NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

RISK ASSESSMENT IN INCREMENTAL SOFTWARE DEVELOPMENT

by

Eric K. Matsuo

December 1999

Thesis Advisor:	Luqi
Second Reader:	Nogueira

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 1999	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE RISK ASSESSMENT IN INCREMENTAL SOFTWARE DEVELOPMENT	**5. FUNDING NUMBERS** MIPR9MNPSAR071
6. AUTHOR(S) Matsuo, Eric K.	
7. PERFORMING ORGANIZATION NAME (S) AND ADDRESS (ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME (S) AND ADDRESS (ES) Army Research Office (ARO)	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

Cost overruns, schedule slips, and projects with fewer features or functions than originally specified are some of the difficulties that the software community faces in almost all software projects. The application of proper risk management throughout the lifecycle of the software development can drastically improve the chances of success. Risk management is an essential skill that many good mangers possess. Utilizing proper risk management provides early risk detection, which in turn gives the manager more flexibility to mitigate and resolve the risks within the software development project.

This thesis presents a disciplined and systematic risk management tool that can be utilized to assess risk in incremental software development projects from cradle to grave. This methodology can be applied with limited resources, and is adaptable and flexible enough to be used on all software intensive projects. The methodology incorporates the Software Engineering Institute's proven risk taxonomy and questionnaire. It also provides a project manager or project decision-maker an efficient way of assessing risk in incremental software development. Further, this thesis implements the risk assessment framework on a software development project and validates the validity and usefulness as a risk management tool.

14. SUBJECT TERMS Software Engineering, Risk Management, Systematic Risk Assessment Tool	**15. NUMBER OF PAGES** 114
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFI-CATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 **Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18**

THIS PAGE INTENTIONALLY LEFT BLANK

RISK ASSESSMENT IN INCREMENTAL SOFTWARE DEVELOPMENT

Eric K. Matsuo
B.S., California State Polytechnic University, Pomona, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1999

Author: Eric K. Matsuo
Eric K. Matsuo

Approved by: [illegible]
Luqi, Thesis Advisor

[illegible]
Nogueira, Second Reader

[illegible]
Luqi, Chairman
Software Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Cost overruns, schedule slips, and projects with fewer features or functions than originally specified are some of the difficulties that the software community faces in almost all software projects. The application of proper risk management throughout the lifecycle of the software development can drastically improve the chances of success. Risk management is an essential skill that many good mangers possess. Utilizing proper risk management provides early risk detection, which in turn gives the manager more flexibility to mitigate and resolve the risks within the software development project.

This thesis presents a disciplined and systematic risk management tool that can be utilized to assess risk in incremental software development projects from cradle to grave. This methodology can be applied with limited resources, and is adaptable and flexible enough to be used on all software intensive projects. The methodology incorporates the Software Engineering Institute's proven risk taxonomy and questionnaire. It also provides a project manager or project decision-maker an efficient way of assessing risk in incremental software development. Further, this thesis implements the risk assessment framework on a software development project and validates the validity and usefulness as a risk management tool.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

The author would like to acknowledge the financial support of SPAWAR, Code D32 and the Executive Board, for approving the funding to allow me to attend the Naval Postgraduate School and complete this thesis.

The author would like to especially thank Professor Luqi and MSc Juan Carlos Nogueira for their guidance with this thesis. I greatly appreciate the gracious patience they maintained while I was working on this project.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

With technology growing at exponential rates and software applications exploding with complex functionality, the software development process has become extremely complicated and difficult to manage. In 1994, a survey of over eighty-three hundred commercial software-intensive projects found more than half either over budget, behind schedule, or had fewer features or functions than originally specified [Ref. 1]. In 1995, over a quarter of a trillion dollars was spent on over one hundred seventy-five thousand software projects throughout the United States [Ref. 2]. A quarter of that was spent on cost overruns and another third of that was spent on canceled projects [Ref. 2]. These results may have been attributed to lack of risk management.

Although applying risk management to a software development project will not act as 'a silver bullet' [Ref. 3], it will help resolve and mitigate many problems that managers encounter. The Department of Defense has recognized the importance of proper risk management in software development projects. Listed below are two directives that show the emphasis on risk management.

Department of Defense Directive 5000.1 states in Section D.1.d:

> *Program managers and other acquisition managers shall continually assess program risks. Risks must be well understood, and risk management approaches developed, before decision authorities can authorize a program to proceed into the next phase of the acquisition process* [Ref. 4].

The military standard EIA/IEEE Standard-016 states in Section 5.19.1:

> *The developer shall perform risk management throughout the software development process. The developer shall identify, analyze, and prioritize the areas of the software development project that involve potential technical, cost, or schedule risk; develop strategies for managing those risks; record the risks and strategies in the software development plan; and implement the strategies in accordance with the plan* [Ref. 5].

Emphasizing risk policies within an organization is only the initial step to relieving some of the problems faced by the software community. This thesis describes a disciplined and systematic method to assess risk in any software development project that can be utilized by all project managers to implement these risk policies.

The thesis consists of three key parts that include summaries of the references, a detailed risk assessment framework and the implementation of the framework on a software development project. Prior to the discussion of the thesis, it is important that a definition of risk be presented. Risk is a combination of the probability of occurrence and the severity of the consequence if it occurs. The consequence and probability must be looked at equally when assessing risk. The other factor that must be addressed while analyzing the risk is the timeframe that the risk may occur. There is a direct correlation between the timeframe of occurrence and the severity of the risk. The longer the timeframe, the less severe the risk will be.

The first part contains summaries of twenty-four sources from which this thesis foundation was built on top of. Many of the sources were affiliated with the Software Engineering Institute at Carnegie Mellon University or the Institute of Electrical and Electronics Engineering Computer Society. Both institutes play a key role in helping organizations produce better quality products for the software community. In fact, the SEI established nearly twenty years ago by the Department of Defense to provide leadership in advancing the practice of software engineering, which in turn, has improved the quality of systems that depend on software. The IEEE Computer Society was founded more than fifty years ago, and is currently a leading provider of technical information and services to all computing professionals.

The second part describes the risk assessment framework in detail. The framework consists of an identification and analysis phase. The identification phase utilizes the Software Engineering Institutes Risk-Taxonomy Based Questionnaire. It is used to locate and identify potential risks within a software development project. The analysis phase describes a method to systematically prioritize identified risks. This is accomplished by quantifying the risk criteria to attain a risk factor for each of the identified risks. The end result provides the project manager or decision-maker a list of prioritized risks.

In the third part, the risk assessment framework is implemented on two software development projects currently under development at the Space and Naval Warfare Systems Center, San Diego. The first project's primary goal is to re-host an application from the UNIX environment to the Windows NT environment. The application consists of over two hundred thousand lines of C source code, and utilizes many X11 and motif calls. The second project used to test the validity of the risk assessment framework is a Mapping, Charting, Geodesy and Imagery Server that will be utilized by over one hundred and fifty surface ship is in the US Navy.

Overall, this thesis describes the development of a disciplined and systematic methodology to assess risks within a software development project, and is flexible and comprehensive enough to be applied to any software development project. Another positive characteristic of this framework is that it can be utilized during any stage of the development life cycle. This methodology provides the software community a vital foundation tool to apply proper risk management, and when utilized, shall dramatically improve the probability of a project overall success.

THIS PAGE INTENTIONALLY LEFT BLANK

II. RISK ASSESSMENT FOUNDATION

The foundation of the risk assessment methodology was based on many sources. A majority of the sources had affiliations to the Software Engineering Institute at Carnegie Mellon University or the Institute of Electrical and Electronics Engineering Computer Society. Some of the other sources included: Program Manager Magazine; The Journal of Defense Software Engineering; Managing Risk and Methods for Software Systems Development.

There were four key common issues that a majority of the sources pointed out. First, many of the authors felt risk management is one of the key areas the project managers fail to focus enough attention to. Applying proper risk management increases the chances of success. Second, there is a lack of risk assessment methodology models that program managers can utilize. The risk assessment methodology presented in this thesis provides just that. Third, early and continuous detection of risks throughout the life cycle is a key element to successful risk management. Finally, open communications must be attained within a software organization to effectively handle and mitigate risks.

A. SOFTWARE ENGINEERING INSTITUTE

In December 1984, the Software Engineering Institute was established by the United States Department of Defense to provide leadership in advancing the state of the practice of software engineering. Five years after the foundation, SEI began to formally study and develop risk management concepts to improve the success of software development projects. Summarized below are seven papers and reports affiliated with the SEI that were used to lay a strong foundation the risk assessment methodology.

1. Car, M. J.; Konda, S. L.; Monarch, I.; Ulrich, F. C.; Walker, C. A. *Taxonomy-Based Risk Identification* (CMU/SEI –93-TR-6). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1993.

This report validates the validity of the Software Engineering Institute's taxonomy-based questionnaire, and describes how to properly apply it to software development projects. The questionnaire, consisting of a little less than two hundred questions, provides a systematic methodology to identify risks within a software development project.

2. Gallagher, B. P.; Alberts, C. J.; Barbour, R. E. *Software Acquisition Risk Management Key Process Area (KPA) A Guidebook Version 1.0* (CMU/SEI – 97-HB-002). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1997.

This guidebook describes the Acquisition Risk Management (ARM) Key Process Area (KPA) of the Software Acquisition Capability Maturity Model (SA-CMM) in explicit detail. In summary, ARM is a two phases. The first step is to identify risks early in the lifecycle and incorporate the risks into the software acquisition plan. The second phase of the cycle is to continuously monitor for new risks throughout the life cycle until completion, and continuously integrate the risk mitigation plan into the software acquisition process. The guidebook was very well structured and extremely informative.

3. Gluch, D. P.; Dorofee A.J.; Hubbard, E. A.; Travalent J. J. *A Collaboration in Implementing Team Risk Management* (CMU/SEI –95-TR-016). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1996.

This report describes the implementation of the Software Engineering Institute's Team Risk Management process. The process was conducted on a team consisting of the SEI, government program office and a private contractor. The results of the study determined that the success of team risk management was dependant upon several key factors. These factors included good communication, a strong commitment to create a team culture, and the continual application of the risk model throughout the life cycle of the software development project.

4. Higuera, R. P., *Software Risk Management* (CMU/SEI –96-TR-012). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1996.

This paper presents a broad overview of the Software Engineering Institute's software risk management methodologies. This overview includes the 'big picture' approach on how the SEI plans to resolve and mitigate the risk incurred in software development projects. This paper provides a good introduction in the subject of risk management.

5. Higuera, R. P.; Gluch, D. P.; Dorofee A.J.; Murphy R.L.; Walker J. A.; Williams R.C. *An Introduction to Team Risk Management* (CMU/SEI –94-SR-1). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1994.

This paper effectively describes the Software Engineering Institute's Team Risk Management approach. The nine key ingredients to the success of the team risk management include: open communication; shared product vision; forward-looking search for uncertainties; value of individual perception; systems perspective; integration into program management; proactive strategies; systematic and adaptable methodology; and routine and continuous processes.

6. Higuera, R. P.; Dorofee A.J.; Walker J. A.; Williams R.C. *Team Risk Management: A New Model for Customer Supplier Relationships* (CMU/SEI –94-SR-5). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1994.

This paper provides a nice overview of the Software Engineering Institute's Team Risk Management approach. The team approach was based upon the philosophy of cooperative teams and the principles of risk management. The advantages of team risk management over individual risk management include the following: improved communications; multiple perspectives on risk; broader base of expertise; broader based buy in; and risk consolidation.

7. Sisti, F.J.; Joseph, S. *Software Risk Evaluation Method* (CMU/SEI –94-TR-019). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1994.

This report provides a comprehensive description of the Software Engineering Institute's Software Risk Evaluation Method. This methodology provides the software community an effective means to identify, analyze and mitigate risk within a software development project.

B. INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (IEEE) COMPUTER SOCIETY

The Institute of Electrical and Electronics Engineers Computer Society was founded in 1946 and is currently one hundred thousand members strong. The Computer Society's vision is to be the leading provider of technical information and services to the world's computing professionals. Summarized below are twelve papers and reports affiliated with the IEEE Computer Society that were used to lay a strong foundation the risk assessment methodology.

1. Boehm, B.W.; DeMarco T. *Software Risk Management.* IEEE Software. pp. 17-19, May/June 1997.

This article provides an overview of risk management. An interesting point Boehm and DeMarco bring up is that the software community often sees risk with such negative connotations and defeatism attitudes that some project managers deliberately ignore risks within a software development project in order to project a confident, "can-do" attitude. This is one of the problems that project managers have to face when facing risk management issues.

2. Boehm, B.W. *Software Risk Management: Principles and Practices.* IEEE Software. pp. 32-41, January 1991.

This article describes six areas that fall under two categories of risk management that can be used to help achieve successful risk management. The two categorizes are risk assessment and risk control. The risk assessment category includes risk identification, risk analysis and risk prioritization. The risk control category consists of risk-management planning, risk resolution and risk implementation. Boehm elaborates on each of these areas. He also includes a satellite experiment project as an example of how each of these areas are implemented.

3. Charette, R. N.; Adams, K.M; White, M. B. *Managing Risk in Software Maintenance.* IEEE Software. pp. 43-50, May/June 1997.

This article discusses the risk management involved with software maintenance. Maintaining existing systems generally provides more opportunities for risk with less freedom to mitigate them. The authors describe how the US Navy Maintenance Support Office handles the risk management in this area.

4. Conrow, E. H.; Shishido, P. S. *Implementing Risk Management in Software Intensive Projects.* IEEE Software. pp. 83-89, May/June 1997.

This article identifies four area of risk management that must occur for to achieve good risk management. The four areas include risk planning, risk assessment, risk handling and risk monitoring. Conrow and Shishido also point out several key risk issues that they aggregate into five categories including project level, project attributes, management, engineering and work environment.

5. Fairley, R. *Risk Management for Software Projects.* IEEE Software. pp. 57-67, May 1994.

This article describes a seven-step procedure to identify risk factors, calculate their probability and effect on a project, and plan for and conducts risk management. The seven steps are to: identify the risk factors; assess risk probabilities and effects on the project; develop strategies to mitigate identified risks; monitor risk factors; invoke a contingency plan; manage the crisis and recover from a crisis.

6. Garvey P. R.; Phair D. J.; Wilson J. A. *An Information Architecture for Risk Assessment and Management.* IEEE Software. pp. 25-34, May/June 1997.

This article discusses the Risk Assessment and Management Program (RAMP). RAMP is a risk management system that provides interactive support for identifying, analyzing and sharing risk mitigation experience. This program was created do to a need of project managers to share experiences on risk management on to learn from fellow colleagues.

7. Kansala, K. *Integrating Risk Assessment with Cost Estimation.* IEEE Software. pp. 61-68, May/June 1997.

In this article, the author describes a method and tool to help calculate project risk contingencies based upon project history and questionnaires. The risk management activities supported by the Risk Method include risk identification, risk analysis, risk prioritization, risk management planning, risk resolution and risk monitoring.

8. Lister T.; Carr, M.J. *Point / Counterpoint: Is Risk Management Risky?* IEEE Software. pp. 20-24, May/June 1997.

This brief article suggests that two activities are required into order to achieve proper risk management. The first is to make continual informed decisions about risk now and prepare for future risks. The second is to take appropriate actions to mitigate current and future risks.

9. Kitchenham, B.; Linkman, S. *Estimates, Uncertainty, and Risk.* IEEE Software. pp. 69-74, May/June 1997.

This article includes methods on how to estimate project completion time considering risk, errors and uncertainties. They discuss in detail measurement error, model error, assumption error, and scope error.

10. Madachy, R. J. *Heuristic Risk Assessment Using Cost Factors.* IEEE Software. pp. 51-60, May/June 1997.

In this article, the author describes a method to enhance the user's ability to rank associated sources of project risks. To accomplish this, the author uses an extension of the Constructive Cost Model (COCOMO).

11. Moynihan, T. *How Experienced Project Managers Assess Risk.* IEEE Software. pp. 35-42, May/June 1997.

This article describes the experiences of 14 software project managers, and how these managers assess risk. The author focuses on three areas. These areas include; which characteristics of the customer, the application, and so on do project managers consider important when planning new development projects for new clients; how these characteristics relate to accepted software project risks; do most software managers characterize new projects in generally the same way.

12. Williams, R. C.; Walker J. A.; Dorofee, A. J. *Putting Risk Management into Practice.* IEEE Software. pp. 75-82, May/June 1997.

In this article, the authors describes lessons learned and experiences gained by working in the Software Engineering Institute's Risk Program with the Department of Defense programs, at the customer and supplier levels. Some of the key area covered in the article include identify new risks, evaluating new risks, classifying new risks, prioritizing new risks, plan risk mitigation, track risks and mitigation plans and review and modifying mitigation plans.

C. OTHER SOURCES OF INFORMATION

Several other sources were used to gather information for the risk assessment methodology. These sources are as follows: Program Manager Magazine; Crosstalk: The Journal of Defense Software Engineering; *Managing Risk: Methods for Software Systems Development*; and Communications of the ACH. Summarized below are five books and articles used to lay a strong foundation the risk assessment methodology.

1. Conrow, E. H.; Fredrickson, M. A. *Some Considerations for Implementing Risk Management in Defense Programs*. Program Manager. pp. 6-11, January-February 1996.

This article provides guidance for project managers to enhance risk management in their organizations. The authors mention four risk management deficiencies along with a detailed discussion about risk management implementation on a Department of Defense program.

2. Dorofee, A. J.; Walker, J. A; Williams, R. C. *Risk Management In Practice.* Crosstalk: The Journal of Defense Software Engineering. pp. 8-12, April 1997.

This article draws information from the research, development, and testing of risk management practices of over 50 client programs of the Software Engineering Institute over the past 7 years. This article focuses on the practice of risk management and the transition or establishment of risk management in a project or organization.

3. Hall, E.M. *Managing Risk: Methods for Software Systems Development.* Massachusetts, Addison-Wesley, 1998.

This comprehensive risk management book is divided into five parts. Part I, "Risk Management Discovery", provides an introduction to the role of risk management in software engineering. Part II, "Risk Management Process", as the title suggests, describes a standardized process to handle risk in a project. The process consists of a 5-step plan that including the following: identify risk; analyze risk; plan risk; track risk; and resolve risk. Part III, Risk Management Infrastructure", describes methods to increase risk awareness in the organization. Part IV; "Risk Management Implementation", outlines an approach to following in order to successfully manage risk in a software project. Part V, "People in Crisis and Control", describes actual experiences with risk management in industry.

4. Keil M.; Cule, .P. E.; Lyytinen, K.; Schmidt, R. C. *A Framework for Identifying Software Project Risks*. Communications of the ACH. Vol. 41 No. 11 pp. 76-83, November 1998.

This article draws information from 41 experienced software project managers residing in US, Finland and Hong Kong. The article describes a universal set of risk factors. The top three risk factors were as follows: lack of top management commitment to the project; a failure to gain user commitment; and misunderstanding the requirements.

5. Statz, J.; Oxley, D.; O'Toole, P. *Identifying and Managing Risks for Software Process Improvement*. Crosstalk: The Journal of Defense Software Engineering. pp. 8-12, April 1997.

This article describes methods to identify and manage risks at key points in a software process improvement program. The article elaborates on several risk factor categories including budget and cost, content of deliverables, culture, maintenance of SPI deliverables, and mission and goals.

THIS PAGE INTENTIONALLY LEFT BLANK

III. RISK ASSESSMENT FRAMEWORK

The framework used to assess risk in incremental software development consists of a two-phase process. The first phase describes a process to systematically identify risks within a software development project. The second describes a disciplined methodology that can be used to analyze the risks. Due to the nature of risks appearing rapidly and unexpectedly, the risk assessment methodology should be applied continuously at each stage of the software development cycle. The result of the risk assessment provides the project manager or project decision-maker with a prioritized list of risks in the software project.

A. IDENTIFY RISK

Identifying risks is the key to proper risk management. The sooner risks are identified, the better off the project manager or decision-maker will be to mitigate and resolve risks. Described below is an effective methodology to find and document risks within a software development project.

1. Identification of Risks

The first step to identifying risks involves using the Software Engineering Institute's risk-based questionnaire. The questionnaire provides a systematical tool to identify risks within an organization. The questionnaire is consists of a little less than two hundred questions and can be applied to any sized software development project. To meet the needs of all software development projects, the Software Engineering Institute's questionnaire was very thorough covering all aspects of the software development project. Therefore, some of the questions may not pertain to every project, and should be modified accordingly to meet the needs of the software project.

The Software Engineering Institute's questionnaire follows the risk taxonomy structure described in Table1. Software Engineering Institute's Risk Taxonomy [Ref. 6].

Software Engineering Institute's Risk Taxonomy		
A. Product Engineering	B. Development Environment	C. Program Constraints
1. Requirements	1. Development Process	1. Resources
a. Stability	a. Formality	a. Schedule
b. Completeness	b. Suitability	b. Staff
c. Clarity	c. Process Control	c. Budget
d. Validity	d. Familiarity	d. Facilities
e. Feasibility	e. Product Control	2. Contract
f. Precedent	2. Development System	a. Type of Contract
g. Scale	a. Capacity	b. Restrictions
2. Design	b. Suitability	c. Dependencies
a. Functionality	c. Usability	3. Program Interfaces
b. Difficulty	d. Familiarity	a. Customer
c. Interfaces	e. Reliability	b. Associate Contractors
d. Performance	f. System Support	c. Subcontractors
e. Testability	g. Deliverability	d. Prime Contractor
f. Hardware Constraints	3. Management Process	e. Corporate Management
g. Non-Developmental	a. Planning	f. Vendors
Software	b. Project Organization	g. Politics
3. Code and Unit Test	c. Management Experience	
a. Feasibility	d. Program Interfaces	
b. Testing	4. Management Methods	
c. Coding/Implementation	a. Monitoring	
4. Integration Testing	b. Personnel Management	
a. Environment	c. Quality Assurance	
b. Product	d. Configuration Management	
c. System	5. Work Environment	
5. Engineering Specialties	a. Quality Attitude	
a. Maintainability	b. Cooperation	
b. Reliability	c. Communication	
c. Safety	d. Morale	
d. Security		
e. Human Factors		
f. Specifications		

Table 1. Software Engineering Institute's Risk Taxonomy [Ref. 6]

The questions are aggregated into three classes. The classes are product engineering, development environment and program constraints. Product engineering covers the technical aspects of the work. It includes the requirements, design, code and unit test, integration and test, and the engineering specialties. The development environment includes the development process, development system, management process and work environment. The program constraints cover the resources, contracts and program interfaces of the software development cycle.

Each class is divided into several elements and each element is divided into several attributes. There are several questions for each attribute to help the administrator of the questionnaire locate potential risks in the software development project.

2. Documentation of Risks

Documenting the risks is the second step in the identification process. It is essential to document all risks in a consistent manner. To do this, a risk management form with the following attributes should be generated. The attributes should include a risk identification tracking number, date of identification, originator, probability of occurrence, severity of occurrence and a description of the risk.

Risk Identification Form		
Tracking Number:	Date of Identification:	Originator:
Description of the Risk:		
Probability of Occurrence:		
Severity of Occurrence:		
Timeframe of Occurrence:		
Additional Comments:		

Table 2. Risk Identification Form

B. ANALYZE RISK

The analysis portion of the risk assessment framework provides the project manager or decision-maker a tool to help prioritize risks. These prioritized risks can enhance the ability of the project manager or decision maker to properly allocate resources to the most severe risks. The severity of each risk is determined based on three criteria, including the probability of occurrence, consequence of the occurrence, and timeframe. This framework provides a methodology to quantify each of the risk criteria so that the prioritized list of risk can be systematically generated.

1. Probability of Occurrence

The probability of occurrence can be quantified using Probability Criteria Table shown below. There is a direct correlation between the probability of occurrence and the quantitative criteria risk value. The higher the probability of occurrence is, the higher the quantitative criteria risk value.

Quantitative Value	Probability	Comments
10	> 90%	**Very likely**
9	80-89%	
8	70-79%	
7	60-69%	**Probable**
6	50-59%	
5	40-49%	
4	30-39%	
3	20-29%	**Unlikely**
2	10-19%	
1	< 9%	

Table 3. Probability of Occurrence

2. Consequence of Occurrence

The consequence of occurrence can be quantified based on the unmet requirements, lost time or the cost incurred by the risk occurring. The table below can be used as a guide, but must be modified to meet the needs of individual software development project.

Quantitative Value	Unmet Requirements / Functionality	Lost Time	Cost Incurred
10	**Severe** *(Application Non-functional)*	**Severe** *(Months)*	**Severe** *(> 10% of project)*
9			
8			
7	**Substantial** *(Application Partially Functional)*	**Substantial** *(Weeks)*	**Substantial** *(3-9%)*
6			
5			
4			
3	**Minimal** *(Work-around Available)*	**Minimal** *(Days)*	**Minimal** *(< 3%)*
2			
1			

Table 4. Consequence of Occurrence

3. Timeframe of Risk

The time frame in which the risk could occur affects the importance that the project manager or decision-maker places on the risk. The table below provides a guide to quantifying the time frame so that each risk may be systematically ranked.

Quantitative Value	Time to Occurrence
10	**Severe** *(Weeks)*
9	
8	
7	**Substantial** *(Months)*
6	
5	
4	
3	**Minimal** *(Quarters)*
2	
1	

Table 5. Risk Time Frame

4. Evaluation of Criteria

Once each of the criteria has been quantified, each criterion should be multiplied together for each risk. The results will reveal the highest risks to the program manager or decision-maker.

Prioritized Risk Summary				
Prioritized Rank of the Risk	Description of the Risk			
	Total *(Quantitative Risk Criteria Value)*	**Probability** *(Quantitative Value)*	**Severity** *(Quantitative Value)*	**Timeframe** *(Quantitative Value)*

Table 6. Prioritized Risk Summary

IV. IMPLEMENTATION ON A SOFTWARE PROJECT

To test the validity of the risk assessment framework, the methodology was implemented on the Display Control Subsystem and the Mapping, Charting, Geodesy, and Imagery Chart Server software project developed at the Space and Naval Warfare Systems Center, San Diego.

A. DISPLAY CONTROL SUBSYSTEM PORT

The Display Control Subsystem software project is in its initial design phase of porting over from a Unix (HP-UX 10.20) environment to a Windows NT (WinNT) environment. The application is currently utilized by many of the US Navy's surface ships. It is over two hundred thousand lines of C, X11, and motif code. Described below is a general description of the task at hand.

Software

Current Status:

- Language: C, X11, Motif
- Compiler: CC –ANSI
- Operating System: HP-UX 10.20

Porting to:

- Language: Visual C++
- Compiler: Microsoft Visual C++ Version 6.0
- Operating System: Microsoft NT 4.0 Workstation

Hardware

Current Status:

- Hardware Platform: HP 770 workstation (RISC CPU)

Porting to:

- Hardware Platform: PC – high end (Intel CPU)

The application has been on US Navy surface ships for more than five years and is currently in its fourth build. The following section describes the risks identified from the framework. Later it describes how the framework is used to systematically analyze these risks.

1. Identified Risks For the Risk-Based Questionnaire

The risk-based questionnaire helped to identify twenty risks. Of the twenty risks, four were removed due to duplication. Please see *Appendix C: Results of he Risk Questionnaire for the Display Control Subsystem* for a detailed version description of the questionnaire process. Described below are the sixteen identified risks at this point in the development process.

1. Microsoft NT Workstation operating system does not support dual login, therefore a background process must be developed to handle dual login.
2. The personnel are lacking expertise in the areas of Win32 APIs, Microsoft NT architecture and Microsoft Foundation Class Libraries.
3. The operating system calls are different. Therefore, the design and algorithms will have to be modified accordingly.
4. Meeting the requirement of having multiple applications running simultaneously sharing control of the Real Time Subsystem will pose a problem.
5. The DCS shall be using a Microsoft operating system.
6. The DCS shall be using a Department of Defense common operating environment. We are dependent on them to produce a reliable operating environment.
7. The DCS shall be using Microsoft's Visual C++ compiler.
8. Memory management will have to be implemented differently using Win 32 APIs.
9. Process handling will have to be implemented differently using Win 32 APIs.

10. Socket handling will have to be implemented differently using Win 32 APIs.
11. Script files will have to be implemented differently.
12. All the graphical user interfaces (X11, Xt, and Motif widgets/gadgets) will have to be rewritten.
13. The current application is written in C, but we will be implementing some functionality in C++ using Microsoft Foundation class library.
14. The DCS shall be integrating our software with a coast guard segment. Although we are separate executables, we are able to pass data messages to each other.
15. Microsoft Visual C++ is new to the developers.
16. Communication within the organization could use improvement.

2. Risk Analysis

The next phase of the risk assessment framework involves adjusting the quantitative risk values of the probability, consequence, and timeframe attributes to meet the needs of the specific software development project. Described below are the adjusted quantitative values for each of the risk attributes.

a. *Probability of Occurrence*

The probability of occurrence can be quantified using Probability Criteria Table shown below. There is a direct correlation between the probability of occurrence and the quantitative criteria risk value. The higher the probability of occurrence is, the higher the quantitative criteria risk value.

Quantitative Value	Probability
10	> 90%
9	80-89%
8	70-79%
7	60-69%
6	50-59%
5	40-49%
4	30-39%
3	20-29%
2	10-19%
1	< 9%

Table 7. Re-host Probability of Occurrence

b. Consequence of Occurrence

The consequence of occurrence can be quantified based on the unmet requirements, lost time or the cost incurred by the risk occurring. The table below can be used as a guide, but must be modified to meet the needs of individual software development project.

Quantitative Value	Lost Functionality	Lost Time
10	> 9 %	>18 days
9	8-9 %	16-17 days
8	7-8 %	14-15 days
7	6-7 %	12-13days
6	5-6 %	10-11 days
5	4-5 %	8-9 days
4	3-4 %	6-7 days
3	2-3 %	4-5 days
2	1-2 %	1-3 days
1	< 1 %	< 1 day

Table 8. Re-host Consequence of Occurrence

c. *Timeframe of Risk*

The time frame in which the risk could occur affects the importance that the project manager or decision-maker places on the risk. The table below provides a guide to quantifying the time frame so that each risk may be systematically ranked.

Quantitative Value	Lost Time
2	< 1 day
1.75	1-3 days
1.50	4-8 days
1.25	9-15 days
1.00	16-30 days
0.75	31-40 days
0.50	41-60 days
0.25	> 60 days

Table 9. Re-host Risk Time Frame

3. Risk Assessment Results From The Display Control Subsystem Port

The risk assessment framework identified and prioritized nine risks in the software development project. The project manager can utilize these results by applying proper resources to resolve or mitigate each of them.

Described below in *Table 10. Prioritized Risk Summary* shows each of the risk attributes were quantified for each of the risks. Each attribute was multiplied together to attain the total risk value. The risks were ranked in order from most to least severe.

Priority	Description of Risk			
	Total Quantitative Value	Probability	Severity	Timeframe
1	Microsoft NT Workstation operating system does not currently support dual login, therefore a creating a creating a process to handle dual login will be a risk.			
	Total Quantitative Value	Probability	Severity	Timeframe
	54	9	6	1
2	Meeting the requirement of having multiple applications running simultaneously sharing control of the Real Time Subsystem will pose a risk.			
	Total Quantitative Value	Probability	Severity	Timeframe
	54	9	6	1
3	All the graphical user interfaces (X11, Xt, and Motif widgets/gadgets) will have to be rewritten.			
	Total Quantitative Value	Probability	Severity	Timeframe
	50	10	5	1
4	The personnel are lacking expertise in the areas of Win32 APIs, Microsoft NT architecture and Microsoft Foundation Class Libraries.			
	Total Quantitative Value	Probability	Severity	Timeframe
	48	8	6	1
5	The system calls in the two operating systems are different. Therefore, the design and algorithms will have to be modified accordingly.			
	Total Quantitative Value	Probability	Severity	Timeframe
	40	10	4	1
6	The DCS is integrating software with a coast guard segment. Although we are separate executables, we do pass data messages to each other.			
	Total Quantitative Value	Probability	Severity	Timeframe
	36	6	6	1

7	The current application is written in C, but we will be implementing some functionality in C++ using Microsoft Foundation class library.			
	Total Quantitative Value	Probability	Severity	Timeframe
	32	8	4	1
8	The DCS is using a Microsoft operating system			
	Total Quantitative Value	Probability	Severity	Timeframe
	30	10	3	1
9	Microsoft Visual C++ is new to the developers.			
	Total Quantitative Value	Probability	Severity	Timeframe
	24	8	3	1
10	Communication within the organization could use improvement.			
	Total Quantitative Value	Probability	Severity	Timeframe
	24	6	4	1
11	The DCS is using a Department of Defense common operating environment. We are dependent on them to produce a reliable operating environment.			
	Total Quantitative Value	Probability	Severity	Timeframe
	20	10	2	1
12	The DCS is using Microsoft's Visual C++ compiler.			
	Total Quantitative Value	Probability	Severity	Timeframe
	20	10	2	1
13	Memory management will have to be implemented differently using Win 32 APIs.			
	Total Quantitative Value	Probability	Severity	Timeframe
	20	10	2	1
14	Process handling will have to be implemented differently using Win 32 APIs.			
	Total Quantitative Value	Probability	Severity	Timeframe
	20	10	2	1

15	Socket handling will have to be implemented differently using Win 32 APIs.			
	Total Quantitative Value	Probability	Severity	Timeframe
	20	10	2	1
16	Script files will have to be implemented differently.			
	Total Quantitative Value	Probability	Severity	Timeframe
	10	10	1	1

Table 10. Prioritized Risk Summary

B. MAPPING, CHARTING, GEODESY AND IMAGERY CHART SERVER

The Mapping, Charting, Geodesy and Imagery Server software project is in its initial coding phase. Once complete, the application will be utilized by many of the US Navy's surface ships. The MCG&I Server shall manage and distribute National Imagery and Mapping Agency (NIMA) data residing on a Redundant Array of Independent Disks (RAID) to various clients throughout the ship.

The following section describes the risks identified from the risk assessment framework. This section also describes how the framework is used to systematically analyze and prioritize these risks.

1. Identified Risks For the Risk-Based Questionnaire

The risk-based questionnaire helped to identify twelve risks. Of the twelve risks, three were removed due to duplication. Please see *Appendix D: Results of the Risk Questionnaire for the MCG&I Server* for a detailed version description of the questionnaire process. Described below are the sixteen identified risks at this point in the development process.

1. The requirements stability is a risk because the role of the MCG&I Server has been modified in the past, and it is still an issue in the working group meetings.
2. The backup and availability requirement must be in accordance to the changing Electronic Chart Display and Information System, Navy (ECDIS-N) requirements.
3. There may be new requirements in the specifications due to inadequacies of the third party, Joint Mapping Toolkit (JMTK), Application Program Interfaces (APIs).
4. There may be new requirements in the specifications due to inadequacies of the third party, Vector Data Update (VDU), Application Program Interfaces (APIs).

5. Security requirements are subject to change due to the classifications of the LANs that the server resides on.
6. The personnel do not have an expertise in the client/server area of programming.
7. Communication within the organization could use improvement.
8. The scheduled release date of the JMTK APIs tends to continuously shift to the right.
9. The scheduled release date of the VDU APIs tends to continuously shift to the right.

2. Risk Analysis

The next phase of the risk assessment framework involves adjusting the quantitative risk values of the probability, consequence, and timeframe attributes to meet the needs of the specific software development project. Described below are the adjusted quantitative values for each of the risk attributes.

a. Probability of Occurrence

The probability of occurrence can be quantified using Probability Criteria Table shown below. There is a direct correlation between the probability of occurrence and the quantitative criteria risk value. The higher the probability of occurrence is, the higher the quantitative criteria risk value. All of the risks in the project had low probabilities, therefore the scale had to be adjusted accordingly.

Quantitative Value	Probability
10	> 45%
9	40-45%
8	35-39%
7	30-34%
6	25-29%
5	20-24%
4	15-19%
3	10-14%
2	5-9%
1	< 5%

Table 11. MCG&I Server Probability of Occurrence

b. Consequence of Occurrence

The consequence of occurrence can be quantified based on the unmet requirements, lost time or the cost incurred by the risk occurring. The table below has been modified to meet the needs of individual software development project.

Quantitative Value	Lost Functionality	Lost Time
10	> 17 %	>18 days
9	15-17%	16-17 days
8	13-15 %	14-15 days
7	11-13 %	12-13days
6	9-11 %	10-11 days
5	7-9 %	8-9 days
4	5-7 %	6-7 days
3	3-5 %	4-5 days
2	1-3 %	1-3 days
1	< 1 %	< 1 day

Table 12. MCG&I Server Consequence of Occurrence

c. *Timeframe of Risk*

The time frame in which the risk could occur can be quantified based on the severity that the project manager or decision-maker places on the schedule. The table has been calibrated to meet the needs of this software development project.

Quantitative Value	**Time Before Occurrence**
1.25	< 15 day
1.00	15-90 days
0.75	> 90 days

Table 13. MCG&I Server Risk Time Frame

3. Risk Assessment results from the MCG&I Server

The risk assessment framework identified and prioritized nine risks in the software development project. The project manager can utilize these results by applying proper resources to resolve or mitigate each of them.

Described below in *Table 14. Prioritized Risk Summary* shows how each of the risk attributes were quantified for each of the risks. Each attribute was multiplied together to attain the total risk value. The risks were ranked in order from most to least severe.

Priority	Description of Risk			
	Total Quantitative Value	Probability	Severity	Timeframe
1	The requirements stability is a risk because the role of the MCG&I Server has been modified in the past, and it is still an issue in the working group meetings.			
	Total Quantitative Value 60	Probability 6	Severity 10	Timeframe 1
2	There may be new requirements in the specifications due to inadequacies of the third party, Joint Mapping Toolkit (JMTK), Application Program Interfaces (APIs).			
	Total Quantitative Value 40	Probability 4	Severity 10	Timeframe 1
3	The scheduled release date of the VDU APIs tends to continuously shift to the right.			
	Total Quantitative Value 36	Probability 6	Severity 6	Timeframe 1
4	Security requirements are subject to change due to the classifications of the LANs that the server resides on			
	Total Quantitative Value 30	Probability 3	Severity 10	Timeframe 1
5	The scheduled release date of the JMTK APIs tends to continuously shift to the right.			
	Total Quantitative Value 27	Probability 3	Severity 9	Timeframe 1
6	Communication within the organization could use improvement.			
	Total Quantitative Value 25	Probability 4	Severity 5	Timeframe 1.25

7	The backup and availability requirement must be in accordance to the changing Electronic Chart Display and Information System, Navy (ECDIS-N) requirements.			
	Total Quantitative Value	Probability	Severity	Timeframe
	18	2	9	1
8	There may be new requirements in the specifications due to inadequacies of the third party, Vector Data Update (VDU), Application Program Interfaces (APIs).			
	Total Quantitative Value	Probability	Severity	Timeframe
	12	3	6	1
9	The personnel do not have an expertise in the client / server area of programming.			
	Total Quantitative Value	Probability	Severity	Timeframe
	10	4	2	1.25

Table 14. Prioritized Risks Summary

V. CONCLUSION

One of the key ingredients to enhance the probability of a successful software development project is applying proper risk management. Although it is mandated for Department of Defense programs in several directives and initiatives, risk management implementations can be improved. One of the causes for the deficient risk management implementations is the lack of a systematic and disciplined methodology to implement the risk policies. The risk assessment framework presented in this thesis was developed to provide the solution.

Unfortunately, risk has some attributes similar to cancer. First, new risks can appear unexpectedly throughout the lifecycle of the development project. Second, early detection is key to resolving and mitigating risks. Third, if risks go undetected, they will often lead to the termination of the project. Fortunately for risks, there are remedies. The wonder drug is the risk assessment framework. The key features that make the framework effective and efficient include its flexibility and the ability to be applied in a consistent systematic manner. These features allow the project manager or decision-maker to use it on any size and during any stage of the software development project. This leads to early detection of identify risks and expands the time that the project manager or decision maker will have to mitigate and resolve the risks.

This thesis describes a risk assessment framework that enhances the project manager or decision-maker's risk management abilities by providing a systematic and disciplined method to identify, analyze and prioritize risks in any software development project. To validate the validity of the risk assessment framework, it was implemented on two software development projects currently under development at the Space and Naval Warfare Systems Center, San Diego. The results were very positive. The risk assessment for the Display Control Subsystem re-host project identified and prioritized sixteen risks. The risk assessment for the MCG&I project identified and prioritized nine risks. These prioritized risks enhance the project mangers or decision maker's ability to apply proper resource in order to resolve or mitigate each one of them. In return, the project has a much better chance to succeed.

Although these prioritized risks will help the project manager apply resources properly to mitigate and resolve the current risks, it is essential that this framework be applied continuously throughout the life cycle of the development project. Although the scope of the thesis ends at this point, it is recommended that a risk resolution plan be developed and each of the prioritized risks are monitored and tracked. Risk management is a continuous process, therefore the periodically application the risk assessment framework is recommended throughout the lifecycle of the project.

Overall, the risk assessment framework enhances the project manager or decision-maker's ability as a risk manager, which ultimately makes them more successful in all software development projects that they lead.

APPENDIX A. SOFTWARE ENGINEERING INSTITUTE'S RISK TAXONOMIC GROUP DEFINITIONS

The Software Engineering Institute's Software Development Risk Taxonomy is a well structure classification of areas within a software development project. It consists of three main classes including: product engineering; development engineering; and program constraints. These classes are further divided into elements, and each of the elements is further divided into attributes. Please refer to Table A-1. Software Development Risk Taxonomy [Ref. 6] for the complete picture of the taxonomy. Following the table is a comprehensive explanation of each attribute, element, and class of the taxonomy written by the SEI [Ref. 6].

A. Product Engineering

1. Requirements
   a. Stability
   b. Completeness
   c. Clarity
   d. Validity
   e. Feasibility
   f. Precedent
   g. Scale
2. Design
   a. Functionality
   b. Difficulty
   c. Interfaces
   d. Performance
   e. Testability
   f. Hardware Constraints
   g. Non-Developmental Software
3. Code and Unit Test
   a. Feasibility
   b. Testing
   c. Coding/Implementation
4. Integration Testing
   a. Environment
   b. Product
   c. System
5. Engineering Specialties
   a. Maintainability
   b. Reliability
   c. Safety
   d. Security
   e. Human Factors
   f. Specifications

B. Development Environment

1. Development Process
   a. Formality
   b. Suitability
   c. Process Control
   d. Familiarity
   e. Product Control
2. Development System
   a. Capacity
   b. Suitability
   c. Usability
   d. Familiarity
   e. Reliability
   f. System Support
   g. Deliverability
3. Management Process
   a. Planning
   b. Project Organization
   c. Management Experience
   d. Program Interfaces
4. Management Methods
   a. Monitoring
   b. Personnel Management
   c. Quality Assurance
   d. Configuration Management
5. Work Environment
   a. Quality Attitude
   b. Cooperation
   c. Communication
   d. Morale

C. Program Constraints

1. Resources
   a. Schedule
   b. Staff
   c. Budget
   d. Facilities
2. Contract
   a. Type of Contract
   b. Restrictions
   c. Dependencies
3. Program Interfaces
   a. Customer
   b. Associate Contractors
   c. Subcontractors
   d. Prime Contractor
   e. Corporate Management
   f. Vendors
   g. Politics

Table A-1. Software Development Risk Taxonomy [Ref. 6]

A. Product Engineering

Product engineering refers to the system engineering and software engineering activities involved in creating a system that satisfies specified requirements and customer expectations. These activities include system and software requirements analysis and specification, software design and implementation, integration of hardware and software components, and software and system test.

The elements of this class cover traditional software engineering activities. They comprise those technical factors associated with the deliverable product itself, independent of the processes or tools used to produce it or the constraints imposed by finite resources or external factors beyond program control.

Product engineering risks generally result from requirements that are technically difficult or impossible to implement, often in combination with inability to negotiate relaxed requirements or revised budgets and schedules; from inadequate analysis of requirements or design specification; or from poor quality design or coding specifications.

1. Requirements

Attributes of the requirements element cover both the quality of the requirements specification and also the difficulty of implementing a system that satisfies the requirements.

a) Stability

The stability attribute refers to the degree to which the requirements are changing and the possible effect changing requirements and external interfaces will have on the quality, functionality, schedule, design, integration, and testing of the product being built.

The attribute also includes issues that arise from the inability to control rapidly changing requirements. For example, impact analyses may be inaccurate because it is impossible to define the baseline against which the changes will be implemented.

b) Completeness

Missing or incompletely specified requirements may appear in many forms, such as a requirements document with many functions or parameters "to be defined"; requirements that are not specified adequately to develop acceptance criteria, or inadvertently omitted requirements. When missing information is not supplied in a timely manner, implementation may be based on contractor assumptions that differ from customer expectations.

When customer expectations are not documented in the specification, they are not budgeted into the cost and schedule.

c) Clarity

This attribute refers to ambiguously or imprecisely written individual requirements that are not resolved until late in the development phase. This lack of a mutual contractor and customer understanding may require re-work to meet the customer intent for a requirement.

d) Validity

This attribute refers to whether the aggregate requirements reflect customer intentions for the product. This may be affected by misunderstandings of the written requirements by the contractor or customer, unwritten customer expectations or requirements, or a specification in which the end user did not have inputs.

This attribute is affected by the completeness and clarity attributes of the requirements specifications, but refers to the larger question of the system as a whole meeting customer intent.

e) Feasibility

The feasibility attribute refers to the difficulty of implementing a single technical or operational requirement, or of simultaneously meeting conflicting requirements. Sometimes two requirements by themselves are feasible, but together are not; they cannot both exist in the same product at the same time.

Also included is the ability to determine an adequate qualification method for demonstration that the system satisfies the requirement.

f) Precedent

The precedent attribute concerns capabilities that have not been successfully implemented in any existing systems or are beyond the experience of program personnel or of the company. The degree of risk depends on allocation of additional schedule and budget to determine the feasibility of their implementation; contingency plans in case the requirements are not feasible as stated; and flexibility in the contract to allocate implementation budget and schedule based on the outcome of the feasibility study.

Even when unprecedented requirements are feasible, there may still be a risk of underestimating the difficulty of implementation and committing to an inadequate budget and schedule.

g) Scale

This attribute covers both technical and management challenges presented by large complex systems development.

Technical challenges include satisfaction of timing, scheduling and response requirements, communication among processors, complexity of system integration, analysis of inter-component dependencies and impact due to changes in requirements.

Management of a large number of tasks and people introduces a complexity in such areas as project organization, delegation of responsibilities, communication among management and peers, and configuration management.

2. Design

The attributes of the design element cover the design and feasibility of algorithms, functions or performance requirements, and internal and external product interfaces. Difficulty in testing may begin here with failure to work to testable requirements or to include test features in the design. The following attributes characterize the design element.

a) Functionality

This attribute covers functional requirements that may not submit to a feasible design, or use of specified algorithms or designs without a high degree of certainty that they will satisfy their source requirements. Algorithm and design studies may not have used appropriate investigation techniques or may show marginal feasibility.

b) Difficulty

The difficulty attribute refers to functional or design requirements that may be extremely difficult to realize. Systems engineering may design a system architecture difficult to implement, or requirements analysis may have been based on optimistic design assumptions.

The difficulty attribute differs from design feasibility in that it does not proceed from pre-ordained algorithms or designs.

c) Interfaces

This attribute covers all hardware and software interfaces that are within the scope of the development program, including interfaces between configuration items, and the techniques for defining and managing the interfaces. Special note is taken of non-developmental software and developmental hardware interfaces.

d) Performance

The performance attribute refers to time-critical performance: user and real-time response requirements, throughput requirements, performance analyses, and performance modeling throughout the development cycle.

e) Testability

The testability attribute covers the amenability of the design to testing, design of features to facilitate testing, and the inclusion in the design process of people who will design and conduct product tests.

f) Hardware Constraints

This attribute covers target hardware with respect to system and processor architecture, and the dependence on hardware to meet system and software performance requirements. These constraints may include throughput or memory speeds, real-time response capability, database access or capacity limitations, insufficient reliability, unsuitability to system function, or insufficiency in the amount of specified hardware.

g) Non-Developmental Software

Since non-developmental software (NDS) is not designed to system requirements, but selected as a "best fit," it may not conform precisely to performance, operability or supportability requirements.

The customer may not accept vendor or developer test and reliability data to demonstrate satisfaction of the requirements allocated to NDS. It may then be difficult to produce this data to satisfy acceptance criteria and within the estimated NDS test budget.

Requirements changes may necessitate re-engineering or reliance on vendors for special purpose upgrades.

3. Code and Unit Test

Attributes of this element are associated with the quality and stability of software or interface specifications, and constraints that may present implementation or test difficulties.

a) Feasibility

The feasibility attribute of the code and unit test element addresses possible difficulties that may arise from poor design or design specification or from inherently difficult implementation needs.

For example, the design may not have quality attributes such as module cohesiveness or interface minimization; the size of the modules may contribute complexity; the design may not be specified in sufficient detail, requiring the programmer to make assumptions or design decisions during coding; or the design and interface specifications may be changing, perhaps without an approved detailed design baseline; and the use of developmental hardware may make an additional contribution to inadequate or unstable interface specification. Or, the nature of the system itself may aggravate the difficulty and complexity of the coding task.

b) Unit Test

Factors affecting unit test include planning and preparation and also the resources and time allocated for test.

Constituents of these factors are: entering unit test with quality code obtained from formal or informal code inspection or verification procedures; pre-planned test cases that have been verified to test unit requirements; a test bed consisting of the necessary hardware or emulators, and software or simulators; test data to satisfy the planned test; and sufficient schedule to plan and carry out the test plan.

c) Coding/Implementation

This attribute addresses the implications of implementation constraints. Some of these are: target hardware that is marginal or inadequate with regard to speed, architecture, memory size or external storage capacity; required implementation languages or methods; or differences between the development and target hardware.

4. Integration and Test

This element covers integration and test planning, execution, and facilities for both the contractual product and for the integration of the product into the system or site environment.

a) Environment

The integration and test environment includes the hardware and software support facilities and adequate test cases reflecting realistic operational scenarios and realistic test data and conditions.

This attribute addresses the adequacy of this environment to enable integration in a realistic environment or to fully test all functional and performance requirements.

b) Product

The product integration attribute refers to integration of the software components to each other and to the target hardware and testing of the contractually deliverable product. Factors that may affect this are internal interface specifications for hardware or software, testability of requirements, negotiation of customer agreement on test criteria, adequacy of test specifications and sufficiency of time for integration and test.

c) System

The system integration attribute refers to integration of the contractual product to interfacing systems or sites. Factors associated with this attribute are external interface specifications, ability to faithfully produce system interface conditions prior to site or system integration, access to the system or site being interfaced to, adequacy of time for testing and associate contractor relationships.

5. Engineering Specialties

The engineering specialty requirements are treated separately from the general requirements element primarily because specialists who may not be full time on the program often address them. This taxonomic separation is a device to ensure that these specialists are called in to analyze the risks associated with their areas of expertise.

a) Maintainability

Maintainability may be impaired by poor software architecture, design, code, or documentation resulting from undefined or un-enforced standards, or from neglecting to analyze the sys-tem from a maintenance point of view.

b) Reliability

System reliability or availability requirements may be affected by hardware not meeting its reliability specifications or system complexity that aggravates difficulties in meeting recovery timelines. Reliability or availability requirements allocated to software may be stated in absolute terms, rather than as separable from hardware and independently testable.

c) Safety

This attribute addresses the difficulty of implementing allocated safety requirements and also the potential difficulty of demonstrating satisfaction of requirements by faithful simulation of the unsafe conditions and corrective actions. Full demonstration may not be possible until the system is installed and operational.

d) Security

This attribute addresses lack of experience in implementing the required level of system security that may result in underestimation of the effort required for rigorous verification methods, certification and accreditation, and secure or trusted development process logistics; developing to unprecedented requirements; and dependencies on delivery of certified hardware or software.

e) Human Factors

Meeting human factors requirements is dependent on understanding the operational environment of the installed system and agreement with various customer and user factions on a mutual understanding of the expectations embodied in the human factors requirements. It is difficult to convey this understanding in a written specification. Mutual agreement on the human interface may require continuous prototyping and demonstration to various customer factions.

f) Specifications

This attribute addresses specifications for the system, hardware, software, interface, or test requirements or design at any level with respect to feasibility of implementation and the quality attributes of stability, completeness, clarity, and verifiability.

B. Development Environment

The development environment class addresses the project environment and the process used to engineer a software product. This environment includes the development process and system, management methods, and work environment. These environmental elements are characterized below by their component attributes.

1. Development Process

The development process element refers to the process by which the contractor proposes to satisfy the customer's requirements. The process is the sequence of steps the inputs, outputs, actions, validation criteria and monitoring activities—leading from the initial requirement specification to the final delivered product. The development process includes such phases as requirements analysis, product definition, product creation, testing and delivery. It includes both general management processes such as costing, schedule tracking, and personnel assignment, and also project-specific processes such as feasibility studies, design reviews, and regression testing. This element groups risks that result from a development process that is inadequately planned, defined and documented; that is not suited to the activities necessary to accomplish the project goals; and that is poorly communicated to the staff and lacks enforced usage.

a) Formality

Formality of the development process is a function of the degree to which a consistent process is defined, documented and communicated for all aspects and phases of the development.

b) Suitability

Suitability refers to the adequacy with which the selected development model, process, methods, and tools support the scope and type of activities required for the specific program.

c) Process Control

Process control refers not only to ensuring usage of the defined process by program personnel, but also to the measurement and improvement of the process based on observation with respect to quality and productivity goals. Control may be complicated due to distributed development sites.

d) Familiarity

Familiarity with the development process covers knowledge of, experience in, and comfort with the prescribed process.

e) Product Control

Product control is dependent on tractability of requirements from the source specification through implementation such that the product test will demonstrate the source requirements. The change control process makes use of the tractability mechanism in impact analyses and reflects all resultant document modifications including interface and test documentation.

2. Development System

The development system element addresses the hardware and software tools and supporting equipment used in product development. This includes computer-aided software engineering tools, simulators, compilers, test equipment and host computer systems.

a) Capacity

Risks associated with the capacity of the development system may result from too few workstations, insufficient processing power or database storage, or other inadequacies in equipment to support parallel activities for development, test, and support activities.

b) Suitability

Suitability of the development system is associated with the degree to which it is supportive of the specific development models, processes, methods, procedures, and activities required and selected for the program. This includes the development, management, documentation, and configuration management processes.

c) Usability

Usability refers to development system documentation, accessibility and workspace, as well as ease of use.

d) Familiarity

Development system familiarity depends on prior use of the system by the company and by project personnel as well as adequate training for new users.

e) Reliability

Development system reliability is a measure of whether the needed components of the development system are available and working properly whenever required by any program personnel.

f) System Support

Development system support involves training in use of the system, access to expert users or consultants, and repair or resolution of problems by vendors.

g) Deliverability

Some contracts require delivery of the development system. Risks may result from neglecting to bid and allocate resources to ensure that the development system meets all deliverable requirements.

3. Management Process

The management process element pertains to risks associated with planning, monitoring, and controlling budget and schedule; with controlling factors involved in defining, implementing, and testing the product; with managing project personnel; and with handling external organizations including the customer, senior management, matrix management and other contractors.

a) Planning

The planning attribute addresses risks associated with developing a well-defined plan that is responsive to contingencies as well as long-range goals and that was formulated with the input and acquiescence of those affected by it. Also addressed are managing according to the plan and formally modifying the plan when changes are necessary.

b) Project Organization

This attribute addresses the effectiveness of the program organization, the effective definition of roles and responsibilities, and the assurance that program personnel understand these roles and lines of authority.

c) Management Experience

This attribute refers to the experience of all levels of managers with respect to management, software development management, the application domain, the scale and complexity of the system and program, the selected development process, and hands-on development of software.

d) Program Interfaces

This attribute refers to the interactions of managers at all levels with program personnel at all levels, and with external personnel such as the customer, senior management and peer managers.

4. Management Methods

This element refers to methods for managing both the development of the product and program personnel. These include quality assurance, configuration management, staff development with respect to program needs and maintaining communication about program status and needs.

a) Monitoring

The monitoring includes the activities of obtaining and acting upon status reports, allocating status information to the appropriate program organizations and maintaining and using progress metrics.

b) Personnel Management

Personnel management refers to selection and training of program members and ensuring that they: take part in planning and customer interaction for their areas of responsibility; work according to plan; and receive the help they need or ask for to carry out their responsibilities.

c) Quality Assurance

The quality assurance attribute refers to the procedures instituted for ensuring both that contractual processes and standards are implemented properly for all program activities, and that the quality assurance function is adequately staffed to perform its duties.

d) Configuration Management

The configuration management (CM) attribute addresses both staffing and tools for the CM function, as well as the complexity of the required CM process. This is with respect to such factors as multiple development and installation sites and product coordination with existing, possibly changing, systems.

5. Work Environment

The work environment element refers to subjective aspects of the environment. Examples of this are the amount of care given to ensuring that people are kept informed of program goals and information, the way people work together, responsiveness to staff inputs and the attitude and morale of the program personnel.

a) Quality Attitude

This attribute refers to the tendency of program personnel to do quality work in general and to conform to specific quality standards for the program and product.

b) Cooperation

The cooperation attribute addresses lack of team spirit among development staff both within and across work groups and the failure of all management levels to demonstrate that best efforts are being made to remove barriers to efficient accomplishment of work.

c) Communication

Risks that result from poor communication are due to lack of knowledge of the system mission, requirements, and design goals and methods, or to lack of information about the importance of program goals to the company or the project.

d) Morale

Risks that result from low morale range across low levels of enthusiasm and thus low performance, productivity or creativity; anger that may result in intentional damage to the project or the product; mass exodus of staff from the project; and a reputation within the company that makes it difficult to recruit.

C. Program Constraints

Program constraints refer to the "externals" of the project. These are factors that may be outside the control of the project but can still have major effects on its success or constitute sources of substantial risk.

1. Resources

This element addresses resources for which the program is dependent on factors outside program control to obtain and maintain. These include schedule, staff, budget, and facilities.

a) Schedule

This attribute refers to the stability of the schedule with respect to internal and external events or dependencies and the viability of estimates and planning for all phases and aspects of the program.

b) Staff

This attribute refers to the stability and adequacy of the staff in terms of numbers and skill levels, their experience and skills in the required technical areas and application domain, and their availability when needed.

c) Budget

This attribute refers to the stability of the budget with respect to internal and external events or dependencies and the viability of estimates and planning for all phases and aspects of the program.

d) Facilities

This attribute refers to the adequacy of the program facilities for development, integration, and testing of the product.

2. Contract

Risks associated with the program contract are classified according to contract type, restrictions, and dependencies.

a) Type of Contract

This attribute covers the payment terms (cost plus award fee, cost plus fixed fee, etc.) and the contractual requirements associated with such items as the Statement of Work, Contract Data Requirements List, and the amount and conditions of customer involvement.

b) Restrictions

Contract restrictions and restraints refer to contractual directives to, for example, use specific development methods or equipment and the resultant complications such as acquisition of data rights for use of non-developmental software.

c) Dependencies

This attribute refers to the possible contractual dependencies on outside contractors or vendors, customer-furnished equipment or software, or other outside products and services.

3. Program Interfaces

This element consists of the various interfaces with entities and organizations outside the development program itself.

a) Customer

The customer attribute refers to the customer's level of skill and experience in the technical or application domain of the program as well as difficult working relationships or poor mechanisms for attaining customer agreement and approvals, not having access to certain customer factions, or not being able to communicate with the customer in a forthright manner.

b) Associate Contractors

The presence of associate contractors may introduce risks due to conflicting political agendas, problems of interfaces to systems being developed by outside organizations, or lack of cooperation in coordinating schedules and configuration changes.

c) Subcontractors

The presence of subcontractors may introduce risks due to inadequate task definitions and subcontractor management mechanisms, or to not transferring subcontractor technology and knowledge to the program or corporation.

d) Prime Contractor

When the program is a subcontract, risks may arise from poorly defined task definitions, complex reporting arrangements, or dependencies on technical or programmatic information.

e) Corporate Management

Risks in the corporate management area include poor communication and direction from senior management as well as non-optimum levels of support.

f) Vendors

Vendor risks may present themselves in the forms of dependencies on deliveries and support for critical system components.

g) Politics

Political risks may accrue from relationships with the company, customer, associate contractors or subcontractors, and may affect technical decisions.

APPENDIX B. SOFTWARE ENGINEERING INSTITUTE'S RISK TAXONOMY-BASED QUESTIONNAIRE

The Software Engineering Institute's risk taxonomy-based questionnaire consists of one hundred ninety-four thorough questions that help identify risks within a software development project. Many of the questions have follow-up probe questions that contain issues, concerns and risks to help elicit all the risks in the development project. The questionnaire was developed by the SEI to be a comprehensive guide to cover all aspects for almost any organization to utilize. As a result, certain questions may not pertain to the specific software project during a specific stage of the development life cycle.

A. Product Engineering

1. Requirements

a. Stability

[1] Are the requirements stable?

(No) (1.a) What is the effect on the system?

- Quality
- Functionality
- Schedule
- Integration
- Design
- Testing

[2] Are the external interfaces changing?

b. Completeness

[Are requirements missing or incompletely specified?]

[3] Are there any "To Be Determined" in the specifications?

[4] Are there requirements you know should be in the specification but aren't?

(Yes) (4.a) Will you be able to get these requirements into the system?

[5] Does the customer have unwritten requirements/expectations?

(Yes) (5.a) Is there a way to capture these requirements?

[6] Are the external interfaces completely defined?

c. Clarity

[Are requirements unclear or in need of interpretation?]

[7] Are you able to understand the requirements as written?

(No) (7.a) Are the ambiguities being resolved satisfactorily?

(Yes) (7.b) There are no ambiguities or problems of interpretation?

d. Validity

[Will the requirements lead to the product the customer has in mind?]

[8] Are there any requirements that may not specify what the customer really wants?

(Yes) (8.a) How are you resolving this?

[9] Do you and the customer understand the same thing by the requirements?

(Yes) (9.a) Is there a process by which to determine this?

[10] How do you validate the requirements?

- Prototyping
- Analysis
- Simulations

e. Feasibility

[Are requirements infeasible from an analytical point of view?]

[11] Are there any requirements that are technically difficult to implement?

(Yes) (11.a) What are they?

(Yes) (11.b) Why are they difficult to implement?

(No) (11.c) Were feasibility studies done for these requirements?

(Yes) (11.c.1) How confident are you of the assumptions made in the studies?

f. Precedent

[Do requirements specify something never done before, or that your company has not done before?]

[12] Are there any state-of-the-art requirements?

- Technologies
- Methods
- Languages
- Hardware

(No) (12.a) Are any of these new to you?

(Yes) (12.b) Does the program have sufficient knowledge in these areas?

(No) (12.b.1) Is there a plan for acquiring knowledge in these areas?

g. Scale

[Do requirements specify a product larger, more complex, or requiring a larger organization than in the experience of the company?]

[13] Is the system size and complexity a concern?

(No) (13.a) Have you done something of this size and complexity before?

[14] Does the size require a larger organization than usual for your company?

2. Design

a. Functionality

[Are there any potential problems in meeting functionality requirements?]

[15] Are there any specified algorithms that may not satisfy the requirements?

(No) (15.a) Are any of the algorithms or designs marginal with respect to meeting requirements?

[16] How do you determine the feasibility of algorithms and designs?

- Prototyping
- Modeling
- Analysis
- Simulation

b. Difficulty

[Will the design and/or implementation be difficult to achieve?]

[17] Does any of the design depend on unrealistic or optimistic assumptions?

[18] Are there any requirements or functions that are difficult to design?

(No) (18.a) Do you have solutions for all the requirements?

(Yes) (18.b) What are the requirements?

- Why are they difficult?

c. Interfaces

[Are the internal interfaces (hardware and software) well defined and controlled?]

[19] Are the internal interfaces well defined?

- Software-to-software
- Software-to-hardware

[20] Is there a process for defining internal interfaces?

(Yes) (20.a) Is there a change control process for internal interfaces?

[21] Is hardware being developed in parallel with software?

(Yes) (21.a) Are the hardware specifications changing?

(Yes) (21.b) Have all the interfaces to software been defined?

(Yes) (21.c) Will there be engineering design models that can be used to test the software?

d. Performance

[Are there stringent response time or throughput requirements?]

[22] Are there any problems with performance?

- Throughput
- Scheduling asynchronous real-time events
- Real-time response
- Recovery timelines
- Response time
- Database response, contention, or access

[23] Has a performance analysis been done?

(Yes) (23.a) What is your level of confidence in the performance analysis?

(Yes) (23.b) Do you have a model to track performance through design and implementation?

e. Testability

[Is the product difficult or impossible to test?]

[24] Is the software going to be easy to test?

[25] Does the design include features to aid testing?

[26] Do the testers get involved in analyzing requirements?

f. Hardware Constraints

[Are there tight constraints on the target hardware?]

[27] Does the hardware limit your ability to meet any requirements?

- Architecture
- Memory capacity
- Throughput
- Real-time response

- Response time
- Recovery timelines
- Database performance
- Functionality
- Reliability
- Availability

g. Non-Developmental Software

[Are there problems with software used in the program but not developed by the program?]

If re-used or re-engineered software exists

[28] Are you reusing or re-engineering software not developed on the program?

(Yes) (28.a) Do you foresee any problems?

- Documentation
- Performance
- Functionality
- Timely delivery
- Customization

If COTS software is being used

[29] Are there any problems with using COTS (commercial off-the-shelf) software?

- Insufficient documentation to determine interfaces, size, or performance
- Poor performance
- Requires a large share of memory or database storage
- Difficult to interface with application software
- Not thoroughly tested
- Not bug free
- Not maintained adequately

- Slow vendor response

[30] Do you foresee any problem with integrating COTS software updates or revisions?

3. Code and Unit Test

a. Feasibility

[Is the implementation of the design difficult or impossible?]

[31] Are any parts of the product implementation not completely defined by the design specification?

[32] Are the selected algorithms and designs easy to implement?

b. Testing

[Are the specified level and time for unit testing adequate?]

[33] Do you begin unit testing before you verify code with respect to the design?

[34] Has sufficient unit testing been specified?

[35] Is there sufficient time to perform all the unit testing you think should be done?

[36] Will compromises be made regarding unit testing if there are schedule problems?

c. Coding/Implementation

[Are there any problems with coding and implementation?]

[37] Are the design specifications in sufficient detail to write the code?

[38] Is the design changing while coding is being done?

[39] Are there system constraints that make the code difficult to write?

- Timing
- Memory
- External storage

[40] Is the language suitable for producing the software on this program?

[41] Are there multiple languages used on the program?

(Yes) (41.a) Is there interface compatibility between the code produced by the different compilers?

[42] Is the development computer the same as the target computer?

(No) (42.a) Are there compiler differences between the two?

If developmental hardware is being used

[43] Are the hardware specifications adequate to code the software?

[44] Are the hardware specifications changing while the code is being written?

4. Integration and Test

a. Environment

[Is the integration and test environment adequate?]

[45] Will there be sufficient hardware to do adequate integration and testing?

[46] Is there any problem with developing realistic scenarios and test data to demonstrate any requirements?

- Specified data traffic
- Real-time response
- Asynchronous event handling
- Multi-user interaction

[47] Are you able to verify performance in your facility?

[48] Does hardware and software instrumentation facilitate testing?

(Yes) (48.a) Is it sufficient for all testing?

b. Product

[Is the interface definition inadequate, facilities inadequate, time insufficient?]

[49] Will the target hardware be available when needed?

[50] Have acceptance criteria been agreed to for all requirements?

(Yes) (50.a) Is there a formal agreement?

[51] Are the external interfaces defined, documented, and base lined?

[52] Are there any requirements that will be difficult to test?

[53] Has sufficient product integration been specified?

[54] Has adequate time been allocated for product integration and test?

If COTS

[55] Will vendor data be accepted in verification of requirements allocated to COTS products?

(Yes) (55.a) Is the contract clear on that?

c. System

[System integration uncoordinated, poor interface definition, or inadequate facilities?]

[56] Has sufficient system integration been specified?

[57] Has adequate time been allocated for system integration and test?

[58] Are contractors involved with the integration team?

[59] Will the product be integrated into an existing system?

(Yes) (59.a) Is there a parallel cutover period with the existing system?

(No) (59.a.1) How will you guarantee the product will work correctly when integrated?

[60] Will system integration occur on customer site?

5. Engineering Specialties

a. Maintainability

[Will the implementation be difficult to understand or maintain?]

[61] Does the architecture, design, or the code create any maintenance difficulties?

[62] Are the maintenance people involved early in the design?

[63] Is the product documentation adequate for maintenance by an outside organization?

b. Reliability

[Are the reliability or availability requirements difficult to meet?]

[64] Are reliability requirements allocated to the software?

[65] Are availability requirements allocated to the software?

(Yes) (65.a) Are recovery timelines any problem?

c. Safety

[Are the safety requirements infeasible and not demonstrable?]

[66] Are safety requirements allocated to the software?

(Yes) (66.a) Do you see any difficulty in meeting the safety requirements?

[67] Will it be difficult to verify satisfaction of safety requirements?

d. Security

[Are the security requirements more stringent than the current state of the practice or program experience?]

[68] Are there unprecedented or state-of-the-art security requirements?

[69] Is it an Orange Book system?

[70] Have you implemented this level of security before?

e. Human Factors

[Will the system be difficult to use because of poor human interface definition?]

[71] Do you see any difficulty in meeting the Human Factors requirements?

(No) (71.a) How are you ensuring that you will meet the human interface requirements?

If prototyping

(Yes) (71.a.1) Is it a throwaway prototype?

(No) (71.a.1a) Are you doing evolutionary development?

(Yes) (71.a.1a.1) Are you experienced in this type of development?

(Yes) (71.a.1a.2) Are interim versions deliverable?

(Yes) (71.a.1a.3) Does this complicate change control?

f. Specifications

[Is the documentation adequate to design, implement, and test the system?]

[72] Is the software requirements specification adequate to design the system?

[73] Are the hardware specifications adequate to design and implement the software?

[74] Are the external interface requirements well specified?

[75] Are the test specifications adequate to fully test the system?

If in or past implementation phase

[76] Are the design specifications adequate to implement the system?

- Internal interfaces

B. Development Environment

1. Development Process

a. Formality

[Will the implementation be difficult to understand or maintain?]

[77] Is there more than one development model being used?

- Spiral
- Waterfall
- Incremental

(Yes) (77.a) Is coordination between them a problem?

[78] Are there formal, controlled plans for all development activities?

- Requirements analysis

- Design
- Code
- Integration and test
- Installation
- Quality assurance
- Configuration management

(Yes) (78.a) Do the plans specify the process well?

(Yes) (78.b) Are developers familiar with the plans?

b. Suitability

[Is the process suited to the development model, e.g., spiral, and prototyping?]

[79] Is the development process adequate for this product?

[80] Is the development process supported by a compatible set of procedures, methods, and tools?

c. Process Control

[Is the software development process enforced, monitored, and controlled using metrics? Are distributed development sites coordinated?]

[81] Does everyone follow the development process?

(Yes) (81.a) How is this insured?

[82] Can you measure whether the development process is meeting your productivity and quality goals?

If there are distributed development sites

[83] Is there adequate coordination among distributed development sites?

d. Familiarity

[Are the project members experienced in use of the process? Do all staff members understand the process?]

[84] Are people comfortable with the development process?

e. Product Control

[Are there mechanisms for controlling changes in the product?]

[85] Is there a requirements traceability mechanism that tracks requirements from the source specification through test cases?

[86] Is the traceability mechanism used in evaluating requirement change impact analyses?

[87] Is there a formal change control process?

(Yes) (87.a) Does it cover all changes to base lined requirements, design, code, and documentation?

[88] Are changes at any level mapped up to the system level and down through the test level?

[89] Is there adequate analysis when new requirements are added to the system?

[90] Do you have a way to track interfaces?

[91] Are the test plans and procedures updated as part of the change process?

2. Development System

a. Capacity

[Is there sufficient workstation processing power, memory, or storage capacity?]

[92] Are there enough workstations and processing capacity for all staff?

[93] Is there sufficient capacity for overlapping phases, such as coding, integration and test?

b. Suitability

[Does the development system support all phases, activities, and functions?]

[94] Does the development system support all aspects of the program?

- Requirements analysis

- Performance analysis
- Design
- Coding
- Test
- Documentation
- Configuration management
- Management tracking
- Requirements traceability

c. Usability

[How easy is the development system to use?]

[95] Do people find the development system easy to use?

[96] Is there good documentation of the development system?

d. Familiarity

[Is there little prior company or project member experience with the development system?]

[97] Have people used these tools and methods before?

e. Reliability

[Does the system suffer from software bugs, down-time, insufficient built-in back-up?]

[98] Is the system considered reliable?

- Compiler
- Development tools
- Hardware

f. System Support

[Is there timely expert or vendor support for the system?]

[99] Are the people trained in use of the development tools?

[100] Do you have access to experts in use of the system?

[101] Do the vendors respond to problems rapidly?

g. Deliverability

[Are the definitions and acceptance requirements defined for delivering the development system to the customer not budgeted? HINT: If the participants are confused about this, it is probably not an issue from a risk perspective.]

[102] Are you delivering the development system to the customer?

(Yes) (102.a) Have adequate budget, schedule, and resources been allocated for this deliverable?

3. Management Process

a. Planning

[Is the planning timely, technical leads included, contingency planning done?]

[103] Is the program managed according to the plan?

(Yes) (103.a) Do people routinely get pulled away to fight fires?

[104] Is re-planning done when disruptions occur?

[105] Are people at all levels included in planning their own work?

[106] Are there contingency plans for known risks?

(Yes) (106.a) How do you determine when to activate the contingencies?

[107] Are long-term issues being adequately addressed?

b. Project Organization

[Are the roles and reporting relationships clear?]

[108] Is the program organization effective?

[109] Do people understand their own and others' roles in the program?

[110] Do people know who has authority for what?

c. Management Experience

[Are the managers experienced in software development, software management, the application domain, the development process, or on large programs?]

[111] Does the program have experienced managers?

- Software management
- Hands-on software development
- With this development process
- In the application domain
- Program size or complexity

d. Program Interfaces

[Is there poor interface with customer, other contractors, senior and/or peer managers?]

[112] Does management communicate problems up and down the line?

[113] Are conflicts with the customer documented and resolved in a timely manner?

[114] Does management involve appropriate program members in meetings with the customer?

- Technical leaders
- Developers
- Analysts

[115] Does management work to ensure that all customer factions are represented in decisions regarding functionality and operation?

[116] Is it good politics to present an optimistic picture to the customer or senior management?

4. Management Methods

a. Monitoring

[Are management metrics defined and development progress tracked?]

[117] Are there periodic structured status reports?

(Yes) (117.a) Do people get a response to their status reports?

[118] Does appropriate information get reported to the right organizational levels?

[119] Do you track progress versus plan?

(Yes) (119.a) Does management have a clear picture of what is going on?

b. Personnel Management

[Are project personnel trained and used appropriately?]

[120] Do people get trained in skills required for this program?

(Yes) (120.a) Is this part of the program plan?

[121] Do people get assigned to the program who do not match the experience profile for your work area?

[122] Is it easy for program members to get management action?

[123] Are program members at all levels aware of their status versus plan?

[124] Do people feel it's important to keep to the plan?

[125] Does management consult with people before making decisions that affect their work?

[126] Does program management involve appropriate program members in meetings with the customer?

- Technical leaders
- Developers
- Analysts

c. Quality Assurance

[Are there adequate procedures and resources to assure product quality?]

[127] Is the software quality assurance function adequately staffed on this program?

[128] Do you have defined mechanisms for assuring quality?

(Yes) (128.a) Do all areas and phases have quality procedures?

(Yes) (128.b) Are people used to working with these procedures?

d. Configuration Management

[Are the change procedures or version control, including installation site(s), adequate?]

[129] Do you have an adequate configuration management system?

[130] Is the configuration management function adequately staffed?

[131] Is coordination required with an installed system?

(Yes) (131.a) Is there adequate configuration management of the installed system?

(Yes) (131.b) Does the configuration management system synchronize your work with site changes?

[132] Are you installing in multiple sites?

(Yes) (132.a) Does the configuration management system provide for multiple sites?

5. Work Environment

a. Quality Attitude

[Is there a lack of orientation toward quality work?]

[133] Are all staff levels oriented toward quality procedures?

[134] Does schedule get in the way of quality?

b. Cooperation

[Is there a lack of team spirit? Does conflict resolution require management intervention?]

[135] Do people work cooperatively across functional boundaries?

[136] Do people work effectively toward common goals?

[137] Is management intervention sometimes required to get people working together?

c. Communication

[Is there poor awareness of mission or goals, poor communication of technical information among peers and managers?]

[138] Is there good communication among the members of the program?

- Managers
- Technical leaders
- Developers
- Testers
- Configuration management
- Quality assurance

[139] Are the managers receptive to communication from program staff?

(Yes) (139.a) Do you feel free to ask your managers for help?

(Yes) (139.b) Are members of the program able to raise risks without having a solution in hand?

[140] Do the program members get timely notification of events that may affect their work?

(Yes) (140.a) Is this formal or informal?

d. Morale

[Is there a non-productive, non-creative atmosphere? Do people feel that there is no recognition or reward for superior work?]

[141] How is morale on the program?

(No) (141.a) What is the main contributing factor to low morale?

[142] Is there any problem keeping the people you need?

C. Program Constraints

1. Resources

a. Schedule

[Is the schedule inadequate or unstable?]

[143] Has the schedule been stable?

[144] Is the schedule realistic?

(Yes) (144.a) Is the estimation method based on historical data?

(Yes) (144.b) Has the method worked well in the past?

[145] Is there anything for which adequate schedule was not planned?

- Analysis and studies
- QA
- Training
- Maintenance courses and training
- Capital equipment
- Deliverable development system

[146] Are there external dependencies that are likely to impact the schedule?

b. Staff

[Is the staff inexperienced, lacking domain knowledge, lacking skills, or understaffed?]

[147] Are there any areas in which the required technical skills are lacking?

- Software engineering and requirements analysis method
- Algorithm expertise
- Design and design methods
- Programming languages
- Integration and test methods
- Reliability
- Maintainability
- Availability
- Human factors
- Configuration management
- Quality assurance

- Target environment
- Level of security
- COTS
- Reuse software
- Operating system
- Database
- Application domain
- Performance analysis
- Time-critical applications

[148] Do you have adequate personnel to staff the program?

[149] Is the staffing stable?

[150] Do you have access to the right people when you need them?

[151] Have the program members implemented systems of this type?

[152] Is the program reliant on a few key people?

[153] Is there any problem with getting cleared people?

c. Budget

[Is the funding insufficient or unstable?]

[154] Is the budget stable?

[155] Is the budget based on a realistic estimate?

(Yes) (155.a) Is the estimation method based on historical data?

(Yes) (155.b) Has the method worked well in the past?

[156] Have features or functions been deleted as part of a design-to-cost effort?

[157] Is there anything for which adequate budget was not allocated?

- Analysis and studies
- QA
- Training
- Maintenance courses
- Capital equipment
- Deliverable development system

[158] Do budget changes accompany requirement changes?

(Yes) (158.a) Is this a standard part of the change control process?

d. Facilities

[Are the facilities adequate for building and delivering the product?]

[159] Are the development facilities adequate?

[160] Is the integration environment adequate?

2. Contract

a. Type of Contract

[Is the contract type a source of risk to the program?]

[161] What type of contract do you have? (Cost plus award fee, fixed price...)

(161a) Does this present any problems?

[162] Is the contract burdensome in any aspect of the program?

- Statement of Work
- Specifications
- Data Item Descriptions
- Contract parts
- Excessive customer involvement

[163] Is the required documentation burdensome?

- Excessive amount
- Picky customer
- Long approval cycle

b. Restrictions

[Does the contract cause any restrictions?]

[164] Are there problems with data rights?

- COTS software
- Developmental software

- Non-developmental items

c. Dependencies

[Does the program have any dependencies on outside products or services?]

[165] Are there dependencies on external products or services that may affect the product, budget, or schedule?

- Associate contractors
- Prime contractor
- Subcontractors
- Vendors or suppliers
- Customer furnished equipment or software

3. Program Interfaces

a. Customer

[Are there any customer problems such as: lengthy document-approval cycle, poor communication, and inadequate domain expertise?]

[166] Is the customer approval cycle timely?

- Documentation
- Program reviews
- Formal reviews

[167] Do you ever proceed before receiving customer approval?

[168] Does the customer understand the technical aspects of the system?

[169] Does the customer understand software?

[170] Does the customer interfere with process or people?

[171] Does management work with the customer to reach mutually agreeable decisions in a timely manner?

- Requirements understanding
- Test criteria
- Schedule adjustments

- Interfaces

[172] How effective are your mechanisms for reaching agreements with the customer?

- Working groups (contractual?)
- Technical interchange meetings (contractual?)

[173] Are all customer factions involved in reaching agreements?

(Yes) (173.a) Is it a formally defined process?

[174] Does management present a realistic or optimistic picture to the customer?

b. Associate Contractors

[Are there any problems with associate contractors such as inadequately defined or unstable interfaces, poor communication, or lack of cooperation?]

[175] Are the external interfaces changing without adequate notification, coordination, or formal change procedures?

[176] Is there an adequate transition plan?

(Yes) (176.a) Do all contractors and site personnel support it?

[177] Is there any problem with getting schedules or interface data from associate contractors?

(No) (177.a) Are they accurate?

c. Subcontractors

[Is the program dependent on subcontractors for any critical areas?]

[178] Are there any ambiguities in subcontractor task definitions?

[179] Is the subcontractor reporting and monitoring procedure different from the program's reporting requirements?

[180] Is subcontractor administration and technical management done by a separate organization?

[181] Are you highly dependent on subcontractor expertise in any areas?

[182] Is subcontractor knowledge being transferred to the company?

[183] Is there any problem with getting schedules or interface data from subcontractors?

d. Prime Contractor

[Is the program facing difficulties with its Prime contractor?]

[184] Are your task definitions from the Prime ambiguous?

[185] Do you interface with two separate prime organizations for administration and technical management?

[186] Are you highly dependent on the Prime for expertise in any areas?

[187] Is there any problem with getting schedules or interface data from the Prime?

e. Corporate Management

[Is there a lack of support or micro management from upper management?]

[188] Does program management communicate problems to senior management?

(Yes) (188.a) Does this seem to be effective?

[189] Does corporate management give you timely support in solving your problems?

[190] Does corporate management tend to micro-manage?

[191] Does management present a realistic or optimistic picture to senior management?

f. Vendors

[Are vendors responsive to programs needs?]

[192] Are you relying on vendors for deliveries of critical components?

- Compilers
- Hardware
- COTS

g. Politics

[Are politics causing a problem for the program?]

[193] Are politics affecting the program?

- Company
- Customer
- Associate contractors
- Subcontractors

[194] Are politics affecting technical decisions?

APPENDIX C. RESULTS OF THE RISK QUESTIONNAIRE FOR THE DISPLAY CONTROL SUBSYSTEM PORT

The Software Engineering Institute Risk Based Questionnaire was applied to the DCS Re-host project in order to identify risks within the organization. Explanations to each of the SEI questions are described below.

A. Product Engineering

1. Requirements

    a. Stability

        The requirements will remain constant for the application that is ported to the windows environment.

    b. Completeness

        All the requirements are complete, and are fully documented in the Software Requirements Specification.

    c. Clarity

        The requirements are clearly written and the application running on the HP workstation is fully functional.

    d. Validity

        The customer has seen the application running on the HP workstation.

    e. Feasibility

        [Risk #1] Microsoft NT Workstation operating system does not support dual login.

    f. Precedent

        [Risk #2] The personnel are lacking expertise in the areas of Win32 APIs, Microsoft NT architecture, and Microsoft Foundation Class Libraries.

    g. Scale

        The scale of the re-host project does not pose a problem.

2. Design

    a. Functionality

[Risk #3] The system calls in the two operating systems are different. Therefore, the design and algorithms will have to be modified accordingly.

b. Difficulty

[Risk #4] Meeting the requirement of having multiple applications running simultaneously sharing control of the Real Time Subsystem will pose a problem.

c. Interfaces

[Risk #5] Interfacing with the Coast Guard developed interface may pose a problem. They are in the initial stages of re-hosting their software.

d. Performance

Performance issues will not be a problem.

e. Testability

The application will be easy to test.

f. Hardware Constraints

The PC with dual processors (NT supported) shall be able to capable of performing faster than the HP (single RISC CPU) that it is replacing.

g. Non-Developmental Software

[Risk #6] We are using a Microsoft operating system.

[Risk #7] We are using a Department of Defense common operating environment. We are dependent on them to produce a reliable operating environment.

[Risk #8] We are using Microsoft's Visual C++ compiler.

3. Code and Unit Test

a. Feasibility

[Risk #9] The different system calls of the two operating systems will cause the algorithms to be modified when ported.

b. Testing

They have a competent test team. Test procedures have already been developed for the HP version of the application.

c. Coding/Implementation

[Risk #10] Memory management will have to be implemented differently using Win 32 APIs.

[Risk #11] Process handling will have to be implemented differently using Win 32 APIs.

[Risk #12] Socket handling will have to be implemented differently using Win 32 APIs.

[Risk #13] Script files will have to be implemented differently.

[Risk #14] All the graphical user interfaces (X11, Xt, and Motif widgets/gadgets) will have to be rewritten.

[Risk #15] The current application is written in C, but we will be implementing some functionality in C++ using Microsoft Foundation class library.

4. Integration and Test

a. Environment

The integration and test environment is good.

b. Product

[Risk #16] We are integrating our software with a coast guard segment. Although we are separate executables, we do pass data messages to each other.

c. System

A well-defined Interface Design Document exists between the applications that we are interfacing with.

5. Engineering Specialties

a. Maintainability

The product is in the design phase, but will be written in a manner that is well documented and easy to maintain.

b. Reliability

[Risk #17] We are using a Microsoft product.

c. Safety

Safety issues are properly addressed.

d. Security

The application shall remain unclassified, and shall meet security specifications.

e. Human Factors

The application will properly address the human factor issues.

f. Specifications

The documentation is adequate to design, implement, and test the system.

B. Development Environment

1. Development Process

a. Formality

The organization follows many of the policies laid out be the Software Engineering Institute's Capability Maturity Model.

b. Suitability

The development process adequate for this product

c. Process Control

The configuration management is excellent within the organization.

d. Familiarity

The project members are experienced in use of the process.

e. Product Control

The configuration management is excellent within our organization.

2. Development System

a. Capacity

There is sufficient workstation processing power, memory or storage capacity.

b. Suitability

The development system supports all phases, activities, and functions.

c. Usability

The development system is easy to use.

d. Familiarity

[Risk #18] – Microsoft Visual C++ is new to the developers.

e. Reliability

[Risk #19] – Microsoft Products are not very reliable.

f. System Support

There is timely expert or vendor support for the system

g. Deliverability

The definitions and acceptance requirements are defined for delivering the development system to the customer.

3. Management Process

a. Planning

The planning is good.

b. Project Organization

The roles and reporting relationships clear within the organization.

c. Management Experience

The managers are experienced in software development.

d. Program Interfaces

There is good communication between the project manager and customer.

4. Management Methods

a. Monitoring

The management metrics are defined and development progress tracked effectively.

b. Personnel Management

The project personnel are trained and used appropriately.

c. Quality Assurance

There are adequate procedures and resources to assure product quality.

5. Work Environment

a. Quality Attitude

The group has good work ethics.

b. Cooperation

Cooperation within the organization is good.

c. Communication

[Risk # 20] Communication within the organization could use improvement.

d. Morale

Moral is good within the group.

C. Program Constraints

1. Resources

a. Schedule

The schedule is adequate.

b. Staff

As mentioned earlier, the staff is lacking expertise in the areas of Win32 APIs, Microsoft NT architecture, and Microsoft Foundation Class Libraries.

c. Budget

The budget is adequate.

d. Facilities

The facilities are adequate.

2. Contract

a. Type of Contract

Not applicable.

b. Restrictions

Not applicable.

c. Dependencies

There are no other dependencies at this time.

3. Program Interfaces

a. Customer

The customer has a good understanding of the application.

b. Associate Contractors

There are no associate contractors at this time.

c. Subcontractors

Subcontractors are not posing a risk at this time.

d. Prime Contractor

The prime contractor poses no risk at this time.

e. Corporate Management

The support from upper management is excellent.

f. Vendors

Vendors are very supportive.

g. Politics

Politics do not pose a risk at this time.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. RESULTS OF THE RISK QUESTIONNAIRE FOR THE MCG&I SERVER

The Software Engineering Institute Risk Based Questionnaire was applied to the MCG&I Server project in order to identify risks within the organization. Explanations to each of the SEI questions are described below.

A. Product Engineering

1. Requirements

a. Stability

[Risk #1] – The requirements stability is a risk because the role of the MCG&I Server has been modified in the past, and it is still an issue in the working group meetings.

[Risk #2] - The backup and availability requirement must be in accordance to the changing Electronic Chart Display and Information System, Navy (ECDIS-N) requirements.

b. Completeness

[Risk #3] - There may be new requirements in the specifications due to inadequacies of the third party, Joint Mapping Toolkit (JMTK), Application Program Interfaces (APIs).

[Risk #4] - There may be new requirements in the specifications due to inadequacies of the third party, Vector Data Update (VDU), Application Program Interfaces (APIs).

c. Clarity

[Risk #5] – Security requirements are subject to change due to the classifications of the LANs that the server resides on.

d. Validity

The system will meet the customer's intent.

e. Feasibility

The current requirements are feasible.

f. Precedent

[Risk #6] – The personnel do not have an expertise in the client / server area of programming.

g. Scale

The scale of the server project does not pose a problem

2. Design

a. Functionality

[Risk #7] – The server will depend on third party APIs to meet the desired functionality.

b. Difficulty

The server design has been simplified to meet the requirements.

c. Interfaces

The server shall use commercial off the shelf components, and third party software API's.

d. Performance

There are no stringent response time or throughput requirements.

e. Testability

The application shall be easy to test.

f. Hardware Constraints

There are not any tight constraints on the target hardware.

g. Non-Developmental Software

[Risk #8] – The server is relying on third party API's.

3. Code and Unit Test

a. Feasibility

The server coding is feasible.

b. Testing

The specified level and time for unit testing is adequate.

c. Coding/Implementation

Other than plugging in the third party API, the coding should straightforward.

4. Integration and Test

a. Environment

The integration and test environment is adequate.

b. Product

The interface definition, facilities and timeframe are adequate.

c. System

The server shall run as a separate process on top of the Defense Information Infrastructure (DII) Common Operating Environment (COE).

5. Engineering Specialties

a. Maintainability

The implementation be will be easy to understand and maintain.

b. Reliability

The system shall meet reliability and availability requirements.

c. Safety

We properly address the safety issues.

d. Security

[Risk # 9] – The LAN classification that this application resides on shall pose a security issue.

e. Human Factors

The application will properly address the human factor issues.

f. Specifications

The documentation is adequate to design, implement, and test the system.

B. Development Environment

1. Development Process

a. Formality

The organization follows many of the policies laid out be the Software Engineering Institute's Capability Maturity Model.

b. Suitability

The development process adequate for this product

c. Process Control

The configuration management is excellent within the organization.

d. Familiarity

The project members are experienced in use of the process.

e. Product Control

The configuration management is excellent within our organization.

2. Development System

a. Capacity

There is sufficient workstation processing power, memory, or storage capacity.

b. Suitability

The development system supports all phases, activities, and functions.

c. Usability

The development system is easy to use.

d. Familiarity

The developers are familiar coding on Hewlett Packard workstations running HP-UX 10.20.

e. Reliability

The system does not suffer from software bugs, downtime or insufficient built-in back up.

f. System Support

There is timely expert or vendor support for the system

g. Deliverability

The definitions and acceptance requirements are defined for delivering the development system to the customer.

3. Management Process

  a. Planning

    The planning is good.

  b. Project Organization

    The roles and reporting relationships clear within the organization.

  c. Management Experience

    The managers are experienced in software development.

  d. Program Interfaces

    There is good communication between the project manager and customer.

4. Management Methods

  a. Monitoring

    The management metrics are defined and development progress tracked effectively.

  b. Personnel Management

    The project personnel are trained and used appropriately.

  c. Quality Assurance

    There are adequate procedures and resources to assure product quality.

5. Work Environment

  a. Quality Attitude

    The group has good work ethics.

  b. Cooperation

    Cooperation within the organization is good.

  c. Communication

    [Risk # 10] Communication within the organization could use improvement.

  d. Morale

    Moral is good within the group.

C. Program Constraints

1. Resources

   a. Schedule

      [Risk #11] - The scheduled release date of the JMTK APIs tends to continuously shift to the right.

      [Risk #12] - The scheduled release date of the VDU APIs tends to continuously shift to the right.

   b. Staff

      As mentioned earlier, the staff is inexperienced in the client / server implementation skills.

   c. Budget

      The budget is adequate.

   d. Facilities

      The facilities are adequate.

2. Contract

   a. Type of Contract

      Not applicable.

   b. Restrictions

      Not applicable.

   c. Dependencies

      We are utilizing third party APIs to implement the MCG&I server.

3. Program Interfaces

   a. Customer

      The customer has a good understanding of the application.

   b. Associate Contractors

      There are no associate contractors at this time.

   c. Subcontractors

      Subcontractors are not posing a risk at this time.

d. Prime Contractor

The prime contractor poses no risk at this time.

e. Corporate Management

The support from upper management is excellent.

f. Vendors

Vendors are very supportive.

g. Politics

Politics do not pose a risk at this time.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. The Standish Group International *Charting the Seas of Information Technology,* Dennis, Mass., 1994.

2. Johnson, J. *Chaos: The dollar drain of IT project failures. Application Development Trends* 2, 1, pp. 41-47, 1995.

3. Brooks, F.P. *Mythical Man-Month.* Massachusetts, Addison-Wesley, 1995.

4. Department of Defense, "Defense Acquisition," Department of Defense Directives 5000.1, March 1996.

5. Department of Defense, " Software Development and Documentation," MIL-Standards-498, December 1994.

6. Car, M. J.; Konda, S. L.; Monarch, I.; Ulrich, F. C.; Walker, C. A. *Taxonomy-Based Risk Identification* (CMU/SEI –93-TR-6). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1993.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST