297863	2.18756114	0.72918705	0.656324	1375.763911
5	270	781	2.953712673	2.625297863	0.65632447	0.590743	1352.07972
6	309	1090	3.190282495	2.953712673	0.59074253	0.531714	1238.380124
7	346	1436	3.350083092	3.190282495	0.53171375	0.478583	1040.665123
8	388	1824	3.446094434	3.350083092	0.4785833	0.430762	778.9886347
9	414	2238	3.489469006	3.446094434	0.4307618	0.387719	417.1889041
10	427	2665	3.489767423	3.489469006	0.38771878	0.348977	3.289038739
11	437	3102	3.455165207	3.489767423	0.34897674	0.314106	-433.6339346
12	442	3544	3.392634132	3.455165207	0.31410593	0.28272	-880.5954213
13	442	3986	3.308101161	3.392634132	0.28271951	0.254469	-1322.595421
14	442	4428	3.206587635	3.308101161	0.25446932	0.229042	-1764.595421
15	443	4871	3.092331079	3.206587635	0.22904197	0.206155	-2211.587719
16							
17							
18							
19							
20							1682.51969
0.105275		1682.009097					
558.0437317							
Test time / months	R(0.1/T)	R(1/T)					
16	0.299761365	3.20937E-05					
17	0.338109849	9.02468E-05					
18	0.376804249	0.000228864					
19	0.415404874	0.000528858					
20	0.453518558	0.001123966					

From Chapter 5 equation (5.3), (5.4), (5.5) and (5.6)

$$\hat{a} = 558.043$$

$$\hat{b} = 0.105$$

$$\hat{m} = 558.043X(1 - e^{-0.105})$$

$$R = (x/t) = e^{-(558.043)[e^{-(0.105)t} - e^{-(0.105)(T+x)}]}$$

Table 6.7 Goel's Model Analysis of the Release Three Error Data

tk	yk-yk-1	yk	tk*exp(-b*tk)	tk-1*exp(-b*tk-1)	exp(-b*tk-1)	exp(-b*tk)	(b*(d-e)/9f-g)
0	0	0					
1	12	12	0.863000507	0	1	0.863001	75.59156526
2	43	55	1.489539751	0.863000507	0.86300051	0.74477	227.8697755
3	127	182	1.928210342	1.489539751	0.74476988	0.642737	546.0107323
4	208	390	2.218728671	1.928210342	0.64273678	0.554682	686.2537978
5	307	697	2.393454961	2.218728671	0.55468217	0.478691	705.8842112
6	330	1027	2.478663415	2.393454961	0.47869099	0.413111	428.7680446
7	350	1377	2.495602415	2.478663415	0.41311057	0.356515	104.7539867
8	373	1750	2.461378458	2.495602415	0.35651463	0.307672	-261.3621799
9	387	2137	2.389692215	2.461378458	0.30767231	0.265521	-658.1720205
10	389	2526	2.29145066	2.389692215	0.26552136	0.229145	-1050.573426
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							805.0244865
0.14734		805.1005449					
497.5793154							
Test time / months	R(0.1/T)	R(1/T)					
16	0.299761365	9.84696E-05					
17	0.338109849	0.003814688					
18	0.376804249	0.00818084					
19	0.415404874	0.015803145					
20	0.453518558	0.027894253					

From Chapter 5 equation (5.3), (5.4), (5.5) and (5.6)

$$\hat{a} = 497.579$$

$$\hat{b} = 0.147$$

$$\hat{m} = 497.579\,X(1 - e^{-0.147})$$

$$R = (x/t) = e^{-(497.579)[e^{-(0.147)t} - e^{-(0.147)(T+x)}]}$$

Table 6.8 Goel's Model Analysis of the Release Four Error Data

tk	yk-yk-1	yk	tk*exp(-b*tk)	tk-1*exp(-b*tk-1)	exp(-b*tk-1)	exp(-b*tk)	(b*(d-e)/9f-g)
0	0	0					
1	5	5	0.875705878	0	1	0.875706	35.2271638
2	23	28	1.53372157	0.875705878	0.87570588	0.766861	139.0449535
3	131	159	2.014633491	1.53372157	0.76686078	0.671544	660.9516915
4	214	373	2.352301854	2.014633491	0.6715445	0.588075	865.7226105
5	257	630	2.574905701	2.352301854	0.58807546	0.514981	782.6762191
6	312	942	2.705832069	2.574905701	0.51498114	0.450972	638.1750209
7	379	1321	2.76443189	2.705832069	0.45097201	0.394919	396.2190158
8	402	1723	2.766662006	2.76443189	0.39491884	0.345833	18.26396925
9	426	2149	2.725629954	2.766662006	0.34583275	0.302848	-406.6456445
10	427	2576	2.652055747	2.725629954	0.30284777	0.265206	-834.6002118
11	428	3004	2.554662888	2.652055747	0.26520557	0.232242	-1264.554779
12							
13							
14							
15							
16							
17							
18							
19							
20							1030.480009
0.132725		1030.834793					
513.0734528							
Test time / months	R(0.1/T)	R(1/T)					
16	0.396965795	0.000487046					
17	0.445272342	0.001256857					
18	0.49237895	0.002882886					
19	0.537705952	0.005964252					
20	0.580813278	0.011273047					

From Chapter 5 equation (5.3), (5.4), (5.5) and (5.6)

$\hat{a} = 513.073$

$\hat{b} = 0.133$

$\hat{m} = 513.073\,X(1 - e^{-0.147})$

$$R = (x/t) = e^{-(513.073)[e^{-(0.133)t} - e^{-(0.133)(T+x)}]}$$

The Data Fitting – NHPP S-Shaped Model

The S-Shaped NHPP reanalysis of the data for the four releases are shown in Tables 6.9 and 6.10. The simulation programs are listed in Appendix A.

Table 6.9 Estimated Total Errors (a) using an S-Shaped Model of the Four Releases

Release 1	Release 2	Release 3	Release 4
1000	1000	1000	1000
97	159	940	38
295	1000	158	163
1000	483	1000	235
1000	397	1000	1000
1000	405	666	754
1000	430	485	840
1179	477	495	606
843	485	441	558
747	474	418	491
760	469		463
685	463		
629	454		
620	450		
	448		

Table 6.10 Estimated Error Detection Rate (b) using an S-Shaped Model of the Four Releases

Release 1	Release 2	Release 3	Release 4
0	0	0	0
1	1	0.17	1
0.38	0.08	1	1
0.1	0.4	0.01	1
0.14	0.47	0.05	0.18
0.14	0.43	0.28	0.24
0.1	0.38	0.36	0.22
0.17	0.38	0.38	0.28
0.22	0.39	0.4	0.31
0.24	0.4	0.43	0.35
0.23	0.4		0.38
0.25	0.42		
0.28	0.43		
0.29	0.43		
0.3			

It can be seen for Table 6.10 that for b in the S-Shaped model the detection rate b is mostly in the 0. 3 to 0.43 range.

Data of Steady State Analysis

The simulation programs listed in Appendix A were used to get Tables 6.11 (p.66) and 6.12 (p.67). As the weeks continued the estimate became a stable value. Figure 6.7 shows that after 8-10 months, the estimation of errors became stable. Similarly in Figure 6.8, the detection rate b becomes stable as the 8-10 months error data was collected and fitted into the S-Shaped model.

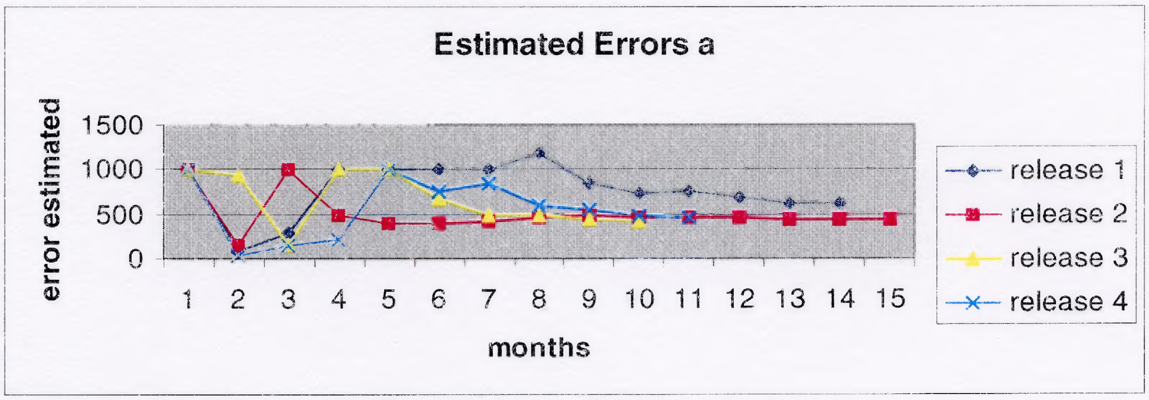

Figure 6.7 Estimated a in time domain

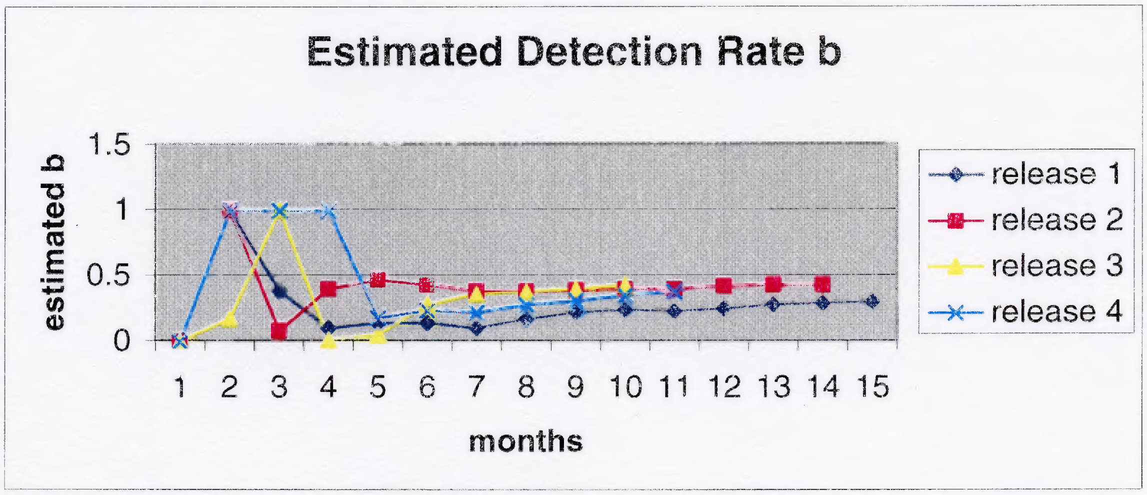

Figure 6.8 Estimated b in time domain

The Reliability Model

From the chapter 5 equation (5.7)

$$\hat{m}(t) - Z\alpha\sqrt{\hat{m}} \leq N(t) \leq \hat{m}(t) + Z\alpha\sqrt{\hat{m}}$$

Confidence intervals for N (t) based on the Poisson distribution.

From the data, the project manager can derive the software reliability performance measures table.

Table 6.11 Reliability of Release One as Time Extended

Test time / months	R(0.1/T)	R(1/T)
16	0.379077202	0.000110414
17	0.43065464	0.000365939
18	0.48111075	0.001035993
19	0.529703974	0.002557816
20	0.575871799	0.005607408

R (x/t) means the possibility next x time will have an error happen at time t.

The Software Cost Analysis

The Staff at Y Service Provision Company receive salaries similar to those of the staff at X Juice Company, for comparable computer skill levels. The weekly salary of a tester from Table 6.10 is around \$1,200 to \$1,500. The project needs three full time testers; the personnel cost is 88% of the total cost, so the total cost should be 100/88; thus the cost is in the \$3,000 to \$5,000/week range and there are the figures to be used in the next chapter. The debugging costs are the same. In Table 6.12 the developer's salary is shown at \$50/hour, and one bug may need 10 to 20 developer-hours to solve, so the cost is around \$500 to \$1,000 for one bug.

The Simulation Conditions

Based on the assumptions, the errors are a random number between 300-900. The experiment uses 600+ 600, random (-0.5,0.5) as the input errors. The software lifecycle is one year. Each year there are two releases, and each release earns a profit of $1,000,000. The first delivery is made after one year, and then the delivery interval Ti is 12 weeks or 16 weeks. The installation fee for each delivery is $40,000 and the penalty cost for each error is $2,000 per month. An experimental simulation involved about six factors, k=six factors in two levels. The total factorial run is n=64. Once the significant factors are identified, the impact of the factors and the confidence interval of the decision based on the factor level are studied. The testing cost, debugging and penalty cost come from the survey of the local software industry and project size, as described above.

Table 6.12 High and Low Level Factors

Factors Levels		
	High +	Low -
A: The error detection rate	0.5	0.3
B: The software revenue decay/week	0.02	0.01
C: The delivery interval in weeks	16 weeks	12 weeks
D: The software test cost/week	$5,000	$3,000
E: The cost of removing an error during test	$1,000	$500
F: The error penalty cost/week	$4,000	$2,000

In order to process the independent variability on the six factors surveyed, the simulation program generates the random errors number into the system and gets the total profit based on the different factor combinations. That is, $2^6 = 64$ sets of the total cost result. The factorial result of profits of the software projects if delivery is on release 1 is displayed in Appendix B Table B.1. The factorial profits of the software project if delayed delivery is postponed are shown in Table B.2.

CHAPTER 7

ANALYSIS OF RESULTS

Based on data in Chapter 6, this chapter analyzes software development costs for the X Juice Company. It also analyzes the software multiple delivery policy based on the Multiple-Release model.

7.1 Results of ABC Analysis

In the X Juice Company case, the salaries of staff can be grouped into 6 salary pools, which are summarized in Table 7.1.

Table 7.1 Summary of Annual Salaries of Six Pools

Pool 1	Supervisor	Average	$154,000
Pool 2	System Engineer	Average	$114,187
Pool 3	Software Developer	Average	$112,965
Pool 4	Supporting Overhead	Average	$112,000
Contractors from software consultant companies			
Pool 5	Software Developers	Average	$100,000
Pool 6	Testers	Average	$65,000

The manager needs to decide whether the outsourcing portion should be increased in order to reduce the total project cost. The total development cost includes the expensive for staff, software and hardware in Table 7.2.

Table 7.2 Total Cost Analyses

Personnel Resource	Number	Pool Avg	Amount
Supervisor	6	$154,000	$924,000
System Engineers	16	$114,187	$1,827,000
Software Developer	29	$112,965	$3,276,000
Supporting Overhead	6	$112,000	$672,000
Subtotal			*$6,699,000*
Contractors from software consultant companies			
Software Developers	20	$100,000	$2,000,000
Testers	20	$ 65,000	$1,300,000
Subtotal			$3,300,000
Total Personnel			$9,999,000
Equipment and Space			$1,322,500
Total Estimated Annual Cost for X Juice Company Project			**$11,321,500**

According to Table 7-2, the in-house development cost (with a staff of 57) is:

$6,699,000 + (57/97) x $1,322,500 = $7,476,139

The outsourcing cost (with a staff of 40 staff)is:

$3,300,000 + (40/97) x $1,322,500 = $3,845,361

Consequently, we can derive the following conclusions: The current in-house expense is 66% and for outsourcing is 34% of the total project. The average support staff cost on the X Company is $112,000, while for a tester from a consultant house with the same computer skill level costs $65,000. It's easy to outsource this portion and reduce the cost significantly. Replacement of the 29 software developers is a major portion of cutting the cost. Note that the personnel expense of $9,999,000 is 88%.

7.2 Analysis of Multiple-Release Model

DOE methodology is used to analyze the results of the six controllable factors which impact the total cost of the software, listed as follow:

A The error detection rate/week

B The amount of software revenue decay rate/week

C The delivery interval in weeks

D The software test cost/week

E The cost of removing error during test

F The error penalty cost/week

There are also several elements that can improve profit after the profit results analysis: accurate cost information, speeded-up error detection rate, or modification of delivery interval time. One can also get alternative software developers, e.g., subcontract to consultant firms, or outsource to overseas countries like India or China, to lower the software development cost factors. The detailed analyses are shown below. The corresponding contrasts of the total profit mean $\bar{Y}$, $(S/N)_i = -10.\log_{10}\frac{1}{n}(\sum_i 1/y_i^2)$ and $\log_{10}(S)$ are presented in Table 7-3.

Table 7.3 Contrast Effect for Total Profit

	Ave On-Time Delivery $\overline{Y}$	**Ave Delayed Delivery** $\overline{Y}$	**On-Time** $\log_{10}(S)$	**Delayed** $\log_{10}(S)$	**On – Time S/N**	**Delayed S/N**
A	**188,921**	**82,155**	-0.82	**-0.72**	**12.33**	**7.08**
B	-14,000	-28,000	0.00	0.00	-0.68	-4.02
C	**50,449**	-39,515	-0.25	-0.02	2.85	-5.13
D	-8,122	-36,000	0.00	0.00	0.32	-4.79
E	-19,634	-38,293	0.09	0.29	1.90	**1.37**
F	-38,008	-2,245	0.18	0.01	-3.34	-0.19
AB	**174,921**	**54,155**	-0.82	**-0.72**	**11.65**	**3.06**
AC	**234,274**	**42,156**	**-83.34**	**-0.74**	**14.68**	**1.68**
AD	**180,921**	**46,155**	-0.82	**-0.72**	**12.85**	**2.28**
AE	**165,781**	**43,863**	-0.70	-0.43	**13.93**	**7.36**
AF	**126,063**	**63,903**	-0.64	**-0.70**	**8.99**	**6.89**
BC	31,353	-67,999	-0.25	-0.02	1.67	-9.41
BD	-14,375	-57,274	0.28	0.25	5.97	-2.52
BE	-37,140	-66,293	0.12	0.29	0.92	-2.64
BF	-76,858	-30,245	0.18	0.01	-4.02	-4.20
CD	37,353	-75,999	-0.25	-0.02	2.67	-10.19
CE	22,213	-78,292	-0.13	0.27	3.95	-4.03
CF	-17,505	-39,646	-0.06	-0.01	-0.99	-5.58
DE	-31,140	-74,060	0.12	0.29	1.92	-3.42
DF	-70,858	-35,509	0.18	0.01	-3.02	-4.98
EF	-85,998	-40,538	0.30	0.30	-1.73	**1.19**
mean	38,029	-17,977	-4.12	-0.11	3.94	-1.57
SD	98,895	51,479	18.15	0..39	6.06	5.01
CL(95%)	45,016	23,433	-8.26	-0.18	6.7	0.71

7.3 Analysis of the Means of Total Profit (y)

A probability plot analysis of On-Time delivery factors is shown in Figure 7.1. Clearly, the AC interaction factor and the A factor itself are significant.

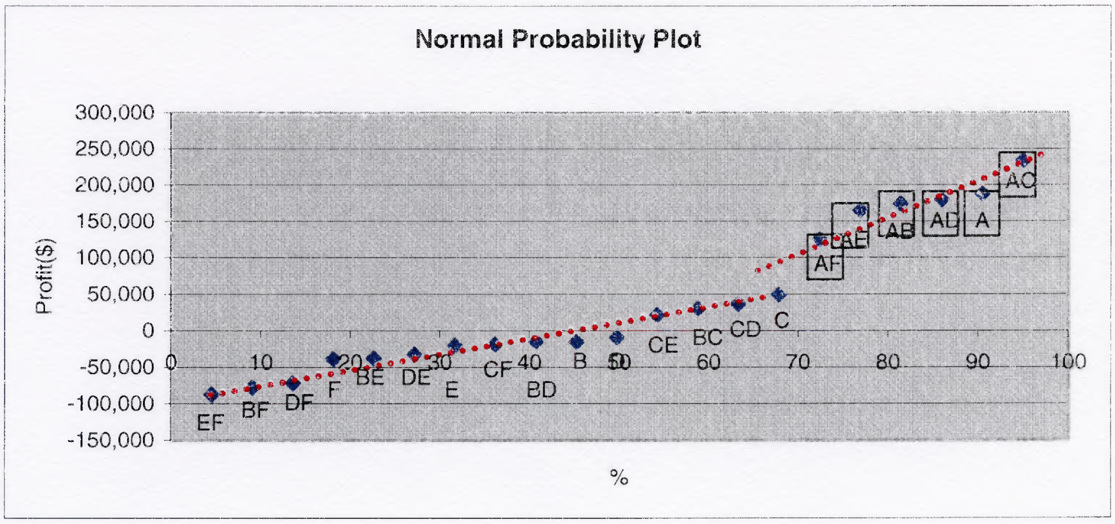

Figure 7.1 Probability plot for On-Time delivery

In Figure 7.1, AC, A, AD, AB, AE, AF and C are significant factors to the total profit. Compared with the A+ the 0.5 error detection rate and the 0.3 error detection rate, there is an $188,921 difference to the total profit, which indicates that the quality and speed of the testers is a dominant factor of the total profit. The detection rate effect to total profit is shown in Table 7.4.

Table 7.4 On-Time Delivery Detection Rate Effect on Total Profit

	0.5 detection rate	0.3 detection rate	Difference
A	128464	-60457	188921

With the A factor the error detection rate at 0.5, and the delivery interval C- (12) weeks give a better result of total profit: $129.004. The difference between these two significant conditions gives a $234,274 difference. The two-way effect is shown on Table 7.5.

Table 7.5 Detection Rate and Delivery Intervals Effect on Total Profit

	C+	C-
A+	127923.68	129004.36
A-	-14563.75	-56456.84

From the table above A+ C+ is $127,923.68 and A- C- is -$56,456.84. There is a $184,380.52 difference. It is a significant two-way factor. Compare A+ C+ and A+ C- and we can make another conclusion: the A factor is a stronger influence over the total profit than the C factor. A probability plot analysis of delayed delivery factors is shown in Figure 7.2.

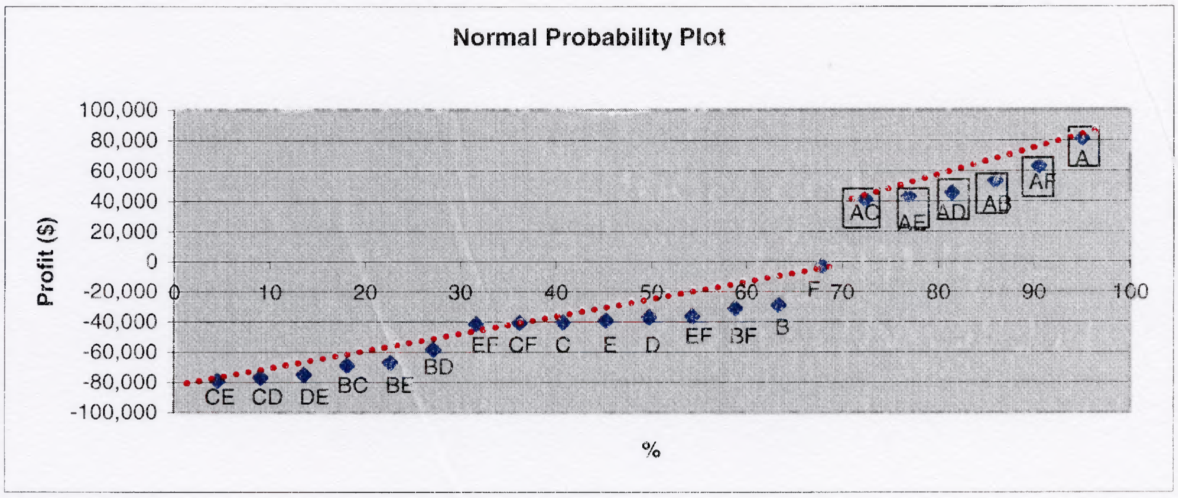

Figure 7.2 Probability plot for Delayed delivery

A, AF, AB, AD, AE and AC are significant factors, compared with A+ being the 0.5 error detection rate and 0.3 error detection rate. There is a $82,155 difference. It shows that the quality and speed of the testers is a dominant factor in the total profit. The detection rate effect on total profit is shown on Table 7.6.

Table 7.6 Delayed Delivery-Detect Rate Effect Total Profit

	0.5 detection rate	0.3 detection rate	Different
A	66271	-15885	82155

The factors B, C, D, E, F are all negative values, and AB, AC, AD, AE, AF all are positives. This means improvement of the error detection rate factor A can make a difference if the software have to be delayed to the second release schedule. Also a 12-week delivery interval time is better than a 16-week delivery interval.

As a conclusion, the following facts were observed.

1. On-Time condition: the error detection rate A+ (0.5) is the dominant factor of the total profit.
2. On-Time condition: the delivery interval under A+ (0.5) error detection rate. For C+ (16) weeks, the profit is $127,923, and for C- (12) weeks the profit is $129,004.
3. Delayed condition: A+ (0.5) error detection rate is the dominant factor.

7.4 Analysis of Variation $\log_{10}(S)$

The standard deviation of each set of three observations provides an estimation of variability of total profit under the particular set of factor levels used in that run. The effects of the factors A, B, C, D, E, F and the two-way interaction on $\log_{10}(S)$ are studied using contrast effects from Table 7-1. The probability plots are shown in Figure 7.3 for On-Time delivery and Figure 7.4 for Delayed delivery.

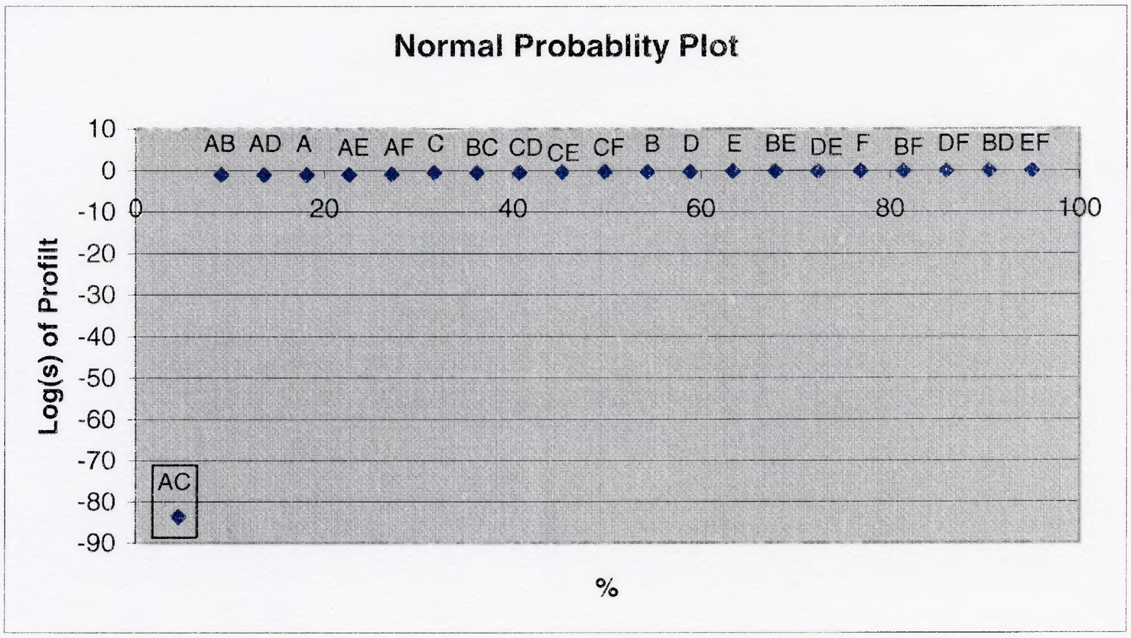

Figure 7.3 Probability plot for On-Time delivery for the $\log_{10}(S)$

This analysis indicates that only AC is a significant factor. A "two-way" means analysis for $\log_{10}(S)$ is presented below for the AC indentation. The two-way effect is shown in Table 7.7.

Table 7.7 Detection Rate and Delivery Interval Effect on Total Profit

$\log_{10}(S)$	C+	C-
A+	4.01	4.29
A-	4.86	87.36

The AC two-way factor A+C+ - (A-C-) =4.01 –87.36 = -83.34 has a significant impact on the variability. A similar result can indicate, as shown previously in Chapter 7.1, that AC is a significant two-way factor.

Clearly, the result is the same as the average analysis AC is significant for the total profit. For the standard deviation of each set of three observations of delayed delivery, the probability plot analysis is shown below.

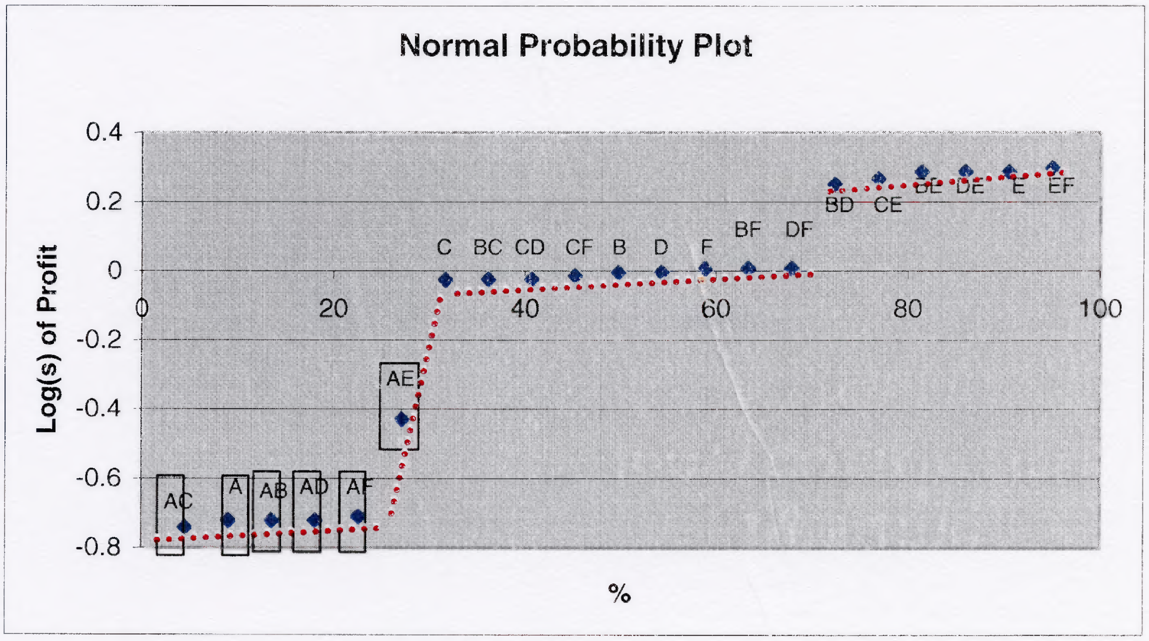

Figure 7.4 Probability plot for Delayed delivery for the $\log_{10}(S)$

The range of the $\log_{10}(S)$ is between –0.8 and 0.4. A, AF, AB, AD, AE and AC are significant factors, the results are similar as the contract effects of the average profit $\bar{Y}$ of the On-Time delivery.

As a conclusion, the following facts were observed.

1. On-Time condition: the error detection rate A+ (0.5) is the dominant factor of the variation of profit.
2. On-Time condition: the delivery interval under A+ (0.5) error detection rate, for the C+ (16) weeks the variation of profit is 4.01, and C- (12) weeks the variation of profit is 4.29. So the values are very close.
3. Delayed condition: no significant dominating factor.

7.5 Analysis of Signal to Noise Ratio $-10.\log_{10}\frac{1}{n}(\sum_i 1/y_i^2)$

Since in this study the higher the S/N value, the better the total profit, it would perform the analysis using the $(S/N)_i = -10.\log_{10}\frac{1}{n}(\sum_i 1/y_i^2)$, where y_i is the observed value at experiment i. signal-to-noise ratio. The values are generated from EOD and the summaries are illustrated in Table 7.3. The On-Time delivery analysis is shown in Figure 7.5 and the Delayed delivery is displayed in Figure 7.6, respectively.

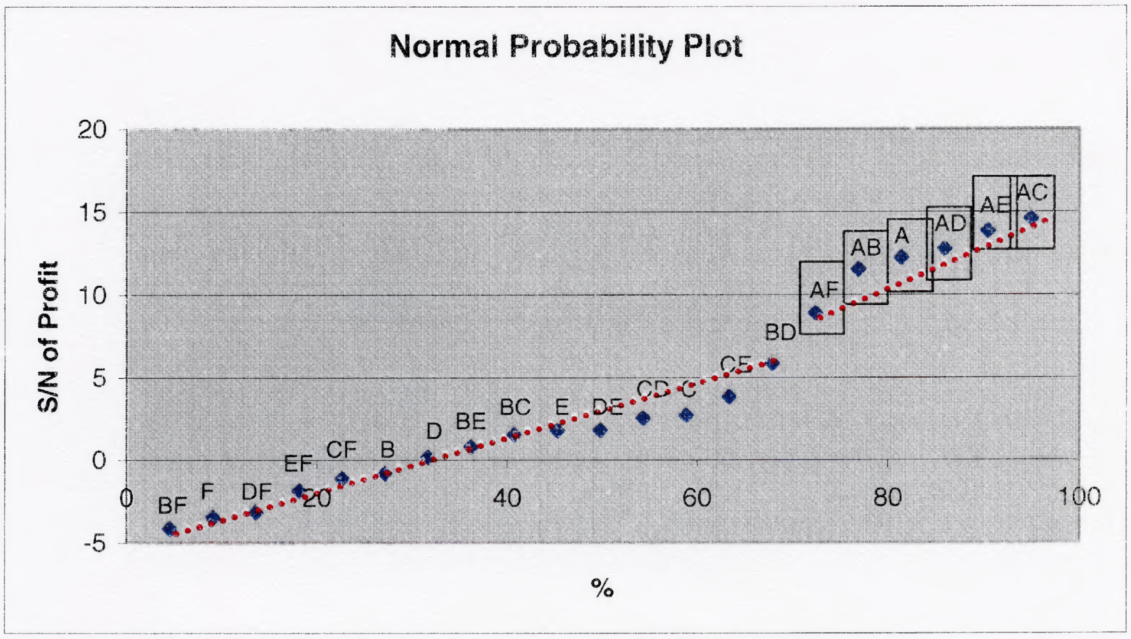

Figure 7.5 Probability plot for On-Time delivery, $S/N = -10.\log_{10}\frac{1}{n}(\sum_i 1/y_i^2)$

The probability plot analysis in Figure 7.5 clearly suggests that A and two-way interaction AB, AC, AD, AE and AF contrast are significant positives. Factor A and two-way interaction effects are shown Table 7.8 below.

Table 7.8 S/N of Significant Factors of On-Time Delivery.

On-Time	A +	A -	Diff S/N
A	101.99	89.66	12.33
B+	101.50	102.00	-0.5
B-	102.48	89.85	12.63
C+	102.04	91.96	10.08
C-	101.94	87.36	14.58
D+	101.71	90.26	11.45
D-	107.27	88.86	18.41
E+	101.62	91.64	9.98
E-	102.36	87.68	14.68
F+	101.47	86.84	14.63
F-	102.51	92.48	10.03

Factor A is the dominant factor, and factors B, C, D, E, F are all smaller values. After combining with factor A, the two-way interactions AB, AC, AD, AE, and AF all are increased and become significant. This means the improvement of the error detection rate A can make a difference. The result is similar to the average error analysis of mean total profit (y).

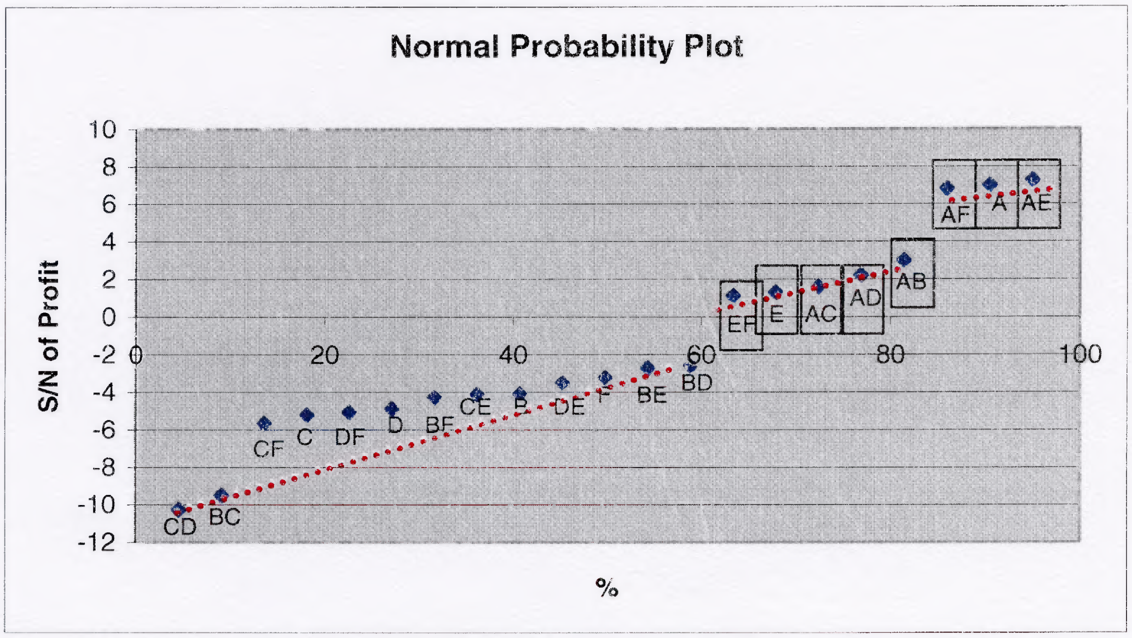

Figure 7.6 Probability plot for Delayed delivery, $S/N = -10.\log_{10}\frac{1}{n}(\sum_i 1/y_i^2)$

The probability plot analysis in Figure 7.6 clearly suggests that A, E and the two-way interactions AB, AC, AD, AE, AF and EF contrasts are significantly affects. The factors and two-way interaction effects are shown Table 7.9 below.

Table 7.9 S/N of Significant Factors of Delayed Delivery.

Delayed	**A +**	**A -**	**Diff S/N**
A	94.56	87.48	7.08
B+	91.91	97.62	-5.71
B-	97.21	88.85	8.36
C+	90.55	86.10	4.45
C-	98.57	88.87	9.7
D+	91.14	86.11	5.03
D-	97.98	88.86	9.12
E+	92.40	89.92	2.48
E-	95.62	85.05	10.57
F+	94.56	87.30	7.26
F-	94.50	87.66	6.84

The result is similar to the On-Time delivery analysis: factor A is the dominant factor, and the detection rate is significantly affects the total profit. Previously it was shown from the S/N probability table that as an overall conclusion as below:

1. A factor and the error detection rate should be set at its high level at 0.5 in order to reduce variability and get the maximum total profit.
2. The delivery interval of each release, factor C, has not been shown to influence either the total profit or the variation of the profit. The level of C may thus be set to12 or 16 weeks on the basis of reduced operating costs.
3. Factor errors penalty cost F has not been shown to have a significant effect on variability in the total cost, so in this respect it is arbitrary which level is chosen. However, since it has a large influence on total average cost, it is important to choose the level carefully so as to achieve the desired optimal level.

7.6 Decisions of Release and t-Test

In order to make a decision based on the previous analysis, the research is using the pair wise comparison t-Test to get the confidence interval of our decision. If the project manager chose to deliver the software on time, there are 64 factorial situations on release1 and 64 situations on delayed delivery.

7.6.1 Comparison of the On-Time Delivery Profit μ_1 with the Delayed Delivery Profit μ_2

Two means were tested. The hypothesis is to test if the profit of delivery on time is better than the delayed delivery.

$$H_0 : \mu_D = 0$$
$$H_{1:} : \mu_D \neq 0$$

The d_1, d_2, $d_3 \ldots d_n$ are the differences between two samples. This situation is often referred to as the

$$t = \frac{\bar{d}}{Sd/\sqrt{n}}$$ With a degree of freedom n-1

Type I error is designed as α =0.05. The rule to reject if $|t| \geq t_{1-\alpha/2}$ is applied.

The lump sum 64-case hypothesis is shown in Table 7.10.

Table 7.10 t-Test of On-Time Delivery and Delayed Delivery

t-Test: Pair Wise Comparison	
	Variable d
Mean	8811
Standard Deviation	110791
Observations	64
df	63
t Stat	0.6362
t Critical two-tail	1.9983
P	>0.05

From Table 7.10, the pair wise comparison t-Test analysis, there is not greater than 95% confidence that the On-Time delivery time frame may be better for the delivery to make more money.

7.6.2 F+ Condition: Comparison of the Delayed Delivery Profit μ_1 with on-time Delivery Profit μ_2

In Table 7.11 below, it is clear that under F+ conditions, in most cases Delayed delivery earns a better total profit than On-Time delivery. The F+ situation means a high error penalty situation, and if the project manager chooses to deliver software at Delayed delivery times, the t-Test result is as follows:

Table 7.11 Different Factors between On-Time Delivery and Delayed Delivery

	Delayed delivery-On-Time delivery
A	-106,765
B	-14,000
C	-90,200
D	-28,000
E	-17,515
F	60,613
AB	-120,765
AC	-83,683
AD	-134,765
AE	-121,919
AF	-46,153
BC	-99,352
BD	-42,899
BE	-29,153
BF	46,613
CD	-113,352
CE	-100,505
CF	-24,739
DE	-43,153
DF	32,613
EF	45,460

Delayed delivery is chosen with F+, from the data of the t-Test, the results will be shown in Table 7.12.

Table 7.12 t- Test of On-Time delivery and Delayed delivery under F+ condition

t-Test: Pair Wise Comparison	
	Variable d
Mean	-21496
Standard Deviation	103968
Observations	32
df	31
t Stat	-1.1512
t Critical two-tail	2.0395
P	<0.05

Under the F+, and the penalty cost is \$4,000/error, the mean of the Delayed delivery is better than On-Time delivery. The higher the error penalty the better to postpone the delivery time to Delayed delivery. However, the absolute value of -1.1512 is smaller than the critical value 2.0395. It is still not enough to say the Delayed delivery is better than On-Time delivery under a 95% confidence interval of the pair wise comparison t-Test.

7.6.3 Comparison of the 12-week Interval Time Profit μ_1 and 16-week Interval Time Profit μ_2

Using the pair wise comparison t-Test, here is a comparison of the result of a 12-week interval time and a 16-week interval time as shown below. Table 7.13 contains the On-Time delivery results and Table 7.14 contains the Delayed delivery results.

Table 7.13 t-Test of On-Time Delivery Comparing 12-week and 16-week Interval

t-Test: Pair Wise Comparison	
	Variable d
Mean	-45343
Standard Deviation	60796
Observations	32
df	31
t Stat	-4.1535
t Critical two-tail	2.0395
P	>0.05

The absolute value of –4.1535 is larger than the critical value 2.0395. Under On-Time delivery, the delivery intervals of 16-week is better than 12-week under 95% confident interval of pair wise comparison t-Test.

Table 7.14 t-Test of Delayed Delivery Comparing 12-week and 16-week Interval

t-Test: Pair Wise Comparison	
	Variable d
Mean	39999
Standard Deviation	10243
Observations	32
df	31
t Stat	21.7424
t Critical two-tail	2.0395
P	>0.05

From the above pair wise comparison t-Test, under a delayed condition, the 12-week interval is clearly better than the 16-week interval, and the confidence interval is more than 95%. In addition, if under the delayed condition the 12-week interval is much better than the 16-week interval, the profit is \$45,192. Comparing that profit with \$5,193, there is about a \$39,999 improvement. Using the release 1 data, from Appendix B, comparing the average profits are good in decreasing order: On-Time 16-week > Delayed 12-week > On-Time 12-week > Delayed 16-week.

7.7 Delivery Case Analysis

Based on the report of Table 7.3, in the dissertation four different delivery condition are under study.

7.7.1 On-Time Delivery Case Analysis

If On-Time delivery profit > 0 and On-Time delivery profit > Delayed delivery profit condition is happened. From Table 7.15, we can see that, when the A+

condition means a high error detection rate, the On-Time delivery gives a larger profit.

Table 7.15 Cases in which On-Time Delivery Profit is Positive

						On-Time delivery	Delayed delivery
A	B	C	D	E	F	Average	Average
+	-	-	-	-	-	152,003	120,809
+	-	+	-	-	-	149,949	88,885
+	-	-	+	-	-	148,003	92,809
+	-	+	-	-	+	146,032	88,883
+	-	-	-	+	-	144,518	107,722
+	-	-	+	+	-	140,518	79,722
+	+	-	-	-	-	140,003	96,809
+	-	+	+	-	-	137,949	44,885
+	-	+	-	+	-	137,815	75,773
+	+	-	+	-	-	136,003	68,809
+	-	+	+	-	+	134,032	44,883
+	+	+	-	-	-	133,949	56,885
+	-	+	-	+	+	133,898	75,770
+	+	-	-	+	-	132,518	83,722
+	+	+	-	-	+	130,032	56,883
+	-	-	-	-	+	129,491	120,706
+	+	-	+	+	-	128,518	55,722
+	-	+	+	+	-	125,815	31,773
+	-	-	+	-	+	125,491	92,706
+	-	-	-	+	+	122,006	107,619
+	+	+	+	-	-	121,949	12,885
+	-	+	+	+	+	121,898	31,770
+	+	+	-	+	-	121,815	43,773
+	+	+	+	-	+	118,032	12,883
+	-	-	+	+	+	118,006	79,619
+	+	+	-	+	+	117,898	43,770
+	+	-	-	-	+	117,491	96,706
+	+	-	+	-	+	113,491	68,706
+	+	-	-	+	+	110,006	83,619
+	+	+	+	+	-	109,815	-227
+	+	-	+	+	+	106,006	55,619
+	+	+	+	+	+	105,898	-230
-	-	+	-	-	-	54,958	36,706
-	-	+	+	-	-	42,958	-7,294
-	+	+	-	-	-	38,958	4,706
-	+	+	+	-	-	26,958	-39,294
-	-	+	-	+	-	5,845	-27,656

7.7.2 Delayed Delivery Case Analysis

If the delayed delivery profit > the On-Time delivery profit, the Delayed delivery profit will be positive. The data are shown in Table 7.16.

Table 7.16 Delayed Delivery Cases

						On-Time delivery	Delayed delivery	
A	B	C	D	E	F	Average	Average	Different
-	-	-	-	-	-	-4,900	63,448	68,348
-	-	-	-	-	+	-167,973	55,505	223,478
-	+	-	-	-	-	-16,900	39,448	56,348
-	-	+	-	-	+	-6,972	35,775	42,747
-	-	-	+	-	-	-8,900	35,448	44,348
-	+	-	-	-	+	-179,973	31,505	211,478
-	-	-	+	-	+	-171,973	27,505	199,478
-	+	-	+	-	-	-20,900	11,448	32,348
-	+	+	-	-	+	-22,972	3,775	26,747
-	+	-	+	-	+	-183,973	3,505	187,478
-	-	-	-	+	-	-28,727	838	29,565

As Table 7.11 shows, some conclusions can be reached:

1. Under the A- condition, the error detection rate is lower.
2. Under the debugging cost is lower (E-) condition.

If the project manager chooses Delayed delivery instead of On-Time delivery, the On-Time delivery shares the resources with Delayed delivery. The project manager needs to consider the profit gained from the On-Time delivery and the loss from Delayed delivery.

7.7.3 Determination of the Project Case Analysis

The cases where On-Time delivery profit < 0 and delayed delivery profit < 0 are analyzed in Table 7.17.

Table 7.17 No profit Cases of On-Time Delivery or Delayed Delivery

						On-Time Delivery	Delayed Delivery
A	B	C	D	E	F	Average	Average
-	+	+	+	+	+	-84,085	-104,588
-	+	+	+	+	-	-22,155	-103,656
-	-	+	+	+	+	-68,085	-72,588
-	-	+	+	+	-	-6,155	-71,656
-	+	+	-	+	+	-72,085	-60,588
-	+	+	-	+	-	-10,155	-59,656
-	+	-	+	+	+	-207,800	-59,105
-	+	-	+	+	-	-44,727	-51,162
-	+	+	+	-	+	-34,972	-40,225
-	-	-	+	+	+	-195,800	-35,105
-	+	-	-	+	+	-203,800	-31,105
-	-	+	-	+	+	-56,085	-28,588
-	-	-	+	+	-	-32,727	-27,162
-	+	-	-	+	-	-40,727	-23,162
-	-	+	+	-	+	-18,972	-8,225
-	-	-	-	+	+	-191,800	-7,105

From Table 7.17 the following conclusions can be made:

1. Under E+, the debugging cost increase also causes the profit to decrease.
2. B, C, D, and F seem not to strongly impact the termination decision.

7.8 Decision Method Applied to Four Real-World Releases

From Chapter 6 about the S-Shaped model, the cases of the Y Service Provision Company four releases estimated errors and error detection rates are shown in Table 7.18.

Table 7.18 Parameters of the Four Release Cases

	$\hat{a}$ Estimated errors	$\hat{b}$ Estimated error detection rate	The profit of On-Time delivery, interval 24 weeks
Phase 1	620	0.30	-10,986
Phase 2	448	0.43	74,010
Phase 3	418	0.43	78,945
Phase 4	463	0.38	1,650

The total profit of the four phases is \$143,619. The design of experiments consider the six decision factors. The translation of the four releases to the profit model is as derived in the Chapter 6 data analysis. If project manager chooses C+, the delivery interval of 16 weeks, and C-, the delivery interval of 12 weeks, the analysis is below in Table 7.19.

Table 7.19 Profit of the Four Release Cases

	A	B	C	D	E	F	On time	Delayed
Phase 1	-	+	+	-	-	+	-11,486	1,888
	-	+	-	-	-	+	-89,987	15,726
Phase 2	+	+	+	-	-	-	66,975	3,775
	+	+	-	-	-	-	70,002	28,443
Phase 3	+	-	+	-	-	-	74,975	44,443
	+	-	+	-	-	-	76,002	60,405
Phase 4	-	-	+	-	-	-	29,729	18,353
	-	-	-	-	-	-	-2,450	31,724

After combining the profits from Table 7.19, the sorted profits from the cost model are shown in Table 7.20.

Table 7.20 Sorted profit Table of Different Release Policies (1 means on time, 2 means delayed)

	Release1	Reelase2	Releas3	Release4	Profit 1	Profit 2	Profit 3	Profit 4	Total profit
12-week	2	1	1	2	15,753	70,002	76,002	31,724	193,480
12-week	2	1	2	2	15,753	70,002	60,405	31,724	177,883
16-week	2	1	1	1	1,888	66,975	74,975	29,729	173,566
12-week	2	2	1	2	15,753	48,405	76,002	31,724	171,883
16-week	2	1	1	2	1,888	66,975	74,975	18,353	162,190
16-week	1	1	1	1	-11,486	66,975	74,975	29,729	160,192
12-week	2	1	1	1	15,753	70,002	76,002	-2,450	159,306
12-week	2	2	2	2	15,753	48,405	60,405	31,724	156,286
16-week	1	1	1	2	-11,486	66,975	74,975	18,353	148,816
12-week	2	1	2	1	15,753	70,002	60,405	-2,450	143,709
16-week	2	1	2	1	1,888	66,975	44,443	29,729	143,034
12-week	2	2	1	1	15,753	48,405	76,002	-2,450	137,709
16-week	2	2	1	1	1,888	28,443	74,975	29,729	135,034
16-week	2	1	2	2	1,888	66,975	44,443	18,353	131,658
16-week	1	1	2	1	-11,486	66,975	44,443	29,729	129,660
16-week	2	2	1	2	1,888	28,443	74,975	18,353	123,658
12-week	2	2	2	1	15,753	48,405	60,405	-2,450	122,112
16-week	1	2	1	1	-11,486	28,443	74,975	29,729	121,660
16-week	1	1	2	2	-11,486	66,975	44,443	18,353	118,284
16-week	1	2	1	2	-11,486	28,443	74,975	18,353	110,284
16-week	2	2	2	1	1,888	28,443	44,443	29,729	104,502
16-week	2	2	2	2	1,888	28,443	44,443	18,353	93,126
16-week	1	2	2	1	-11,486	28,443	44,443	29,729	91,128
12-week	1	1	1	2	-89,987	70,002	76,002	31,724	87,741
16-week	1	2	2	2	-11,486	28,443	44,443	18,353	79,752
12-week	1	1	2	2	-89,987	70,002	60,405	31,724	72,144
12-week	1	2	1	2	-89,987	48,405	76,002	31,724	66,144
12-week	1	1	1	1	-89,987	70,002	76,002	-2,450	53,567
12-week	1	2	2	2	-89,987	48,405	60,405	31,724	50,547
12-week	1	1	2	1	-89,987	70,002	60,405	-2,450	37,970
12-week	1	2	1	1	-89,987	48,405	76,002	-2,450	31,970
12-week	1	2	2	1	-89,987	48,405	60,405	-2,450	16,373

The best profit occurs when the delivery interval becomes 12 weeks.

7.9 Comparison with Zhang's Model

In the paper by X. Zhang and H. Pham, (1998) the expected total software cost can be expressed as

$E(T) = C_1T+C_2m(T)uy+C_3[1-R(x/T)]$ Where

E(T)	expected total cost of a software by time T
C1	software test cost per unit time
C2	cost of removing each error per unit time during testing
C3	risk cost due to system failure
Uy	expected time to remove an error
R(x/T)	reliability function of software by time T for a mission time x

Case 1

Using Zhang's data input into the Multiple-Release Model:

Case 1: C1=$600, C2=$20, Cw=$10, C3=$20,000, Cp =$2000,00 and x= 1 hour.

The calculation results are shown in Table 7.21.

Table 7.21 In Zhang's Model, the Market Drop Rate at 0.01/hour and 0.05/hour

Delivery after T hours Testing	Zhang's model	Multiple Release, 0.01 drop rate	Multiple Release 0.05 drop rate
48	58,000	22,195	-361,804
49	58,911	24,681	-367,318
50	59,183	26,663	-37,336
51	59,356	28,191	-379,808
52	59,441	29,307	-393,948
53	* 59,448	30,051	-393,948
54	59,386	30,460	-401,539
55	59,261	* 30,566	-409,433
56	59,082	30,399	-417,600

The optimal delivery time of the Multiple-Release Model at drop 0.01/hour is 55 hours is similar to Zhang's model of 53 hours. However, if the market drop rate is put in as 0.05/hour, the software company may lose $393,948. The Multiple-Release model can reflect the software market drop and give a better release decision.

Case 2

Case 2 uses the software development costs from the X Juice Company case input into Zhang's model. Case 2 C1 =\$5000, C2=\$1,000, C3=\$4,000, and the Uy=0 if the project manager does not consider the market value drop factor P. For Y Service Provision Company case release 1.

$$\hat{a} = 620$$

$$\hat{b} = 0.3$$

$$\hat{m} = 620(1 - e^{-0.3})$$

$$R = (x/t) = e^{-(620)[e^{-(0.3)t} - e^{-(0.3)(T+x)}]}$$

Put P=0 and the mission time at T to next release 2T so the software reliability during the T and 2T is R(2T/T).

The above data are input into Zhang's model. The results are shown in Table 7.22, the data from the proposed method into Zhang's model.

Table 7.22 Optimal Release Time for C1=\$5000, C2=\$1,000, C3=\$4,000

Release Time T*(week)	C1T	C2.m(T).Uy	C3[1-R(x/T)]	Expected Total cost E(T)
1	5,000	160,693	4,000	169,693
2	10,000	279,737	4,000	293,737
3	15,000	367,927	4,000	386,927
4	20,000	433,260	4,000	457,260
5	25,000	481,659	4,000	510,659
6	30,000	517,515	4,000	551,515
7	35,000	544,077	4,000	583,077
8	40,000	563,755	4,000	607,755
9	45,000	578,333	4,000	627,333
10	50,000	589,132	4,000	643,132
11	55,000	597,132	4,000	656,132
12	60,000	603,059	4,000	667,059
13	65,000	607,450	4,000	676,450
14	70,000	610,703	4,000	684,702
15	75,000	613,112	3,996	692,108
16	80,000	614,898	3,975	698,872

Clearly, the first week has the lowest total cost, \$169,692. The reason is that Zhang's model counts the penalty of the reliability at most at \$4,000, so the rather low penalty cost can almost be ignored compared with the testing Cost C_1 and the debugging cost C_2. The delivery error penalty is low and the testing and debugging costs are high. Earlier delivery will be better. Also, Zhang's model uses the next hour to judge the cost of the error penalty. The Multiple-Release Model uses the next delivery time frame to get the error penalty cost.

7.10 Comparison with Zheng's model

Defined: $n_{1,t} := \min\{ n : \Psi(n,t) - \Phi(n,t) < 0\},$

$n_{2,t} = \min\{ n : Cp - \min\{ \Phi(n,t), \Psi(n,t)\} < 0\}$

If $N_{1,t} <= N <= N_{2,t}$ then release the system. Where

Φ (n, t) discounted fixing and penalty cost after release if the project manager stops testing at t;

Ψ (N, t) the discounted cost-to-go if the project manager continues testing system at time t.

$$\Phi(n,t) = C_R.E[Xn(t)](1 - e^{-b})e^{-bt}(\frac{1 - e^{-(a+b)T_l}}{1 - e^{-(a+b)}})$$

Ψ (n,t) = Ct+ Cr E[Dn(t)] if no variance

b	error detection rate
X = x ,	expected number of errors detected up to time t
a	discount factor

C_R	penalty cost
Cr	bug removal cost
Ct	testing cost
Dn (t)	number of errors within a unit of time following t
TL	warranty period

Inputting the data from the proposed model gets Table 7.20:

$\alpha = 0$ is the discount; factor b = 0.3 the detection rate; the expected errors E [Xn (t)] =620;

$m(t) = 620[1-(1+bt)e^{-bt}]$,

Cr= \$1,000,Ct= \$5,000,C_R= \$4,000,Dn = m(t+1)- m(t) ,TL =6 months

The calculation results are shown in Table 7.23.

Table 7.23 Shaohui Zheng's Model

Weeks	M (t)	Φ(n,t)	Ψ (n,t)
1	22.90	828,936	57,678
2	75.58	463,972	70,482
3	141.06	307,736	73,110
4	209.17	215,226	69,977
5	274.15	154,098	63,893
6	333.04	111,703	56,598
7	384.64	81,558	49,128
8	428.77	59,819	42,064
9	465.83	44,004	35,698
10	496.53	32,436	30,141
11	521.67	23,943	25,403
12	542.07	17,691	21,432
13	558.51	13,081	18,149
14	571.65	9,677	15,464
15	582.12	7,161	13,288
16	590.41	5,301	11,536
17	596.94	3,925	10,136
18	602.08	2,907	9,023
19	606.10	2,153	8,141
20	609.24	1,594	7,447
21	611.69	1,181	6,901
22	613.59	875	6,474

Clearly, the cost to stop testing Φ (n, t) is larger than the cost Ψ (n, t) of continuous testing up to 11 months. The better decision is to deliver the software at 12 months. The drawback of Zheng's model is the decision policy. From Zheng's model, when the testing cost is larger than the penalty cost, the software is delivered. However, the first match unit time may not always be the optimal decision. The later unit time may generate better profit than the first match unit time.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

According to the objectives of the research, the software cost analysis is derived from the case study of Chapter 6. The Multiple-Release model and software lifecycle conditions were analyzed in Chapter 7. Chapter 8 contains the conclusions and future research direction.

8.1 Software Cost Analysis

From the ABC analysis, the personnel expense is around 88% of the total software project cost. A decrease in the personnel cost can significantly lower the total project cost. The estimated cost of the X Juice Company software staff is $112,000 compared with the consultant firm staff estimated cost of $65,000. The software project of out-sourcing to consultant firms is a good solution to lower the total costs. Considering the software testing cost from the consultant firm, the hourly rate is around $32.5 /hour ($65,000/2000 hours per year). The weekly rate is around $1,300 per staff. It is a relatively small portion of the project cost; however, the testing quality impacts the total software project profit dramatically. It is a better policy to out-source the testing portion to a consultant firm.

8.2 Multiple-Release and Life Cycle Analysis

Applying the Multiple-Release models into the 4-phase Y Service Provision

Company case, the optimal delivery timing is a delivery interval of 12 weeks between each release. The delivery time is one year, so the first release is one year + 12 weeks (three months). In the same development resource, each phase, divided into two releases, can improve the profit from \$143,619 to \$193,480 in a 2-year delivery time frame. This also saves three months, total delivery time. The delivery time frame is shown in Table 8.1.

Table 8.1 Optimal Release Intervals of the four-Release Cases

YY/MM	PHASE 1		PHASE 2		PHASE 3		PHASE 4	
	Release 1	Release2	Release 3	Release 4	Relase 5	Release 6	Release 7	Release 8
97/12	5							
98/1	24							
98/2	2							
98/3	37	5						
98/4	34	24						
98/5	38	2						
98/6	50	37	5					
98/7	23	34	43					
98/8	16	38	42					
98/9	15	50	26	5				
98/10	20	23	21	43				
98/11	11	16	19	42				
98/12	5	15	19	26	6			
99/1	2	20	21	21	16			
99/2	0	11	13	19	42			
99/3	2	5	7	19	41	6		
99/4		2	5	21	51	16		
99/5		0	3	13	11	42		
99/6		2	0	7	10	41	3	
99/7			0	5	12	51	9	
99/8			1	3	7	11	54	
99/9				0	1	10	42	3
99/10				0		12	22	9
99/11				1		7	28	54
99/12						1	34	42
00/1							12	22
00/2							12	28
00/3							1	34
00/4							1	12
00/5								12
00/6								1
00/7								1
00/8								
00/9								
00/10								

The above total delivery time from 12/1997 to 07/2000 can save 3 months' software delivery time. Factor A-, the error detection rate/week, is the dominant factor compared with the 0.5 error detection rate and 0.3 error detection rate. There is an $188,921 benefit improvement. The total profit is $386,959 of these four-phase releases. As of the improvement of the factors D and E, the testing cost and the debugging cost, it is usual to hire lower cost contractors or out-source projects to other companies or other countries.

From the above analysis, the results of the 4-phased releases are that during the software project development of the first release, the error number is large and the error detection rate is slow. The later release can prevent some errors and get a better error detection rate.

The End of Software Life Cycle Conditions

From the analysis in Chapter 7, the project manager can know that under some conditions, no matter whether there is On-Time delivery of the software or delayed delivery of the software, the project cannot get any profit. The situation summary is below:

1. Lower A-: the detection rate is the dominant factor to terminate the project from the profit point of view.
2. E+: the debugging cost is higher, so it is not worthwhile to put resources to do the debugging.
3. B, C, D, and F: seem not to strongly impact the termination decision.

Comparison with Existing Software Cost Models

Zhang's model

In Zhang's model the software delivery penalty comes from software reliability. After testing and debugging, and improving the reliability, the penalty cost will become $4,000 times (1- reliability). However, Zheng's model is not valid for large commercial software like an Internet case. In large commercial software, it is almost impossible to prevent the errors from happening, however, during a set of periods, for example, one month or one year. Zhang's model gives you a similar result -- $4,000. In the Y Service Provision Company case, there are 620 errors in the first releases. It should be 620 errors x $4,000 = $2,480,000 penalty cost. If Zheng's software cost model is followed, the company loses $2,476,000. If the cost information from Zheng's model is applied, e.g. to input Zhang's example, the result comes from Multiple-Release model in Case 1: C1=$600, C2=$20, Cw=$10, C3=$20,000, Cp =$2000,00 an x= 1.0 hour, the optimal delivery time is 55 hours which is similar to Zhang's model of 53 hours. However, if the market drop rate was put in at 0.05/hour, clearly, when the software value is under a larger market drop, no profit will be made for the release. Zhang's model cannot reflect the market value. Also, the testing cost of $600 and the debugging cost $20 are relatively low, so Zhang's model can be driven to continuous testing. So the pretty low penalty cost almost can be ignored. Also Zhang's model uses the next hour to judge the total cost. The proposed mode considers the cost during one release and should use the delivery interval as the total cost.

Zheng's model

Zheng's model only considers the to-go cost Ψ (n, t) and the penalty cost Φ (n, t) without the market value drop, and the detection rate will change during the warranty period. Another drawback is that the estimation errors number "a " and the detection rate "b" are unknown beforehand. Users cannot make good decisions using a small number of error data. If the error data have not come to steady state, this may give users a wrong direction.

8.3 Future Research

Obviously, software technology updates very quickly. The life cycle of the software is shortened excessively. Managerial staffs must know the cost and use the budget wisely. The presented methodology can be used to help them make the best decisions by offering detailed and accurate cost analysis results for a complex software project.

Not much existing research effort has resulted in Software Quality Assurance (SQA), so further research may address the issues of SQA. Exploration of the issue of quality and opportunity cost may be a future research topic. For a sustainable software industry firm, the human quality is a key component to success, so proper training and motivation are needed.

The relationship between the training cost and performance output needs to be explored further. To summarize the further research to construct the whole Multiple-Release decision model, the following directions are pointed out:

- Propose the spectrum of the software's bug-solving difficulty and look into the relationship of difficulty level and debugging/fixing time.

- Catalogue errors as to severity level and penalty cost index to provide reference for a future software maintenance and for measuring warranty or fixing response time when working out contracts with customers.
- Study the testing group's performance as its members increase.
- Study the relative effects between the financial incentive and course training in improving the performance of the software development staff.

In future research, as described above, software reliability and cost measurement can be integrated to maximize the business profit under limited resource requirements. Different programming languages used as software development tools, for example JAVA or C++ that are widely used today, may have different error patterns. The exploration into the issue of errors or problems that are generated by the programming language can be a good direction for the next generation language's implementation.

Going back to the basic software reliability theory of Geol's Non-Homogeneous Poisson Process (NHPP) model (1979), which is the instructed model and widely used in industrial fields, NHPP is not the only choice of modeling the error data behavior, despite its wide popularity. Other methods, which are presented by other researchers, may derive different error data parameters and produce different release decision results. So, comparing and trying different methods may lead to a better choice that satisfies the software industry's specific needs. For a researcher, to provide a more robust and easy to use model to match the new programming languages and hardware architectures is a direction for further research. The application of decision support or expert system needs to relocate resources and train people to use it. It needs extra costs and human resources to create a user-friendly expert system. The issue of how to integrate it into the company's mainstream operation needs to be addressed in the future.

APPENDIX A

NHPP DATA FITTING

The following are the NHPP data fitting and cost simulation programs.

```
//This program input the errors data into NHPP fitting method
#include<iostream.h>
#include<stdlib.h>
extern void pow1(int, float[] , float&, float&);
extern void m(int, float&, float&);

int main(){
float
release_1[]={0,10,58,93,167,234,310,409,455,486,515,555,576,586,58
9,589,592};
float
release_2[]={0,9,95,178,229,270,309,346,388,414,427,437,442,442,44
2,443};
float release_3[]={0,12,43,127,208,307,330,350,373,387,389};

float release_4[]={0,5,23,131,214,257,312,379,402,426,427,428};
int n;
 float a_release,b_release=0;
 int data_1=16;
 int data_2=15;
 int data_3=10;
 int data_4=11;

 // printf("The on time delivery a= estimated errors number, b=
learning factor\n");
 for (int j=1; j<=data_1;j++)
 {
 a_release=500;//defaul a
 b_release=0.1;//defaul a
 pow1(j, release_1, a_release, b_release);
//   printf("total data= %3d the estimate a = %12.2f      the
estimate b = %12.2f\n",j,a_release, b_release);
  }
  printf("total data= %3d the estimate a = %12.2f    the estimate
b = %12.2f\n",j,a_release, b_release);
 m(data_1, a_release, b_release);
 // printf("\n\nThe release 2 a= estimated errors number, b=
learning factor\n\n");
 for (j=1; j<=data_2;j++)
 {
 a_release=500;//defaul a
 pow1(j, release_2, a_release, b_release);
 // printf("total data= %3d the estimate a = %12.2f      the
estimate b = %12.2f\n",j,a_release, b_release);
  }

  printf("\n\ntotal data= %3d the estimate a = %12.2f      the
estimate b = %12.2f\n",j,a_release, b_release);
 m(data_2, a_release, b_release);
 // printf("\n\nThe release 3 a= estimated errors number, b=
learning factor\n\n");
 for (j=1; j<=data_3;j++)
 {
 a_release=500;//defaul a
 pow1(j, release_3, a_release, b_release);
  //printf("total  data=  %3d  the  estimate  a  =  %12.2f       the
estimate b = %12.2f\n",j,a_release, b_release);
  }

  printf("\n\ntotal  data= %3d the estimate a  =  %12.2f       the
estimate b = %12.2f\n",j,a_release, b_release);
 m(data_3, a_release, b_release);
  //printf("\n\nThe  release  4  a=  estimated  errors  number,  b=
learning factor\n\n");
 for (j=1; j<=data_4;j++)
 {
 a_release=500;//defaul a
 pow1(j, release_4, a_release, b_release);
  //printf("total  data=  %3d  the  estimate  a  =  %12.2f       the
estimate b = %12.2f\n",j,a_release, b_release);
  }

  printf("\n\ntotal  data= %3d the estimate a  =  %12.2f       the
estimate b = %12.2f\n",j,a_release, b_release);
 m(data_4, a_release, b_release);

//   cout<<"total data= "<<j<<"  the estimate a = "<<a_release<<"
the estimate b = "<<b_release<<endl;
return 0;
}


//This program will limit the data parameters a and b
#include <stdlib.h>
#include <iostream.h>
#include <math.h>
void pow1(int in, float y[], float& in1, float& in2)
{
 float yn,module_1, module_2,module_3, module_4;
 float temp_1,temp_2,temp_3,temp_4,temp_5,temp_6;
 yn=y[in];
 int tn=in;
 float testnew=0;
 float test=100000;
 float b;
 for (float test_num = 0; test_num<=1000; test_num++)

 {
   in2=  test_num/1000;
          //cout <<"in2 "<<in2<<endl;

              module_1=yn*tn*tn*exp(-1*in2*tn);
              //cout <<"module_1 "<<module_1<<endl;
              module_2=1-(1+in2*tn)*exp(-1*in2*tn);
              //cout <<"module_2 "<<module_2<<endl;
              module_3=module_1/module_2;
              //cout <<"module_3 "<<module_3<<endl;
              float temp=0;
          for (int num =1; num<=in; num++)
             {
                int num1=num-1;
                temp_1=(y[num]-y[num1]);
                temp_2=num*num*exp(-1*in2*num);
                temp_3=(num1)*(num1)*exp(-1*in2*num1);
                temp_4=(1+in2*(num1))*exp(-1*in2*num1);
                temp_5=(1+in2*(num))*exp(-1*in2*num);
                temp=temp+(temp_1*(temp_2-temp_3))/(temp_4-
temp_5);
             }
           module_4=temp;
          // cout <<"module_4 "<<module_4<<endl;
         testnew=module_4-module_3;
           //cout <<"newtest "<<testnew<<endl;
           //cout <<"test "<<test<<endl;
         if (abs(testnew)< test)
         {
          test=abs(testnew);
          b=in2;
           }
 }

  //cout <<"b=  "<<b<<endl;
  temp_6=1-(1+b*in)*exp(-1*b*in);
// cout <<"temp_6 "<<temp_6<<endl;
 float a=in1;
 in1=yn/temp_6;
 in2=b;

 }
//The program will generate the up and low case of the 6 factors
#include <stdlib.h>
#include <iostream.h>
#include <math.h>

extern void m2(int,float, float,float,int,int,int,int);
   float rand1(int);

int main() {

int initial_time;

float estimate_b[]={0,0.5,0.3};
float P[]={0,0.02,0.01};  //profit decay rate /week
int Ti[]={0,16,12};
int C1[] = {0,5000,3000};

//Testing cost/week  ,  5 workers X 20 dollars /hours X  8 hours X
5 days/week
int C2[]={0,1000,500};    // Debugging cost/case ,  25 workers X
50 dollars
int C3[]={0,4000,2000};     //penalty cost/errors,   system shut
down

char simple[]={'*','+','-'};

int Cr=0;
int release,i1,i2,i3,i4,i5,i6,i7;
int Decision_Time=0;
float Cost_C1;
float Cost_C2;
float Cost_C3;
float Total_Profit;
float estimate_a[]={0,500,600,700};
float k=1; // the sd of the data close to error estimation;
float k1;
int pre_seed=15;
int err=600;//center of the errors
   for (int ra=1; ra<=3; ra++)  //randon number to make the range
      {
          k=rand1(rand()+pre_seed);
          k1=k/65536;
          if (k1 >  0.5)
           // {
            k1=k1-0.5;
            k1=-1*k1;
            //}
          estimate_a[ra]=err+k1*err ;
       printf("estimate a= %10.8f\n",estimate_a[ra]);
      }
   for (i2=1; i2<=2; i2++)
    for (i3=1; i3<=2; i3++)
       for (i4=1; i4<=2; i4++)
         for (i5=1; i5<=2; i5++)
           for (i6=1; i6<=2; i6++)
             for (i7=1; i7<=2; i7++)
               for (release=1;release<=2;release++)
               {
         for (i1=1; i1<=3; i1++)
             {
             if (i1==1&&release==1)
             {
             if (i1==1&&release==1)
     printf("%2c        :%2c        :%2c          :%2c          :%2c:
%2c",simple[i2],simple[i3],simple[i4],simp
le[i5], simple[i6],simple[i7]);

    //printf("%3d,%10.2f,%10.2f,%10.2f,%3d,%3d,%3d,%3d",   release,
estimate_a[i1]
,estimate_b[i2],P[i3],Ti[i4],C1[i5],C2[i6],C3[i7]);
     m2(release,
estimate_a[i1],estimate_b[i2],P[i3],Ti[i4],C1[i5],C2[i6],C3[i7]
);
             if (i1==3&&release==2)
     printf("\n");
                }}

}
   float rand1(int pseed)
   {
    static short seed=1;
   seed =(pseed *25173 +3849) % 65536;
    return (seed);
    }

//The program will generate the up and low case of the 6 factors
#include <stdlib.h>
#include <iostream.h>
#include <math.h>

extern void m1(int,float, float,float,int,int,int,int);

int main() {

int initial_time;
float estimate_a[]={0,400,500,600};
float estimate_b[]={0,0.4,0.3};
float P[]={0,0.005,0.001};  //profit decay rate /week
int Ti[]={0,12,10};
int C1[] = {0,4000,3000};

//Testing cost/week  ,  5 workers X 20 dollars /hours X  8 hours X
5 days/week
int C2[]={0,1000,500};     // Debugging cost/case ,  25 workers X
50 dollars
int C3[]={0,5000,2500};     //penalty cost/errors,  system shut
down

char simple[]={'*','+','-'};

int Cr=0;
int release,i1,i2,i3,i4,i5,i6,i7;
int Decision_Time=0;
float Cost_C1;
float Cost_C2;
float Cost_C3;
float Total_Profit;

   for (i2=1; i2<=2; i2++)
    for (i3=1; i3<=2; i3++)
       for (i4=1; i4<=2; i4++)
         for (i5=1; i5<=2; i5++)
           for (i6=1; i6<=2; i6++)
             for (i7=1; i7<=2; i7++)
             for (release=1;release<=2;release++)
             {
         for (i1=1; i1<=3; i1++)
           {
           if (i1==1&&release==1)
     printf("%2c        :%2c        :%2c        :%2c        :%2c:
%2c",simple[i2],simple[i3],simple[i4],simple[i5],
simple[i6],simple[i7]);

    //printf("%3d,%10.2f,%10.2f,%10.2f,%3d,%3d,%3d,%3d",  release,
estimate_a[i1],estimate_b[i2],P[i3],Ti[i4],C1[i5],C2[i6],C3[i7]);
     m1(release,
estimate_a[i1],estimate_b[i2],P[i3],Ti[i4],C1[i5],C2[i6],C3[i7]);
           if (i1==3&&release==2)
     printf("\n");
            }}
}
//This program will input the factors from money.C and calculate
the cost
//and total profit
#include <stdlib.h>
#include <iostream.h>
#include <math.h>

int initial_time;
int C1=30000;   //Testing cost/week  ,  25 workers X 30 dolloars
/hours X  8 hou
rs X 5 days/week
int C2=1250;    // Debugging cost/case ,  25 workers X 50 dollars
int C3=2000;    //penalty cost/errors, wait for next release
int C4=40000;    //penalty cost/errors,  system shut down
int Cr=0;
```

```
float Profit;  //profit per release
int contract=1000000;
int Ti=16/4;
int Delivery_Time=12;
float Cost_C1;
float Cost_C2;
float Cost_C3;
float Cost_C4; //installation fee
float Total_Profit;

void  m2(int  N,  float  estimate_a,  float  estimate_b,float  P,int
Ti,int C1,int C2,i
nt C3)
{
// initial_time=in;
 float estimate_m[50];
          for (int num =1; num<=Ti*2; num++)
             {
 estimate_m[num]=           estimate_a*(1-(1+estimate_b*num)*exp(-
1*estimate_b*num));
        //    printf("\nTime=    %3d    estimate_m=    %5.2f    ",
num,estimate_m[num]);
              }        //end of num

         //  for (int N =1; N<=2; N++)
         // {
 //printf("\n The Realease (1 mean on time, 2 mean delay to next)
%3d     ", N);
Cost_C1= C1*(Delivery_Time+Ti*N);
 //Cost_C1= C1*(Ti*N-Decision_Time);
// printf("\nThe total testing  Cost_C1= %10.2f ", Cost_C1);
//         Cost_C2=          C2*(estimate_m[(initial_time+1+Ti*N)]-
estimate_m[initial_time+1]);
 Cost_C2= C2*(estimate_m[(Delivery_Time+Ti*N)]);
// printf("\nThe total debug     Cost_C2= %10.2f", Cost_C2);
 Cost_C3= C3*(estimate_a-estimate_m[Delivery_Time])*Ti/48;
 //printf("\nThe total penalty   Cost_C3= %10.2f", Cost_C3);
 Cost_C4= C4*48/Ti;
 //printf("\nThe total penalty   Cost_C3= %10.2f", Cost_C3);
 float rate =P*Ti*N;
 Profit=contract-contract*rate;
// printf("\nThe profit = %10.2f", Profit);
 Total_Profit= Profit-(Cost_C1+Cost_C2+Cost_C3+Cost_C4);
 //printf("\nTotal_Profit= %10.2f\n", Total_Profit);
 printf(": %10.2f", Total_Profit);
          //}
 }
```

APPENDIX B

DESIGN OF EXPERIMENTS RESULTS

The following are the total profits values generated from the design of experiments.

Table B.1 Factorial Results of On-Time Delivery.

A	B	C	D	E	F	P1	P2	P3	Average	S	Log (S)	S/N
+	+	+	+	+	+	96,690	98,673	122,332	105898	14267	4.15	100
+	+	+	+	+	-	101,731	103,471	124,244	109815	12526	4.10	101
+	+	+	+	-	+	112,304	113,537	128,254	118032	8874	3.95	101
+	+	+	+	-	-	117,345	118,336	130,166	121949	7133	3.85	102
+	+	+	-	+	+	108,690	110,673	134,332	117898	14267	4.15	101
+	+	+	-	+	-	113,731	115,471	136,244	121815	12526	4.10	102
+	+	+	-	-	+	124,304	125,537	140,254	130032	8874	3.95	102
+	+	+	-	-	-	129,345	130,336	142,166	133949	7133	3.85	103
+	+	-	+	+	+	88,797	92,502	136,719	106006	26663	4.43	100
+	+	-	+	+	-	117,766	120,081	147,706	128518	16658	4.22	102
+	+	-	+	-	+	98,429	101,672	140,372	113491	23336	4.37	101
+	+	-	+	-	-	127,398	129,251	151,359	136003	13331	4.12	103
+	+	-	-	+	+	92,797	96,502	140,719	110006	26663	4.43	100
+	+	-	-	+	-	121,766	124,081	151,706	132518	16658	4.22	102
+	+	-	-	-	+	102,429	105,672	144,372	117491	23336	4.37	101
+	+	-	-	-	-	131,398	133,251	155,359	140003	13331	4.12	103
+	-	+	+	+	+	112,690	114,673	138,332	121898	14267	4.15	102
+	-	+	+	+	-	117,731	119,471	140,244	125815	12526	4.10	102
+	-	+	+	-	+	128,304	129,537	144,254	134032	8874	3.95	103
+	-	+	+	-	-	133,345	134,336	146,166	137949	7133	3.85	103
+	-	+	-	+	+	124,690	126,673	150,332	133898	14267	4.15	102
+	-	+	-	+	-	129,731	131,471	152,244	137815	12526	4.10	103
+	-	+	-	-	+	140,304	141,537	156,254	146032	8874	3.95	103
+	-	+	-	-	-	145,345	146,336	158,166	149949	7133	3.85	103
+	-	-	+	+	+	100,797	104,502	148,719	118006	26663	4.43	101
+	-	-	+	+	-	129,766	132,081	159,706	140518	16658	4.22	103
+	-	-	+	-	+	110,429	113,672	152,372	125491	23336	4.37	102
+	-	-	+	-	-	139,398	141,251	163,359	148003	13331	4.12	103
+	-	-	-	+	+	104,797	108,502	152,719	122006	26663	4.43	101
+	-	-	-	+	-	133,766	136,081	163,706	144518	16658	4.22	103
+	-	-	-	-	+	114,429	117,672	156,372	129491	23336	4.37	102
+	-	-	-	-	-	143,398	145,251	167,359	152003	13331	4.12	104

Table B.1 (Continued)

A	B	C	D	E	F	P1	P2	P3	Average	S	Log (S)	S/N
-	+	+	+	+	+	-147,789	-134,074	29,608	-84085	98699	4.99	94
-	+	+	+	+	-	-68,095	-58,204	59,833	-22155	71176	4.85	96
-	+	+	+	-	+	-84,588	-73,906	53,578	-34972	76872	4.89	97
-	+	+	+	-	-	-4,895	1,963	83,804	26958	49350	4.69	70
-	+	+	-	+	+	-135,789	-122,074	41,608	-72085	98699	4.99	96
-	+	+	-	+	-	-56,095	-46,204	71,833	-10155	71176	4.85	95
-	+	+	-	-	+	-72,588	-61,906	65,578	-22972	76872	4.89	96
-	+	+	-	-	-	7,106	13,963	95,804	38958	49350	4.69	81
-	+	-	+	+	+	-315,023	-291,938	-16,440	-207800	166124	5.22	89
-	+	-	+	+	-	-105,173	-92,159	63,151	-44727	93651	4.97	98
-	+	-	+	-	+	-284,361	-262,747	-4,810	-183973	155535	5.19	78
-	+	-	+	-	-	-74,511	-62,969	74,780	-20900	83062	4.92	97
-	+	-	-	+	+	-311,023	-287,938	-12,440	-203800	166124	5.22	87
-	+	-	-	+	-	-101,173	-88,159	67,151	-40727	93651	4.97	98
-	+	-	-	-	+	-280,361	-258,747	-810	-179973	155535	5.19	63
-	+	-	-	-	-	-70,511	-58,969	78,780	-16900	83062	4.92	97
-	-	+	+	+	+	-131,789	-118,074	45,608	-68085	98699	4.99	97
-	-	+	+	+	-	-52,095	-42,204	75,833	-6155	71176	4.85	94
-	-	+	+	-	+	-68,588	-57,906	69,578	-18972	76872	4.89	96
-	-	+	+	-	-	11,106	17,963	99,804	42958	49350	4.69	84
-	-	+	-	+	+	-119,789	-106,074	57,608	-56085	98699	4.99	98
-	-	+	-	+	-	-40,095	-30,204	87,833	5845	71176	4.85	92
-	-	+	-	-	+	-56,588	-45,906	81,578	-6972	76872	4.89	95
-	-	+	-	-	-	23,106	29,963	111,804	54958	49350	4.69	90
-	-	-	+	+	+	-303,023	-279,938	-4,440	-195800	166124	5.22	78
-	-	-	+	+	-	-93,173	-80,159	75,151	-32727	93651	4.97	98
-	-	-	+	-	+	-272,361	-250,747	7,190	-171973	155535	5.19	82
-	-	-	+	-	-	-62,511	-50,969	86,780	-8900	83062	4.92	96
-	-	-	-	+	+	-299,023	-275,938	-440	-191800	166124	5.22	58
-	-	-	-	+	-	-89,173	-76,159	79,151	-28727	93651	4.97	98
-	-	-	-	-	+	-268,361	-246,747	11,190	-167973	155535	5.19	86
-	-	-	-	-	-	-58,511	-46,969	90,780	-4900	83062	4.92	95

Table B.2 Factorial Results of Delayed Delivery.

A	B	C	D	E	F	P1	P2	P3	Average	S	Log (S)	S/N
+	+	+	+	+	+	-7,754	-6,134	13,198	-230	11657	4.07	-230
+	+	+	+	+	-	-7,750	-6,131	13,199	-227	11656	4.07	-227
+	+	+	+	-	+	9,120	9,930	19,598	12883	5830	3.77	12883
+	+	+	+	-	-	9,123	9,933	19,599	12885	5829	3.77	12885
+	+	+	-	+	+	36,246	37,866	57,198	43770	11657	4.07	43770
+	+	+	-	+	-	36,250	37,869	57,199	43773	11656	4.07	43773
+	+	+	-	-	+	53,120	53,930	63,598	56883	5830	3.77	56883
+	+	+	-	-	-	53,123	53,933	63,599	56885	5829	3.77	56885
+	+	-	+	+	+	48,051	49,680	69,124	55619	11724	4.07	55619
+	+	-	+	+	-	48,184	49,807	69,175	55722	11678	4.07	55722
+	+	-	+	-	+	64,892	65,713	75,511	68706	5908	3.77	68706
+	+	-	+	-	-	65,026	65,840	75,562	68809	5862	3.77	68809
+	+	-	-	+	+	76,051	77,680	97,124	83619	11724	4.07	83619
+	+	-	-	+	-	76,184	77,807	97,175	83722	11678	4.07	83722
+	+	-	-	-	+	92,892	93,713	103,511	96706	5908	3.77	96706
+	+	-	-	-	-	93,026	93,840	103,562	96809	5862	3.77	96809
+	-	+	+	+	+	24,246	25,866	45,198	31770	11657	4.07	31770
+	-	+	+	+	-	24,250	25,869	45,199	31773	11656	4.07	31773
+	-	+	+	-	+	41,120	41,930	51,598	44883	5830	3.77	44883
+	-	+	+	-	-	41,123	41,933	51,599	44885	5829	3.77	44885
+	-	+	-	+	+	68,246	69,866	89,198	75770	11657	4.07	75770
+	-	+	-	+	-	68,250	69,869	89,199	75773	11656	4.07	75773
+	-	+	-	-	+	85,120	85,930	95,598	88883	5830	3.77	88883
+	-	+	-	-	-	85,123	85,933	95,599	88885	5829	3.77	88885
+	-	-	+	+	+	72,051	73,680	93,124	79619	11724	4.07	79619
+	-	-	+	+	-	72,184	73,807	93,175	79722	11678	4.07	79722
+	-	-	+	-	+	88,892	89,713	99,511	92706	5908	3.77	92706
+	-	-	+	-	-	89,026	89,840	99,562	92809	5862	3.77	92809
+	-	-	-	+	+	100,051	101,680	121,124	107619	11724	4.07	107619
+	-	-	-	+	-	100,184	101,807	121,175	107722	11678	4.07	107722
+	-	-	-	-	+	116,892	117,713	127,511	120706	5908	3.77	120706

Table B.2 (Continoued)

A	B	C	D	E	F	P1	P2	P3	Average	S	Log (S)	S/N
-	+	+	+	+	+	-142,046	-133,981	-37,736	-104588	58036	4.76	-104588
-	+	+	+	+	-	-140,848	-132,840	-37,281	-103656	57622	4.76	-103656
-	+	+	+	-	+	-59,222	-55,132	-6,322	-40225	29432	4.47	-40225
-	+	+	+	-	-	-58,023	-53,991	-5,868	-39294	29018	4.46	-39294
-	+	+	-	+	+	-98,046	-89,981	6,264	-60588	58036	4.76	-60588
-	+	+	-	+	-	-96,848	-88,840	6,719	-59656	57622	4.76	-59656
-	+	+	-	-	+	-15,222	-11,132	37,678	3775	29432	4.47	3775
-	+	+	-	-	-	-14,023	-9,991	38,132	4706	29018	4.46	4706
-	+	-	+	+	+	-99,580	-90,866	13,131	-59105	62710	4.80	-59105
-	+	-	+	+	-	-89,359	-81,135	17,008	-51162	59180	4.77	-51162
-	+	-	+	-	+	-19,011	-14,164	43,689	3505	34885	4.54	3505
-	+	-	+	-	-	-8,790	-4,433	47,566	11448	31355	4.50	11448
-	+	-	-	+	+	-71,580	-62,866	41,131	-31105	62710	4.80	-31105
-	+	-	-	+	-	-61,359	-53,135	45,008	-23162	59180	4.77	-23162
-	+	-	-	-	+	8,989	13,836	71,689	31505	34885	4.54	31505
-	+	-	-	-	-	19,210	23,567	75,566	39448	31355	4.50	39448
-	-	+	+	+	+	-110,046	-101,981	-5,736	-72588	58036	4.76	-72588
-	-	+	+	+	-	-108,848	-100,840	-5,281	-71656	57622	4.76	-71656
-	-	+	+	-	+	-27,222	-23,132	25,678	-8225	29432	4.47	-8225
-	-	+	+	-	-	-26,023	-21,991	26,132	-7294	29018	4.46	-7294
-	-	+	-	+	+	-66,046	-57,981	38,264	-28588	58036	4.76	-28588
-	-	+	-	+	-	-64,848	-56,840	38,719	-27656	57622	4.76	-27656
-	-	+	-	-	+	16,778	20,868	69,678	35775	29432	4.47	35775
-	-	+	-	-	-	17,977	22,009	70,132	36706	29018	4.46	36706
-	-	-	+	+	+	-75,580	-66,866	37,131	-35105	62710	4.80	-35105
-	-	-	+	+	-	-65,359	-57,135	41,008	-27162	59180	4.77	-27162
-	-	-	+	-	+	4,989	9,836	67,689	27505	34885	4.54	27505
-	-	-	+	-	-	15,210	19,567	71,566	35448	31355	4.50	35448
-	-	-	-	+	+	-47,580	-38,866	65,131	-7105	62710	4.80	-7105
-	-	-	-	+	-	-37,359	-29,135	69,008	838	59180	4.77	838
-	-	-	-	-	+	32,989	37,836	95,689	55505	34885	4.54	55505
-	-	-	-	-	-	43,210	47,567	99,566	63448	31355	4.50	63448

REFERENCES

Abdou, G. and Tereshkovich, W. (2001). “Optimal Operating Parameters in High Speed Milling Operation for Aluminum,” *International Journal of Production Research* Vol. 39, No. 10, pp. 2197-2214.

Abdou, G. H. (1989). “Effects of Operating Rates on the Performance of Manufacturing Systems: A Case Study,” *Computers in Industry,* 13, pp. 123-134.

Abdou, G. and Dutta, S. P. (1990). “An Integrated Approach to Facilities Layout using Expert System,” *International Journal of Production Research*, Vol. 28, No. 4, pp. 685-708.

Adamopouloas, G. (1999). “Job Sequencing with Uncertain and Controllable Processing Times,” *Information Transactions in Operation Research,*. 6, pp. 483-493.

Alfredesson, P. and Verrijdt, J. (1999). “Modeling Emergency Supply Flexibility in a Two-Echelon Inventory System,” *Management Science,* Vol. 45, No.10, pp. 1416-1431.

Andreas, C. (1999). “Electronic Commerce and the Banking Industry: The Requirement and Opportunities for New Payment Systems Using the Internet,” *Journal of Computer-Mediated Communication,* Vol. 1, pp. 1-17.

Ansari, A., Essegaeir, S. and Kohli, R. (2000). “Internet Recommendation Systems,” *Journal of Marketing Research*, Vol. 37, pp. 363-375.

Ashrafi, N., Berman, O. and Cutler, M. (1994). “Optimal Design of Software-Systems Using N-Version Programming,” *IEEE Transactions on Reliability*, Vol. 43, No. 2, pp. 344-350.

Bank, R., Gordon, D. and Slaughter, A. (1998). “Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study, ” *Management Science,* Vol. 44, pp. 433-450.

Bock, G. (1999). “Getting Started with an E-Business Strategy.” *E-Business Strategies & Solutions*, pp. 10-12.

Buzby, B. (1999). "Cyclical Schedules for One-warehouse, Multi-Retailer Systems with Dynamic Demands," *Journal of the Operational Research Society*, 50, pp. 850-856.

Cachon, G. (1999). "Managing Supply Chain Demand Variability with Scheduled Ordering Policies," *Management Science*, Vol. 45, No. 6, pp. 843-856

Cavano, J.P., and McCall J.A. (1978). "A Framework for the Measurement of Software Quality," *Production ACM Software Quality Assurance Workshop*, pp. 133-139.

Chen, F., Drezher, Z., Ryan, J. K. and Simchi-Levi, D. (2000). "Quantifying the Bullwhip Effect in a Simple Supply Chain: The Impact of Forecasting, Lead Times and Information," *Management Science*, Vol. 46, No. 3, pp. 436-443.

Crosby, P. (1979), *Quality is Free*, McGraw-Hill.

Dahan, E. and Srinivasan, V. (2000). "The Predictive Power of Internet-Based Product Concept Testing Using Visual Depiction and Animation," *Journal of Product Innovation Management*, 17, pp. 99-109.

Dahlstrom, R. and Nygaard (1999). "An Empirical Investigation of Ex Post Transaction Costs in Franchised Distribution Channel," *Journal of Marketing Research,* Vol. 37, pp. 160-170.

DeVor, R., Graves, R. and Milles, J.J. (1997). "Agile Manufacturing Research: Accomplishments and Opportunities," *IIE Transactions,* 10, pp. 813-823.

Elsayed, A. E. and Thomas, O. B. (1994). *Analysis and Control of Production System*, Second Edition, Prentice Hall.

Goel, A.L. and Okumoto, K. (1979). "Time-dependent error detection rate model for software and other performance measures," IEEE Trans. Reliability, Vol. R-28.

Grady, R.B. and Caswell, D.L. (1987). *Software Metrics: Establishing a Company-Wide Program*, Prentice-Hall.

Halstead, M. (1977). *Elements of Software Science*, North Holland.

Hansen, D. & Mowen, M. (1997). *Cost Management*, South–Western College Publishing.

Hoffman, D. L., Novak, T. and Chatterjee, P. (2000). "Commercial Scenarios for the Web: Opportunities and Challenges," *Journal of Computer-Mediated Communication,* Vol. 1, No. 3, pp. 1-17.

Hoque, A. Y. and Lohse, G. L. (1999). "An Information Search Cost Perspective for Designing Interface For Electronic Commerce," *Journal of Marketing Research*, Vol. 36, pp. 387-394.

Hou, R., Kuo, S. and Chang, Y. (1997). "Optimal Release Times for Software Systems with Scheduled Delivery Time Based on the HGDM," *IEEE Transaction On Computers,* Vol. 46, No. 2, pp. 216-220.

Huang, E., Cheng, F.T. and Yang, H.C. (1999). "Development of a Collaborative and Event-Driven Supply Chain Information System Using Mobile Object Technology," in *Proc. 1999 IEEE Int. Conf. on Robotics and Automation,* Detroit, Michigan, U.S.A., pp. 1776-1781.

Iannino, A., et al. (1984). "Criteria for Software Reliability Model Comparisons," *IEEE Transaction On Software Engineering*, Vol. SE-10, No. 6, 1984, pp. 687-691.

Iyer, S. and Nagi, R. (1997). "Automated Retrieval and Ranking of Similar Parts in Agile Manufacturing," *IIE Transaction*, 10, pp. 859-876.

Jahanian, F., and Mok, A.K. (1986). "Safety Analysis of Timing Properties of Real-Time Systems," *IEEE Trans. Software Engineering*, Vol. SE-12, No. 9, pp. 890-904.

Jo, A., Sterling, L.S., Quinn, R.D., Busey, G.C. and Kim, Y. (1997). "An Agile Manufacturing Workcell Design," *IIE Transactions*, 10, pp. 901-909.

Kapur, K.C. and Lamberson, L.R. (1997). *Reliability in Engineering Design*, John Wiley & Sons.

Khoshgoftaar, T. M. and Allen, E. (2000). "A Practical Classification-Rule for Software-Quality Models," *IEEE Transaction On* Reliability, Vol. 49, No. 2, pp. 209-216.

Khoshgoftaar, T., Ellen, E., Jones, W.D. and Hudepohl, J.P. (2000)."Classification-Tree Models of Software-Quality Over Multiple Releases," IEEE Trans on Reliability, Vol. 49, No. 1, pp.4-11.

Knight, J.C., and Ammenn, P.E. (1985). "An Experimental Evaluation of Simple Methods for Seeding Program Errors," *Procedure 8th International Conference of Software Engineering*, IEEE, London, pp. 337-342.

Lee, H., So, K. and Tang, C. (2000). "The Value of Information Sharing in a Two-Level Supply Chain," *Management Science*, Vol. 46, No. 5, pp. 626-643.

Lee, H. A. (2000). "Heuristic for Scheduling on Nonidentical Machine to Minimize Tardy Jobs," *Intonation Journal of Industrial Engineering*, 7(3), pp. 188-194.

Leveson, N.G. (1986). "Software Safety: Why, What, and How," *ACM Computing Surveys*, Vol. 18, No. 2, pp. 125-163.

Leveson, N.G., and Stolzy, J.L. (1987). "Safety Analysis using petri Nets," *IEEE Trans. Software Engineering*, Vol. SE-13, No.3, pp. 386-397.

Lin, H. and Chen, K. (1993). "Nonhomogeneous Poisson Process Software-Debugging Models With Linear Dependence," *IEEE Transaction on Reliability*, Vol. 42, No. 4, pp. 613-617.

MacCormack, A., Verganti, R., and Iansiti, B.(2001). "Developing Products on Internet Time: The Anatomy of a Flexible Development Process," *Management Science*, Vol. 47, No. 1, pp. 133-150.

Maghsoodllo, S., Brown, D.B. and Lin, C. (1992). "A Reliability & Cost Analysis of an Automatic Prototype Generator Test Paradigm," *IEEE Transactions on Reliability*, Vol. 41, No. 4, pp. 547-553.

Magrab, E. (2001). *An Engineer's Guide to Matlab*, Prentice Hall.

McCall, J. P. and Walters, G. (1977). "Factors in Software Quality," three volumes, NTIS AD-A049-014, 015, 055.

McCutcheon, D. and Staurt, F. (2000). "Issues in the Choices of Supplier Alliance Partners," *Journal of Operations Management*, 18, pp. 279-301.

Merat, F., Sargent, D.M., Barendt, N.A., Velasco, V.B. and Jr. Podgurski, A. (1999). "Editors Business and the Internet," *IEEE Engineering Management Review*, Winter, pp. 6-24.

Michael, P. and Chao, X. (1999). *Operations Scheduling With Application in Manufacturing and Service*, Irwin McGraw-Hill.

Mills, H.D. (1972). "On the Statistical Validation of Computer Programs," FSC72: 6015, IBM Federal Systems Division.

Miller, D.R. and Sofer, A. (1985). "Completely Monotone Regression Estimates for Software Failure Rates," *Procedure 8th International Conference of Software Engineering, IEEE*, London, August 1985, pp. 343-348.

Moranda, P. (1981). "Error Detection Model for Application during Software Development," *IEEE Transaction On Reliability*, Vol. R-30, No. 4, pp. 309-312.

Munson, J. and Khoshgoftaar, T. M. (1992). "The Detection of Fault-Prone Programs," *IEEE Trans. On Software Engineering*, Vol. 18, No. 5, pp. 423-432.

Musa, J.D., Iannino, A. and Okumoto, K. (1987) *Engineering and Managing Software with Reliability Measures*, McGraw-Hill.

Nordmann, L. and Pham, H. (1999). "Weighted Voting System," *IEEE Transaction on Reliability*, Vol. 48, No. 1, pp. 42-49.

Noushin, A. and Berman, O. (1992). "Optimization Models for Selection of Programs, Considering Cost & Reliability," *IEEE Transaction on Reliability*. Vol. 41, No. 2, pp. 281-287.

Nouwan, J. and Harry, B. (2000). "Living Apart Together in Electronic Commerce: The Use of Information and Communication Technology to Create Network Organization," *Journal of Computer-Mediated Communication,* Vol. 1, No. 3, pp. 1-12.

Ohtera, H. (1999). "Optimum Software-Release Time Considering an Error-Detection Phenomenon during Operation," *IEEE Trans*action Vol. 39, No. 5, pp. 596-599.

Paulk, M. C., Curtis, B. Chissis, M.B. and Weber, C.V. (1993). "Capability Maturity Model, Version 1.1," *IEEE Software*, Vol. 10, No. 4, pp. 18-27.

Pham, H., Nordman, L. and Zhang, X. (1999). "A General Imperfect-Software-Debugging Model with S-Shaped Fault-Detection Rate," *IEEE Transactions on Reliability*, Vol. 48, No. 2, pp. 169-175.

Pham, H. and Zhang, X. (1999). " Software Release Policies with Gain in Reliability Justifying the Costs," Annals of Software Engineering 8, pp. 147-166.

Picot, A., Bortenlaenger, C. and Roehrl, H. (2000). "The Automation of Capital Markets," *Journal of Computer-Mediated Communication,* Vol. 1, No. 3, pp. 1-20.

Ross, S. M. (1993). *Probability Model*, Fifth Edition Academic Press, Inc.

Sahin, I. and Zahhedi, F. (1999). "Control Limit Policies for Warranty, Maintenance and Upgrade of Software System," *School of Business Administration University of Wisconsin-Milwaukee*, pp. 1-31.

Sarkar, M. B., Butler, B. and Steinfield, C. (2000). "Intermediaries and Cybermediaries: A Continuing Role for Mediating Players in the Electronic Marketplace," *Journal of Computer-Mediated Communication,* Vol. 1, No. 3, pp. 1-15.

Savsar, M. and Allahverdi, A. (1999). "Algorithm for Scheduling Jobs on Two Serials Duplicate Stations," *Internal Transaction of Operation Reserach.* 6, pp. 411-422.

Schneidewind, N. F. (1997). "Reliability Modeling for Safety-Critical Software," *IEEE Trans. On Reliability*, Vol. 46, No. 1, pp. 88-98.

Schulmeyer, G.C., and McManus, J.I. (1987). *Handbook of Software Quality Assurance*, Van Nostrand Reinhold.

Seybold, P., Marshak, D. and Patricia, S. (2000). "Group's E-commerce Service Evolves into E-Business Strategies & Solution," *E-Business Strategies & Solution/Opinion*.

Shin, H., Collier, D.A. and Wilson, D. (2000). "Supply Management Orientation and Supplier/Buyer Performance," *Journal of Operation Management*, 18, pp. 317-333.

Shooman, M. (1983). *Software Engineering*, McGraw-Hill.

Smids, C., Stutzke, M. and Stoddard, R. W. (1998). "Software Reliability Modeling: An Approach to Early Reliability Prediction," *IEEE Transactions on Reliability*, Vol. 47, No. 3, pp. 268-278.

Software Engineering Standards, 3d ed., IEEE, 1989.

Song, L. and Nagi, R. (1997). "Design and Implementation of a Virtue Information System for Agile Manufacturing," *IIE Transaction*, 10, pp. 839-857.

Steinfield, C., Kraut, R. and Plummer, A. (2000). "The Impact of Interorganization Networks on Buyer-Seller Relationship," *Journal of Computer-Mediated Communication*, Vol. 1, No. 3, pp. 1-13.

Sueyoshi, T. (1999). "DEA Duality on Returns to Scale (RTS) in Production and Cost Analyses: An Occurrence of Multiple Solutions and Differences Between Production-Based and Cost-Based RTS Estimations," *Management Science*, Vol. 45, No. 11, pp. 1593-1608.

Sun, H.W., Zhou, M.C. and Wolf, C. (2001),"A Methodology for Software Development Cost Analysis in Information-Based Manufacturing", 2001 IEEE International Conference on Robotics and Automation. Seoul, Korea

Trachtenberg, M. (1990). "A General Theory of Software-Reliability Modeling," *IEEE Trans. On Reliability*, Vol. 39, No. 1, pp. 92-96.

Tropicana Jersey City Information Technology. January 3, 2000, pp. 1-28.

Tsay, A. (1999). "The Quantity Flexibility Contract and Supplier-Customer Incentives," *Management Science*, Vol. 45, No. 10, pp. 1339-1358.

Upadhyaya, S. (1993). "Analysis of Noncherent Systems and an Architecture for the Computation of the System Reliability," *IEEE Transaction on Computer*, Vol. 42, No. 4, pp. 483-493.

Venkatraman, N. and Henderson, J.C. (1999). "Real Strategies for Virtual Organizing," *IEEE Engineering Management Review*, Winter, pp. 26-40.

Veseley, W.E., et al. (1981). *Fault Tree Handbook*, NUREG-0492, U.S. Nuclear Regulatory Commission.

Whitney, D. (1995). "Agile pathfinders in the Aircraft and Automobile Industrial," *A Process Report MIT.*

Wigand, R. T. and Benjamin, R. I. (2000). "Electronic Commerce: Effects on Electronic Markets," *Journal of Computer-Mediated Communication,* Vol. 1, No. 3, pp. 1-10.

Wu, N., Mao, N. and Qian, Y. (1999). "An Approach to Partner Selection in Agile Manufacture," *Journal of Intelligent Manufacturing*, 10, pp. 519-529.

Yamada, S. (1998). "Quantitative Assessment Models for Software Safety/Reliability," *Electronics and Communications in Japan,* Part 2, Vol. 81(5), pp. 215-222.

Yamada, S., and Ohba, M. and Osaki, S. (1983). "S-Shaped Reliability Growth Modeling for Software Error Detection," *IEEE Trans*action Vol. R-32, No. 5, pp. 475-478.

Yamada, S. and Osaki, S. (1985). "Discrete Software Reliability Growth Models," *Applied Stochastic Models and Data Analysis*, Vol. 1, 1, pp. 65-77.

Zhang, X. and H. Pham,(1998). " A Software Cost Model with Error Removal times and Risk Costs," *International Journal on System Science,* Vol. 29(4), pp134-145.